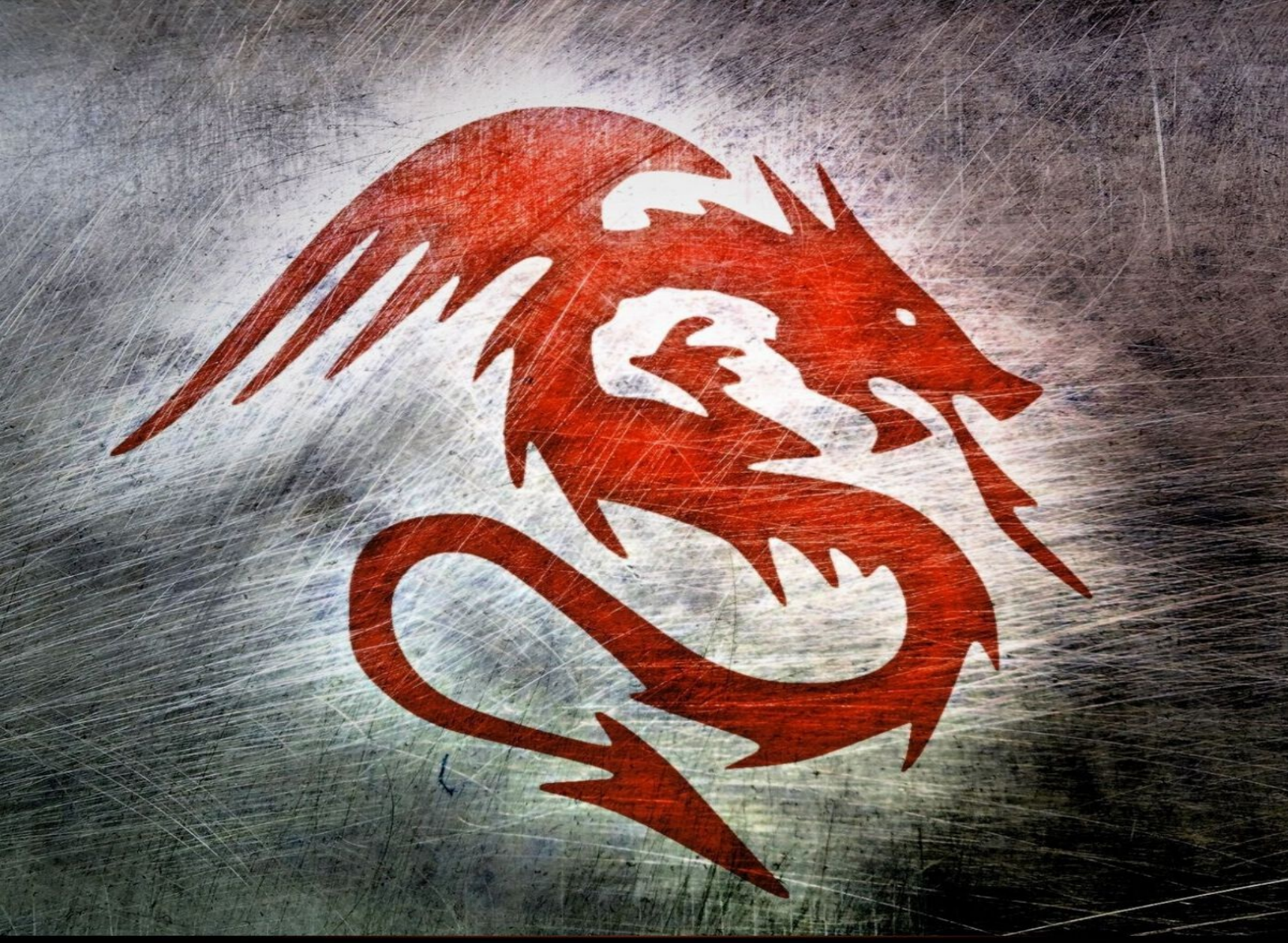
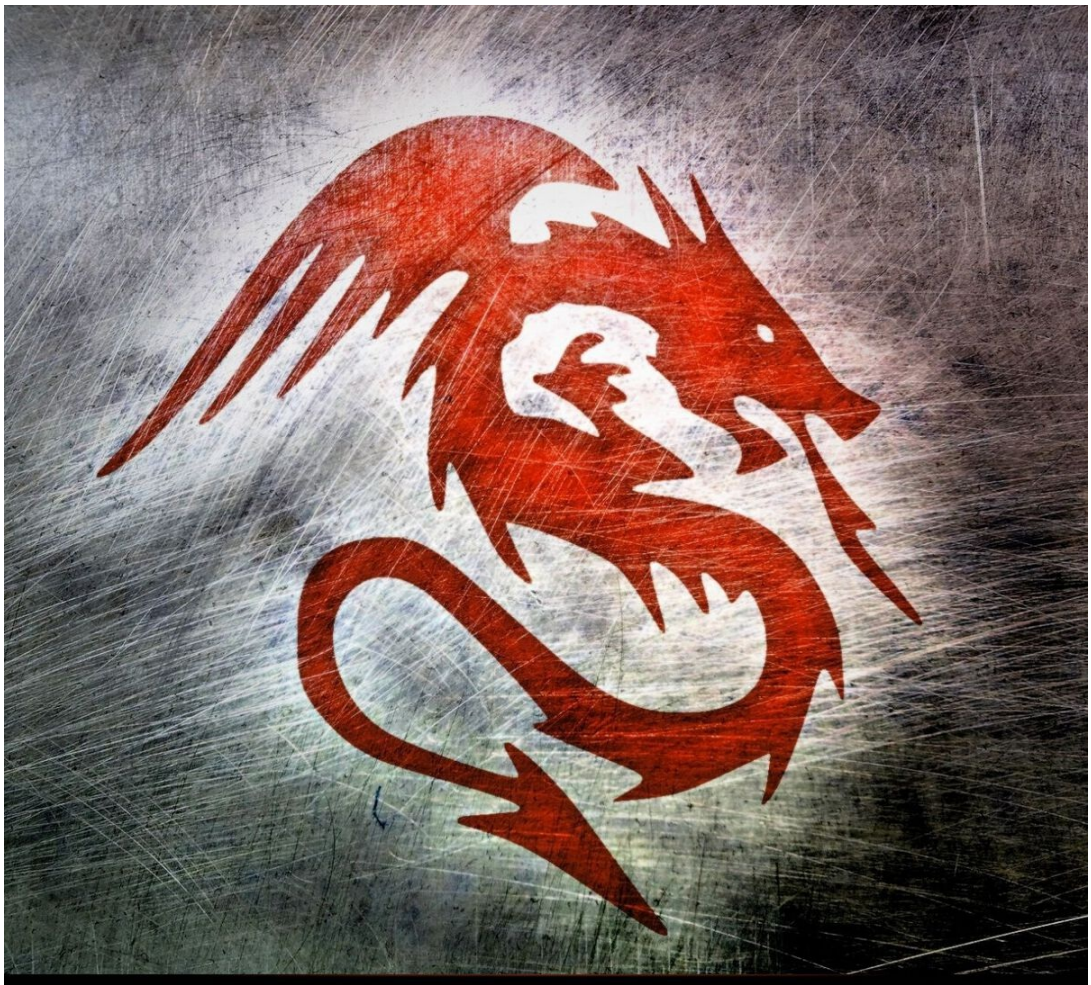


HACKING

ALEX WAGNER

ROOTING & JAILBREAKING



HACKING

ALEX WAGNER

ROOTING & JAILBREAKING

HACKING

Rooting & Jailbreaking

by

ALEX WAGNER

Copyright

Disclaimer

This Book is produced with the goal of providing information that is as accurate and reliable as possible. Regardless, purchasing this Book can be seen as consent to the fact that both the publisher and the author of this book are in no way experts on the topics discussed within and that any recommendations or suggestions that are made herein are for entertainment purposes only.

Professionals should be consulted as needed before undertaking any of the action endorsed herein.

Under no circumstances will any legal responsibility or blame be held against

the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

This declaration is deemed fair and valid by both the American Bar Association and the Committee of Publishers Association and is legally binding throughout the United States.

The information in the following pages is broadly considered to be a truthful and accurate account of facts, and as such any inattention, use or misuse of the information in question by the reader will render any resulting actions solely under their purview. There are no scenarios in which the publisher or the original author of this work can be in any fashion deemed liable for any hardship or damages that may befall the reader or anyone else after undertaking information described herein.

Additionally, the information in the following pages is intended only for informational purposes and should thus be thought of as universal. As befitting its nature, it is presented without assurance regarding its continued validity or interim quality. Trademarks that are mentioned are done without written consent and can in no way be considered an endorsement from the trademark holder.

Introduction

To begin with, I would like to give a few major points on what this book is will cover. The tools that I will describe in this book are available for both white hat and black hat hacking. Once applied the outcome will be the same in both cases. Nevertheless, it can lead to a terrible situation for the person using such hacking tools in any unauthorized manner, which might cause system damage or any system outage.

If you attempt to use any of this tools on a network without being authorized and you disturb or damage any systems, that would be considered illegal black hat hacking. So, I would like to encourage all readers to deploy any tool described in this book for <u>WHITE HAT USE ONLY</u> .

Anything legally authorized to help people or companies to find vulnerabilities and identify potential risks is fine. All tools I will describe, you should use for improving security posture only. I should sound a warning here. If you are eager to learn about hacking and penetration testing, it's recommended to build a home lab and practice using these tools in an isolated network that you have full control over, and it's not connected to any production environment or the internet.

If you use these tools for black hat purposes and you get caught, it will be entirely on you, and you will have no one to blame. So, again I would highly recommend you stay behind the lines, and anything you do should be completely legit and fully authorized.

Lastly, if you are not sure about anything that you are doing and don't have a clue on the outcome, ask your manager or **DO NOT DO IT** .

This book is for education purposes. It is for those who are interested in learning and knowing what is really behind the curtains and would like to become an IT professional or white hat hackers.

In addition to legal issues, before using any of the tools, it is recommended that you have the fundamental knowledge of networking concepts.

Intended Audience

This book is designed for anyone who wishes to become an IT Professional, specifically in the field of Ethical Hacking. This book is written in everyday

English; therefore, no technical background is necessary.

If you are a complete beginner to Informational Technology or Information Security in the realm of Ethical Hacking, the contents in this book will deliver all the answers you have in mind on topics revolving around hacking mobile devices.

Considering that you are preparing to become an Ethical Hacker, IT Security Analyst, IT Security Engineer, or a Cybersecurity Specialist, yet still in doubt and want to know about mobile devices, you will find this book extremely useful.

Not only you will learn key concepts and methodologies revolving around Hacking mobile devices, but you will also learn what key Technologies you should be aware of. If you are genuinely interested in becoming an Ethical Hacker, this book is for you.

Assuming that you are preparing to become an Information Security Professional, this book will undoubtedly provide great details that will benefit you as you enter this industry. The contents of this book are revolving around many topics, such as security mindset, daily security tasks and activities. Still, the main focus of this book is to help you understand how mobile devices work, how they can be exposed, and what security measurements you can deploy to prevent them.

If you are planning to become a Developer or Programmer, this book will help you to understand the Security aspects of applications security, but this is not a technical book on coding.

The content of this book will be an excellent material for anyone interested in Ethical Hacking or Cybersecurity. Let’s get cracking!

Introduction to Mobile Attack Vectors

Welcome to my new Ethical Hacking Book on Hacking Mobile Devices. We're going to be focusing on the mobile device attack vector. Mobile devices are tiny, yet smartphones are more powerful than many computer systems, and the rate at which mobile devices are coming out and applications that are being supported on these devices is mind overwhelming.

Hence I'm going to take you through several different aspects of the mobile platform and how we can hack them, or looking at it from a hacker's perspective. We'll take a look at the effect that these mobile devices have on us in the aspect of the attack vector, meaning how many different ways an attacker can get to them.

We'll then break it down into a platform-based environment where we'll look at hacking Android devices. Then we'll look at hacking iOS platforms.

Then we'll take a look at hacking other mobile platforms too, but this book will focus mainly on the two dominant platforms that compete against each other, which is Android and IOS.

Afterward, we'll take a look at how we can manage these environments and what's the best way that we can maintain the environment because this is going to open up a whole new attack surface for us.

We'll look at MDMs aka Mobile Device Management systems. We'll highlight some of the aspects that MDMs need to make sure that they cover. We'll also take a look at guidelines and tools that we'll use to help secure down this environment.

We'll then take a look at malware at the mobile level as well as mobile payments. We're also going to look at the different avenues as far as the vector is concerned, including how bad is our current environment in terms of mobile device security. And lastly, we'll also take a look at terms that you should get used to and add to your vocabulary.

The fact that mobile devices are moving, as far as cells and implementation, is moving faster than anything we've ever seen in technology. Currently, there are over 5 billion mobile devices in the world. That's enough devices for every person living on this planet.

According to statistics, by 2023, mobile owners will increase to 7.33 billion. Google also announced that more Google searches are done via mobile

platforms than desktop platforms, which proves the point that we're making that transition into the mobile environment and moving further away from the traditional desktop and laptop.

Hence it's worse than you can imagine as far as the security is concerned when it comes to mobile platforms. According to Kensington Infographic, there is one laptop stolen every 53 seconds, while over 70 million mobile devices are lost each year, and only about 7 percent of them are recovered.

The other issue is that when we talk to IT people, they admit that it's their number one weakness because they don't know how to secure mobile devices.

There are so many different platforms to cover. Besides Android and Apple, you also have Blackberry, Microsoft, and everybody wants to use something different. Of those devices that are deployed, stats are showing us that 25% of the mobile devices are receiving security threats every day.

This is the new platform that attackers are going after. As mobile devices get more and more popular, as far as malware is concerned and its threat, there will be a continuous increase in terms of risks.

You can understand why, when you start thinking about from a hacker perspective. The attacker will think like this; “Let's see, I want to create a piece of malware.” “Do I want to do it for Windows 7, Windows 10, Android or IOS?” “Where are my numbers?”

Here's something else that might be interesting to you. Of the Trojans and malware that are out there, we have seen an increase of 17 times. To add that number in the mix of the continuously increasing malware, a majority of it is in banking trojans and malware.

In the meanwhile what we do with our phones? Well, we are doing banking and accounting. As far as the increase in vulnerabilities is concerned, according to AV.test.org, there was 3.14 million Android malware development and close to 60K IOS malware development.

As you see, Android is not alone in this game, simply because IOS is extremely popular with end-users. Most end-users don't understand security. 78% of all apps fail primary security threats and the stats show us that 59% of the IT admins have no idea what to do as far as detecting mobile threats and how to fix them.

Because of BYOD, aka “bring your device”, in the average enterprise. Currently, there are over 1000 malicious apps are installed on employee devices. When you

get down to it, think about what we do on these devices.

Back in the days, we only used to talk. Now we do all kinds of things. We'd send text messages; we post photos, we do our banking, we do email, travel, music and so on.

A single device allows us to do all these different things, while previously we had to have a separate device to do separate items, such as a typewriter for documents, or a physical map to figure out how to get from one place to another, or Discman for our music and so on. Either way, let’s move on and look at some terminologies that you should be aware of.

Table of Contents

Chapter 1 Common mobile platform terminologies

To discuss some of the most common terminologies that you should know when it comes to the mobile platform. It can be overwhelming to think about how many devices are out there and how we're going to control them. The answer to that question is that we're going to do this the same way we do it with desktops.

We have a little bit of a learning curve here. Part of that learning curve is to understand specific terminologies. First of all, let's look at what's important and to whom. Several different stakeholders are involved with mobile platforms, and certain items are essential to each stakeholder that may be a threat to other stakeholders.

For example, one of the stakeholders will be the mobile network operators, and these are the people that are in charge of selling the device and supporting the services for specific devices.

What's important to them is to support the users and giving them an environment that is easy to use. The issue here is that while the technology triangle, as we move towards ease-of-use, we lose functionality and security.

Another stakeholder would be the Original Equipment Manufacturer aka OEM, the people who create the devices. What’s important to them is that they want their devices to be the best, latest and greatest, and there are sometimes assets, as we like to classify them the OEMs want to implement but the carriers won't allow it.

There are many examples of theses where certain Internet Service Providers or ISP-s are going to start allowing phone service over Wi-Fi also known as voice over Wifi. Therefore, that could cause a conflict between the OEM and the carriers.

Another stakeholder would be the app stores, such as Google, Amazon, Apple, and Microsoft marketplace. Being a stakeholder, they want their apps generating revenues for their environment, but by default, for example, Amazon apps can't be installed on an Android device because Google blocks that.

Since Google is the Android OEM in most cases, we have to try to bypass the securities that we can implement or install our Amazon apps. Another known example is Apple because their applications are entirely locked down.

Indeed there are many great features that each of these App stores provides, but there are also some limitations of creating a closed environment. The other issue

that you have is making sure that you get your applications from legitimate app stores.

There are all kinds of app stores out there. In some cases, the OS manufacturer might look at applications as a threat. We also have corporate IT environments or the IT guys that are in charge of these mobile devices, and to them, everything is a threat.

Technically, another stakeholder is an end-user. To an end-user, the threat to them could be that the OS violates their privacy by capturing data. Often people think that Android is an OS that's there to gather statistics and market to individuals.

End-users are threats to the operating system and the manufacturers because if somebody tries to jailbreak or root the phone, it's going to bypass some of the security mechanisms that they put in place.

This is an issue because we've many people involved. Each has different goals set up, therefore that it makes it hard for us to ensure that these environments are secured.

The first terminology I want to discuss is called ROM. A ROM is a modified version of an Android operating system. All our Android devices come with a default ROM by the manufacturer, but we can modify these. Another term is AOSP, aka Android Open Source Project.

This is the Android operating system completely stripped down, and it's given to developers and manufacturers, so they can customize it to create their own ROM.

Next, there is a firmware. Firmware is like a driver, which is a piece of software that makes the hardware function correctly with the OS. Sometimes, we also refer to this as a radio firmware. We also have something called open source. This isn't to be confused with free. Open source is a piece of software that's free to edit and to be used or distributed and shared at no charge. The next term is called bloatware.

This is software or applications that are installed on the mobile devices that you don't necessarily need, that the manufacturer and the OEM have agreed with these pieces of software, they're making a little bit of money in the backside for a reactivation, for example, Dropbox. I mention Dropbox because they pay to get their app installed automatically on your Android device.

Because these are preinstalled, it means that you can't remove them unless we do

something tricky, which we'll discuss shortly, but to give you a hint, it’s called rooting.

The next term is called APK, aka Android Package or Android Application Package, which is the file extension for applications in the Android environment. For example, Dropbox would have a file called “Dropbox.apk.”

Frequently, we download files or apps from a store, and it gets downloaded or installed. But we also have something called “sideloading” that we'll further discuss later on, where you execute the APK off the side or outside of the app stores. That could be dangerous, but we will look at in more detail shortly.

Another term is what we refer to as a “stock ROM”. Stock ROM is the ROM that comes stocked with the device. There is also something called “CyanogenMod”, which is one of the more popular ROMs that are out there.

It is a free, open-source project. It's based on the AOSP, and it's extremely popular with the Android people just because of how stripped-down it is and how customizable it is.

We also have Odex. Odex-es are files that are a collection of parts from applications that are optimized before booting. This speeds up the boot process as it preloads part of the applications into the memory.

We also have OTA, short for Over The Air, that functions as we get our updates OTA. We also have modes. We have both; recovery mode and download mode. Recovery mode is similar to booting up a Windows box in safe mode, where you're going to get an OS that allows you to reimage the phone.

Download modes allow you to push a new image or new software to the phone before it's booted up. It's essential to understand what these two modes are from a security perspective. Lastly, there is another term called “bricking”, which means that you completely messed up your phone to the point that it's now a “brick”.

The term I want to discuss is called jailbreak, which is in the iOS world only. Jailbreak is the process of removing software restrictions that Apple has imposed on a device, and we do it through software exploits. You can jailbreak an iPad, an iPhone, an iTouch, or even second-generation Apple TVs.

What we're doing here is permitting root access to the iOS file system and the manager, which allows us to download additional applications and extensions, as well as themes that are not available through the standard Apple Store.

We also have a term called unlocking. Unlocking is not the same thing as

jailbreaking. There's a lot of misconceptions between the two, but unlocking is allowing you to get the phone to work on a different carrier.

This is not a big issue as it used to be, because the courts have ruled that unlocking your phone is not illegal, so you can technically call upon your providers and say that you are moving over to another carrier. Therefore you need to unlock your phone, and they'll open it for you. Sometimes they might charge for it.

The next term called kernel, which we can use with almost any platform because it means the same thing. It's the computer program that manages the input and output requests from software and translates them into or refers to data processing instructions for the CPU and other components of the system to interpret.

The kernel is a computer program that manages the input and output requests from software, and then it translates them into instructions for the CPU and other components of the device.

When it comes to dealing with the Apple platform, it's important to note that I mentioned jailbreaking before and what it does, but it's not the same thing as rooting.

Rooting is giving me access at the root level, where jailbreaking is, well, if we want to put it in hacking terms, it's a form of privilege escalation.

There are more terms that we could go through on either type of platform, as well as I didn't get into Blackberry or the Microsoft platform, but I wanted to make sure that you have a good understanding of some of the terms that you're going to hear in real life or presented through this book.

Chapter 2 Attack Vectors & Countermeasures

There are more vulnerabilities or attack vectors associated with mobile devices than there are with our standard desktop environments. Some of the same attack vectors that we do experience in the desktop platform, we also see in the mobile environment. For example, malware gets in everywhere on mobile devices, and operating system modifications can cause it.

It can also be caused for application or app modifications, and we'll talk about this a little bit later on, but it's imperative to know where you're getting your apps from because guess what!?

As an attacker, you can take a popular application and add your malware to it and try to distribute it. There are viruses, worms, rootkits and those same type of malware issues that happen on a desktop environment can happen on mobile device environment as well.

The other issue that we have as far as another vector is concerned is going to be the data storage. That could mean internal storage or even your e-mails. E-mails are where we see a lot of security breaches taking place where people are emailing via their mobile devices' sensitive company data.

Another thing that we need to think about when it comes to data storage is copying information. Maybe you get it via an email, and you save it since it comes as an attachment to your local SD card, and then you forward it to your Dropbox. Therefore copying data back and forth is an issue.

When it comes to data storage, something else that you may want to take a look at, and that is encrypting the data storage, so if somebody pulls the SD card, or if they get a hold of your device, most of the devices when you implement encryption, you have to type in a PIN when you turn the phone on.

That's from a cold boot, not from a warm boot. Another attack vector is via social engineering. This is getting people to do things outside of their norm. Social engineering it's a non-technical skill set, or method of getting humans to do things that they normally wouldn't do.

We can use technology to do socially engineering for us too. In either case, there is manipulation that's involved, not at the device level, but the human aspect.

Most social engineering is going to be done through either email or text messages or even multimedia messages; at least from the mobile perspective, that's what we're seeing.

If you want to learn more about social engineering, I suggest you take a look at my other book called Hacking: Social Engineering Attacks, Techniques & Prevention, so that you can master this potent technique.

As far as social engineering being done through social networks, when you think about it, there are advertisements, which are a way of social engineering or a way of implementing social engineering to get you to click on a specific link or to like something.

Better yet, if you can hijack someone's Facebook page, you can post to their friends, and get their attention to click on a specific link, and they click on the link, you infect them all at once. We're also seeing it being done via video chat.

One of the things that we see as far as a considerable increase in attack vectors is concerned, is within the SMS environment, called SMS phishing. The reason why we see this significant increase is because there are so many different opportunities, and even more different types of phishes out there.

People who don't know how devices are designed to work, they can become a victim very quickly. There are many examples where the TXT looks like it's coming from a particular bank, or it comes from a bank that you don't have an account with, but they include links.

Those links could be very easily a link to a phone number to dial a different number than what's being displayed, and once you click on it, they route you to a different phone number.

They're getting so sophisticated that even some of the links that you may click on might take you to a website that looks identical because if you think about what a website is or how it's created, it's just files and images publically available, therefore anyone can duplicate it.

That site can be spoofed, or set up as a phished site, and you can get there because you got a text message from a bank saying that your account was locked out.

There are countermeasures or common sense guidelines, and the first would be that DO NOT CLICK on any links in an SMS or text messages, even if they come from somebody you know.

For all we know, their account could've been breached. Another option is that DO NOT reply to any suspicious SMS without knowing who it is, and check with your financial companies first.

Find out if they sent you a text message when something like an account lockout

was taking place. Some companies typically tell you that they will never email you.

DO NOT call any numbers that might be in a suspicious text message or SMS, either. Another countermeasure is that you should ignore any messages that are urging you to react quickly.

Things like “You're going to miss out!” or “There's an X-amount of dollar Amazon gift card waiting for you if you just do this!” Be careful about those.

Another vector is called “drive-bys”.

Drive-bys are when somebody browses a web page just by visiting the web page. Many websites are created for the soul purpose, so end users download free software, free movies, free music and things like that.

That software could be free Windows OS or also be the type of web plugin that might be infected with a specific malware, waiting for people to download them.

The other issue is that there are older versions of operating systems on many mobile devices. Most vendors do not keep on updating their old operating system, and there are vulnerabilities in them. This is partly because they instead push out their new devices to the market.

Therefore if you don't have the latest generation of iPad or Samsung device, or whichever vendor is your favorite, you are potentially in danger.

Typically updates are fixing security holes in those devices; once there are no more updates, well, you also better move on. Phishing is another attack vector, and this is done through fake emails.

Most people get emails on their devices, therefore sometimes it's hard to realize that the link in the email is invalid as well as we get fake websites, and it is a part of a social engineering environment.

It is getting harder to accomplish phishing type attacks because there are so many companies that have their app. So if you get a link that says, you need to update your password because it's been compromised, if it doesn't launch their app, then you should know that is a bad email.

Because of these upcoming apps form the vendors, it is harder for the attackers to come up with these types of attacks, but it's still possible, especially for the novice.

One of my biggest concerns when it comes to mobile devices is the connections that we are creating, and you need to think about this one. Take your mobile

device, and let's compare it to your desktop. Most phones are always connected to either the cellular network or a Wi-Fi network. There are Bluetooth as well as NFC connections too which we will discuss in the next chapter.

Chapter 3 Introduction to NFC Tags

NFC-s are tags, and the concept behind these small little stickers is that you can program them to do specific things. They've got icons on them to help you determine what you should program them with, but you don't necessarily have to program them.

For example, if it's for a mobile device, you don't have to program it with the ability to turn your phone off and on. You can have it do something completely different. You could have it to install an application, send you to a website that may root your phone, and you could take those tags and put them on top of other legitimate tags that may be out there.

The overall concept is significant because in the aspect of you could take one of these tags and use it for many things. For example, you can take the navigation example and if you were to place it on a dashboard of your car, you could get it to set your phone on it to turn on GPS, open up Google Maps or set up text messaging to send auto-replies like "Sorry, I'm busy driving right now, and unable to text you."

When it comes to attack vectors, this one's probably the biggest and that is the physical aspect. The concept behind this is excellent. You get it on a mobile device, but what if your smartphone gets stolen or you lost it?

Might you have company data or compromising photos or documents on your mobile device? Well, when it comes to the physical side of things, if a bad guy can persuade you to run his program on your device, it's not your device anymore.

Moreover, if a bad guy can alter the OS on your device, it's not your device anymore. Likewise, if a bad guy has unrestricted physical access to your device, once again, it's not your device anymore.

There's no service pack or patch that you can implement to protect yourself from this attack vector. Physical access is just that, it's physical. But you can reduce your risk by taking other steps in protecting your device.

An example of that might be an encryption method. But, nothing is 100% foolproof. Another protection could be customer support-based services. When it comes to these services, one of the significant issues that we have with mobile devices is that most people don’t even know they exist in the first place.

Have you ever phoned up somebody to download an app for your mobile

device? Especially when you forget your password? What do you usually do? Usually, it's a self-help password reset environment.

Typically, what do they ask you when you're doing a self-help password reset? They ask you something that is going to be known to you. For example, what's your Mother’s Maiden name? What's your favorite color? Where were you born? Can I not find out that information?

Have you heard about that breach where all the celebrities, their photos that they took with their cameras, they had no idea that they were being backed up to the iCloud, and attackers were able to get into those photos by doing simple answers to this self-help password reset because the answers or the questions were, what is your dogs name?

Well, they were able to get in and grab the photos and push them out to the internet, making them publically available. It was embarrassing for a lot of those individuals. Therefore the other attack vector is the app itself.

When you download an app, do you download it just because it sounds cool, or do you understand who wrote the app? Or better yet, you go and search for the app, and you see multiple versions of that app, so you might download them all to see which one is better right?

But are you getting the app from the right vendor? If you search Tetris, you'll see more than one Tetris app out there. Also, one of the significant attack vectors when it comes to apps is the permissions that are associated with installing those apps.

There is no reason that some entertainment game that I plan on playing needs access to my contacts or needs access to make a phone call. And sometimes, these permissions are hard to understand because you may be thinking, well, this game doesn't need access to my network.

Well, maybe it does because you're playing an online game or a game against other individuals. Likewise, because computers and mobile devices run a little differently, we have sandboxing environments, where indeed, you can find many vulnerabilities.

What sandboxing does, is that is taking the program and separating it for security purposes to protect the system and the users, by allowing what the application has access to, and what it doesn't have access to.

For example, on computers, an application can access the system RAM, and it can access all of the RAM that's not protected the hard drive information. It can

access your system resources such as the RAM, anything that's in RAM, as well as almost any file that's on the hard drive, except for some of the files that have been protected.

With mobile devices, typically, each application is given its own little virtualized world, but within its own environment, so the app is restricted to access only the things that you've given it permission to access.

The problem here is that some applications could be malicious and exploit loopholes or vulnerabilities of a sandbox, and access to other resources that it usually wouldn't have access to.

When it comes to apps, you must know where it's coming from. If you go and do a search on a torrent site, you can download a bunch of APKs for Android devices or apps for Apple without having to go through the store and paying for them. Then you can install them, known as “sideloading those apps”.

You install it without going through the app store. Yet, I am here to warn you: DON NOT DO IT! Don't trust those sources. Anyone can take an APK, decompile it, put in his malware, recompile it, and push it back up to the torrent sites.

Next, someone going to download it then install it, who might get the paid version for free, but he will also going to get a little bit more than what he thought he is getting. Is it worth it? Do yourself a favor and make sure that the apps are legit.

When we talk about these different platforms and the security vectors, people often ask what's better, open, or closed? Well, when it comes to computers, or anything electronic, there's always going to be weaknesses no matter what you do.

When it comes to an open environment such as Android, the biggest complaint is that it is called fragmentation. Fragmentation is in the aspect of the operating systems.

We already discussed that there are different versions of the operating system out there, and some of them are customized to the hardware vendor. For example, Samsung has something called Touchwiz they put inside of Android, where Motorola may have their interface or their version of Android that's running.

One of the biggest frustrations that many times we'll see is that the same device

just with different carriers running different software. Many times, people think of this as a security issue, but instead of looking at fragmentation as a bad thing, they take it.

Most times, people look at fragmentation as a negative thing, but I also want to think of it as a different word. How about being diverse?

The other side of this argument is a closed environment, which is more restrictive. Apple said that 87% of the active iPhone users are currently running the latest OS.

That makes it easier to protect everyone, right? Because if there is a security vulnerability, everyone gets notified, patch it, and the upgrades get it pushed out to everybody.

That's the very same thing Microsoft is trying to do with Windows 10. It won't matter what platform, whether it's a desktop machine or a new Windows 10 phone or a tablet, they'll be able to fix all their updates at one shot.

That versus Android, which is like at 17%, is running all sorts of Android versions. The reason why the number is so low is because manufacturers decide not to support the older devices.

Additionally, sideloading is taking an application, installing it without going through the official app store. I do understand that some of us not very happy with the fact that Apple won't allow a particular program in their app store because it's restrictive or it does something that Apple doesn't want it to do so that people will sideload those, I get it.

On the Android device side of things, there's a great application out there; for example, there was an app called Adaway that blocked ads, modifying the host file on the device that you wouldn't get pop-ups.

Adaway was also blocked ads from showing up in the games. That's not what the game manufacturer wanted to happen; therefore, they kicked off that particular application off the market.

Hence, we see these third-party app stores show up, which is a warning for you, but if you are going to do any type of sideloading, make sure you understand the source.

Chapter 4 How to Install Android in Hyper-V

In this chapter, I am going to explain how to fire up an Android box inside Hyper-V or VMware virtualized environment. The first thing you need to do is open up a web browser so that you can go and download the Android-x86 project.

You can visit the following website to download the project: https://www.android-x86.org/documentation/virtualbox.html

Once you have downloaded it, go ahead and install it. I will explain in more detail because, as of writing this book, the latest version is Oreo, so I'm going to select I'd like to download the ISO, and you can either grab the ISO for the 32-bit or the 64-bit, it depends on what your platform is.

I'm going to grab the 64-bit myself, which is about 700 MB to download so that it may take a little while, but once you have downloaded your version of ISO, click on continue.

So when you're done downloading, check it in your download directory. It should be called android-x86_64- 8.1.iso. Next, open up either Hyper-V box, or VMware, and select “New Virtual Machine” and hit Next.

You have to name it so you type there Android_VM, then you have to select a location where you want to store it. Once you selected that area, it should create a directory for you, and make sure you select Generation 1.

Do not select Generation 2 because this is an Android operating system that we're installing on a Windows box, therefore select Next. Then it asks for the amount of RAM that it needs, and we need to make sure we give it at least 2 GB of RAM.

Do not select the Dynamic Memory for this virtual machine, and as far as the network connection is concerned, make sure you hook it up to the external network, because it needs to download files while it's installing.

After that, you can put it onto the Lab network if you want to, but for now, leave it as external, and hit Next. It will then tell you that “we're about to create a virtual disk for you called Android_VM.vhdx or whatever you named your new disc then hit Next, so it should prompt you that is about to install an operating system, where you should hit “Yes”, and here you should choose the download location or the folder where you have kept the ISO that you previously downloaded.

Once you have selected the Android image, hit Next, and hit Finished. This will be going to create the virtual machine for you, and now all you have to do is connect to it and start it up, to begin with the installation process.

The first option is where you need to select to do an installation. You don't want to run it from a live CD, but it’s an excellent option to keep in mind in case the machine turns off, you have to wait for it to go back and re-install the image.

It's great if you want to do a live CD, and you want to pop it into a machine, but for our purposes, we're going to do a full-out installation, so hit “Enter” on that option.

Next, select the “Create and Modify partitions” and hit Enter. It will ask if you want to use GPT, and the answer to that is yes, so hit Enter. At this point, if it says that a non-GPT or damaged disk is detected, that's fine, so go ahead and hit any key to continue, because it's a brand-new drive.

Next, you have to provide the amount of disk usage for the Android disc, where you can go ahead and hit the default option. You'll notice at the bottom that you can either select “new” or “load” options.

You can do a new partition, and keep the default size, which is 2 GB, and hit Enter. Next, it will ask you for a new partition name where you can leave it as is and hit Enter because you will see that there's a partition for it.

Now that you created it, you need to tell it to write it so that it writes the partition. You can use the right arrow key, and tap it a couple times to get to Write, and hit Enter.

It will ask you if you are sure you want to do this, so you can type “yes”. Next, you should see the partition there, so go ahead and make sure that's highlighted and hit Enter, and then select what type of format you want to use. Go ahead and use ex4, and hit Enter.

It will ask again: “are you sure you want to write the partition?” Select “Yes”, so it's going to format that partition that will take some time, but once complete, go ahead and hit “Play”.

Next, it will ask if you want to install the boot loader GRUB, so hit Yes, then select that you want to install the system directory as “read-write”, and hit Yes. This means that it will be going to write some files to the virtual disk, and these files are the installation of Android.

Once that it's done, it should say that it has successfully installed Android. The one thing you want to do here is if you just hit to reboot every time you boot up

because you have that ISO listed, you need to get rid of that.

Go under “File”, “Settings”, go to the DVD Drive, and select “None”, and hit OK. Thus you should be able to select “run the Android-x86 project”, but before you do that, you need to change one of the network settings.

By default, Android has a hard time handling network interfaces, at least when they're running off in Hyper-V. Therefore, open up the Settings, and add in a Legacy Network Adaptor, select External, and hit Apply, then hit OK.

Now you can fire it up, by hitting “Start”. It's going to be very similar to what you go through when you turn on an Android tablet or a phone for the first time.

Once it’s up, select English, United States, and go ahead and hit Start. If you don't have any mouse connectivity yet, no worries, because it eventually installs for you so that you can use it.

You want to ensure that your external network is appropriately setup get all the required updates for your operating system installed. You can copy all your data from an existing Android device, or create a new setup.

Hit the Tab key, hit Enter, and if you have an Android, you have to do a Google log-in, and you have to type in your email address or your phone number. If you don't want this to be associated with all your other devices, just because this is going to link you into Google Play, and you perhaps don't want to have that exposed, you may want to create your own private Gmail account.

Once you have decided and fill the information out, you have to enter in your account information and your password, and it's going to continue to authorize your access.

Once complete, it asks for the time and date that you have to verify that. Next, just like a standard Google or Android device, it will ask you if you want to back up your data to the cloud. You can tap through this, hit “agree” to terms and services, and right down below the screen, tab down where it says, “No Thanks”.

Make sure you select the “No Thanks” and not “Add Another Account” option. Next, hit Enter, and you should now have a fully configured Android device virtualized that you can lab or test with.

This is not an emulator or simulator of any type. This is a real product running. Before you start playing around with it, go to Settings, and do any updates that you need to on the built-in applications on this device.

Then you can start looking at some of the mobile applications that we'll be

discussing shortly.

If you can't run this type of virtualized environment, no worries as I will be explaining everything in plain English, but hopefully, you have both; the software and hardware to create a virtual environment.

Chapter 5 Android Architecture

It’s time to take a look at different ways that we can hack the Android platform. That will include Android rooting as well as we will look at various attacks that can be placed on Android devices, and we'll take a look at malware specifically because it's rampant within the Android world.

Next, we'll look at countermeasures and understand what we need to do to lock it down to minimize the chances of security vulnerabilities. We'll begin discussing the architecture of Android by making sure that you understand how the device is operating based on its architecture.

To truly understand any security, you do need to understand the architecture that lays underneath the platform, because the architecture itself can show you where the security holes are going to be on that particular device.

When it comes to Android, it has its layers. The first layer of Android is what it’s called the Linux kernel. The Linux kernel was used by Android, because it's very stable, and it has a proven driver set.

The Linux kernel creates an interface between the hardware that may be installed on the device, whether it’s a display, or the camera on the device, USB ports, networking ports, Wi-Fi, as well as sound, and the power features too.

All Android devices are supported directly to the operating system via this particular layer of the architecture. Windows has its kernel-level. Mac OS has its kernel, and they all do the very same thing, which is to provide a way for the operating system to speak to the hardware.

Above the kernel layer, we have the libraries layer. This layer has some built in libraries, and one of them being the service manager. The service manager is there to compose the windows that you see displayed on the screen.

There's also the OpenGL aka Open Graphics Library, as well as SGL aka Scalable Graphics Library. These two are in charge of both of 2D and 3D displays.

What's great about this infrastructure is that an application can switch from 2D to 3D almost seamlessly.

Another library that's included with most Androids is the Media Framework. The media framework is there to support playback and recording of multimedia, whether it be audio, or video, or pictures back to the display.

For example, most of our codecs are built into this framework, such as mp3 or mp4. This is why when you start up a brand new device, you can take a picture and look at the image immediately.

It's in a JPEG format, or you can download an MP3 file, and immediately listen to it. You don't go and download and install any additional Codecs. We also have a free type.

This particular component is there to render our fonts. For example, if you have an Android device, you can change the fonts that are being displayed throughout the system, or its size, or its type, and some applications will also pull on this free type. Therefore you can change the font within the app itself.

OpenSSL is also implemented, giving us security when we're communicating with other devices, as well as SQLite. Unfortunately, there are few vulnerabilities that SQLite can create on Android devices, the same type of weaknesses that we see in a standard SQL database server can be implemented in some cases.

There is also a WebKit, which is an open browser. It's the same engine that Apple's Safari uses. There is also the Runtime layer, where we have both; the core libraries as well as the virtual machines.

As far as the core libraries are concerned, this is a runtime layer that includes a set of core Java libraries. This allows application programmers to build their apps, just using standard Java programming language.

The VM-s is open source software, and it's been streamlined for mobile devices so that there could be multiple virtual machines running to support numerous apps.

In the Android world, we sandbox each of our apps. It has a low memory requirement, and, it does rely on the underlying OS for process isolation, as well as memory management and thread support.

The layer above this is the framework layer. The framework layer is where we see a lot of our applications or the core applications that come with our device.

For example, a service provider may load some default applications that are proprietary to their environment. This could be the interface that they load on top of Android.

There are other types of applications, such as the Activity Manager. This is what manages the lifecycle of applications, which typically when it comes to an

application, the lifecycle includes the starting of the application, the running of the application, or if the application is paused, because maybe we switch out to different app, a stop state, because perhaps the application hasn't been used in a while, and a destroy state.

The destroys state removes things out of the resources when the activity manager decides that there's no more use for that activity. We also have the Window Manager, which is responsible for what we see on the screen itself.

There is also the Package Manager, which is in charge of keeping track of which applications we have installed, so when it comes to updates or how to uninstall a particular application, the Packet Manager deals with those. We also have the Telephony Manager that does precisely what we think telephony does on a mobile device.

It's in charge of managing your phone calls. Let's say that you've built an application, and once you click on a contact, and you see their phone number when you click on the phone number, you need to dial that phone number. This is where the application's calling on the Telephony Manager.

We also have the Resource Manager, which manages the resources that your application may use. There's also Content Providers, which manages the data shared between applications.

We get that seamless feel in the environment itself. As an example, a few of our default content providers include the call log, which provides our missed calls, call details and times, that we might see that information when we look at the contact.

For example, you can see the last phone call you made to your friend on what date and time. Another content providers would be the address book or the user dictionary.

As you start to type the letters when texting someone, the default user dictionary may not recognize the word, but you know that's a correct word that you sometimes hit.

Another concept here is when you install an app that has an activity, and when you click on the photo button, you might pull up the Content Provider into the Photoshop gallery, so that you're able to share photos via the application.

Another framework is the Location Manager. This is the framework that is used by your GPS, and possibly your cell phone towers, so it can determine where you're located. It does a triangulation through the cell towers.

Next, we have a View System, which is what it sounds like, provides different views that we have of the applications in separate windows or window sizes. It does a lot of communication through the Windows Manager.

Next, we have the Notification Manager. I'm sure you can figure out what that's supposed to do, right? Do we allow an application to notify us that we missed a call, missed a text message?

This layer has the actual Apps layer, and this is where we see things like the Home screen or our Contacts, the built-in applications, the Phone, the Browser, and as you add additional applications to your device, it adds it to this level.

You can see then, it would use the framework, as well as the libraries and the run time to interact with the devices to do what we want our machine to do. It’s important to know this, but we'll also discuss how the device works so that we can understand its security.

Chapter 6 Android Hardware Function Basics

As a security professional, you need to understand how the device is working, or at least the security side of the device. Hence we're going to take a look at first, the security model of Android.

First and foremost, Android it’s permission-based. Commonly, these permissions are implemented at two places in the Android architecture, both of the kernel-level, and also the application framework level.

As far as the number of permissions that are out there, (over 100) are defined, or that can be defined, you have to know that applications can define their permissions. This figure means the number of permissions available.

But, we usually break them down into four different categories. The first one is referred to as standard, which is low-risk permission that we give access to non-sensitive data or possibly even features.

Normal permissions don't require an explicit approval from the user at installation time. An example of this would be for contacts that have access or permissions to the phone application.

The next category defined as “dangerous”. These permissions grant access to data and features that may be classified as sensitive, which does require explicit approval when you do the installation.

An example of that is when you install an application, and you are going to get flagged on some permissions that are required by the application. Some of them are legit, some of them are not.

The next category defined as “signature”. Signature is interesting permission because it's defined by the application in what we refer to as the manifest file, but with signature permissions, the app that requires this permission can only be accessed by other applications that were signed by the same certificate.

Then we have another one that's called “SignatureOrSystem”. It's the same thing as signature permission, but the application installs inside of the system partition, which does have elevated privileges.

So, in that case, the application can either run as signature-based or at a system-level. Typically, when it comes to applications, at least on the Android platform, those applications must be signed to be installed, and this comes into where the app stores help to protect us.

By default, Android allows a self-signed certificate, so that developers can generate their certificate to sign their applications. Besides the application-level security, starting with Android 4.0, we could start getting into what they refer to is ASLR, aka Address Space Layout Randomization. ASLR, it's a technology that makes it more difficult for attackers to exploit any memory corruption issues.

An Android application is composed of about four different types of components, and each one of those components in the application represents a different entry point into the application.

The first one is called “Activities”. An activity defines a single screen of the application's users interface, and because of the way that Android is created, it promotes the reusability of these activities, so that the application doesn't have to redraw and recreate anything all over again.

For example, if you had a contact application and you wanted to pull in people's phone numbers and link it to the phone app, you are going to reuse those activities.

That's great, but on the other hand, this behavior increases an attack surface of the application that you're installing, because we're starting to share information between applications.

Another application module is the Content Providers. One of the things that it does for us is that it allows us to query, insert, update, or delete data to other applications and internal components. For example, if you are inside of your brand new application for contacts, if you delete a contact in it, it's going to delete it in the framework of contacts that came native with the phone.

The application might also store the data in the SQLite database, so you have to be suspicious about poorly written content providers that could open up vulnerabilities to an SQL injection, or other types of injection attacks.

We also have Services that are there to run in the background. While we're operating in the front end, we never see interaction with the services, unless it fails, but other components can start services.

If your application wants access to a service, or implement a service, you could open yourself up to another vulnerability. Finally, we have Broadcast Receivers. Broadcast receivers receive, hence the word receivers, from what we refer to as broadcast intents.

Intents are asynchronous messages that are used to perform in our processes as

well in our component communications. We have to be careful about applications, making sure they don't just accept data from broadcast intents.

This is because those hostile applications may be trying to inject or send data that are originating from a remote system, which could lead to some issues for us.

Now let's discuss the data storage. Data storage can be confusing because it depends on your device itself, but most of the time, data storage is stored on SD cards that are standard micro SD cards.

Samsung released a bunch of phones that had built-in internal SD cards. We refer to these things as NAND flash, aka Non-Volatile Storage Mechanism, which means that it doesn't need the power to retain information. We can use SD cards, and those could be either internal or external cards.

You need to make sure you are careful where you store information, and here's the reason why: files that are stored in external storage locations are publicly available to all applications and files, versus internal storage, by default, are private and accessible only to specific applications.

What does that tell you? Well, we need to make sure that we encrypt them. We already discussed NFC tags earlier, but these are just tags. Their design is excellent in the aspect that they're contactless. Still, you do have to be close to them, but it allows us to put macros or different tasks behind them. This is the same communication that's used for payment services, which we'll talk about later on. Android Beam uses this.

An example of that is known as “bump”, an application that is now being replaced by defaults as part of Android, but you can pull up contact information on your phone and bump it with another person, physically.

Bump the devices together, and it would transfer the contact information via this technology. You're going to start seeing these types of communications being used for advertisement.

There is also something called the Android Debug Bridge, aka ABD. This is a command-line interface that allows you to communicate with the mobile device via USB cable.

It's like a remote shell, and because of that, there are several different commands that we can be implemented remotely to affect the device. For example, we could issue push and pull commands.

The push will copy a file to your system, while a pull will copy a file away from

your system to your remote location. There's also “logcat”, which shows you logging information in the console itself, and this can be very useful if you want to see if an application or underlying operating system is logging sensitive information somewhere.

We can also issue an “install” command that copies the APK files to the mobile device and installs the application. When you're installing stuff from the App Store, that's what you're doing. You're doing a push, and you're doing an install, and this is technically how we “sideload” applications onto a device, which can be good or bad.

Chapter 7 Android Root Level Access

When it comes to rooting, all we do is escalating our privileges so that we have root access to the device. Naturally, applications are restricted by the security model, so that an application can only access its files, or the external SD card, while only has access to resources and features that are requested during the installation time.

By rooting a device, we're allowing specific applications to run as a root user. When this happens, the security model breaks down, because you're bypassing the default permissions. Yet, not all applications will automatically use root-level access.

For example, Facebook is still going to use its resources and the information that it needs, but I might find another application, and here's where it gets tricky. Many people each time when getting an Android device began immediately rooting it, so they can load specialized applications that allows them to do things such as backing up applications, their settings and their data.

When it comes to rooting, you have to know what you're doing. I warn you upfront that You are responsible for what you're about to do, and this is because you can break your device by rooting it.

Once you have root access to a device, you can also overclock the system to make it run faster. With root-level access, you can get rid of all the bloatware that came with the device.

You do have to be careful because if the applications are running with root user-level access, it can directly access the resources and bypassing permission checks. There is also a possibility that you're going to give the application full control over the device, and the other applications that are installed on it, such as your banking applications.

Having root-level access, you can do all kinds of installations by sideloading. Rooting your device will typically void the warranty of the device. What's interesting is that you can always un-root it. Either way, ISP-s don't like rooting because, if you had to do tech support as that carrier and people were running different versions of platforms on that device, it would be a nightmare to troubleshoot for the individuals.

Furthermore, there are also financial issues, as far as the carrier getting paid to put specific applications preloaded on your mobile device. Unlocking a cell phone that other carriers could use them was legal, but it's now illegal to unlock

your phone without your carrier's permission.

However, it depends on the date when you are reading this book; they can change the law at any time. But at the beginning of 2020 in the US, it is legal to root your phone if you're doing so only to use applications that require root-level access.

If you're rooting your phone for any other reason at all, technically, you're breaking the law. Regards to Apple products, if you've got an iPhone and you jailbreak it, you could violate the law. Unfortunately, most lawmakers don't understand how technology is used, so currently, the law has exemptions in there for jailbreaking and rooting tablets.

The exemption that was made only applies to wireless telephone handsets. Now in Canada, they passed the Copyright Modernization Act of 2012. It makes tampering with digital locks illegal, but it does have a specific exemption for interoperability, security, privacy, encryption research, and security.

Therefore, if you're rooting an Android device so that you can run applications that aren't allowed on the App Store, you should be, in the EU, they have Computer Programs Directive, which says that rooting or jailbreaking is a violation of copyright, just like the US and Canadian laws do.

But it also says that if it's for interoperability, or fair use, to root or jailbreak with intent of running, other or legally acquired software, then you're ok. I already mentioned the US, Canada, and the EU, yet you might be located in India, so instead of assuming that rooting or jailbreaking is legal in your country, please double-check so you don’t get into any trouble.

Now let's move on and take a look at some new terminology that we're going to be discussing as we continue in the rooting process. First, we have the bootloader. The bootloader is the very first thing that fires off when you turn on your device.

It's like the BIOS in a PC. It has code in it that it’s an operating system and kernel should boot off from. It understands which OS to boot from. Android devices have multiple partitions, and we can put this in computer terms.

Imagine that you have a computer, and you've got multiple hard drives in it. How does your computer know which hard drive to boot from? Well, in the BIOS, you go and say that: "Boot from a particular drive, " so it looks the partition you told it to.

That's going to be determined by the boot sequence of the operating system, and

it's the same concept when it comes to Android too. Each device is going to have its own bootloader, and most of them are going to be locked, while some of them may be even encrypted.

Manufacturers usually do this so that they can ensure that you stay within their parameters as far as which OS versions should be used that they've designed for the device.

Without an unlocked bootloader, it's almost impossible to flash a custom ROM. Many ISP-s do encrypt their bootloader on several of their devices, which makes it very hard for anybody to take advantage of the running custom ROMs.

When you unlock your bootloader, besides voiding the warranty, it also will wipe the internal memory, so you lose your contacts, your messages, and you start over with a new phone. We also have the kernel. We already discussed it earlier, and this is the level where the apps have access to the hardware via this communication channel.

The kernels are customized by the manufacturer as well, and this is one of the reasons why many people tend rooting Android devices because they underclock the performance of the CPU.

Devices would be getting more battery life if they would stop loading the bloatware to run automatically. People need to make the decision on whether or not they want Twitter installed, and whether or not they wish to be notified every time there's a post.

Most custom ROM-s can be tweaked in such a way, so they can no malware on, no bloatware on, and can run the device faster while can get double the battery life out of the same system. Another feature of the kernel is the sound governor, which would allow you to get the volume even louder if you wanted to.

The next item on our list is called “baseband”. Baseband is a type of software that allows the correct communication with the radio and the operating system. Most manufacturers don't update the baseband and if you find a newer tweaked baseband, you can improve the battery life and the call quality.

Back in my Samsung Note days, there was a radio that improved reception. The next item on our list is called “recovery”. This is a mode that we go into to recover or help us to troubleshoot and upgrade our environment.

For example, if you are about to receive an update from your ISP when you select that you'd like to go ahead install it, it reboots the phone and you get that

little screen that has the Android on it.

That's the recovery mode, but we also have the stock recovery, which is a very plain looking screen. You can install a custom recovery, which allows you to do all kinds of things, such as create a complete backup of your system, or flash a new ROM, or go back to your original ROM.

We also have the Dalvik-Cache and ART. ART is a new runtime for Android, but they released it as an experiment, and it's becoming fully deployed in various devices.

Dalvik is based on a JIT, aka Just In Time compilation. What JIT does for us is that each time when you run the app, a part of the code required for execution is going to be compiled to the machine's code, just at that moment and as you continue through the app, additional code is going to be collected, and it gets cached.

Because JIT or Dalvik compiles only part of the code, it has a smaller memory footprint and uses up less space. ART however, takes the Dalvik code and compiles it into a system-dependent binary.

The whole code of the app will be precompiled during installation, but it's going to take up more space, yet it should execute much faster. They could have implemented ART immediately but the first Android devices had deficient memory and CPU capabilities.

Nowadays, we've got phones that are quad cores and 16 gigs of RAM. As we make the transition over to ART, application developers will have to make sure they write for it, because not all apps are compatible, which means they'll still run, but it may take a little while to convert it.

But once it's done, it increases the CPU by about 20%, RAM operations get increased by about 10%, and storage operations are increased by about 10%. In other words, we can make them stronger and faster.

Besides the recovery mode, the Dalvik-Cache, ART, and baseband, we also have the download mode. Download mode, in some devices known as fast boot mode that gives you the ability to flash images. Those images could be customizations as well as ROMs. The manufacturer uses download mode to install the ROM on the device before they box it up.

Chapter 8 Rooting Android

Each Android device has its steps that you need to go through to root. The particular machine I will discuss with you is a Samsung Galaxy Note 8, and I'm going to share with you what we've discussed already. The bootloader as well as the download mode, and then I'll also look at Odin, which is what we use to root the device.

There are also some other utilities out there. I'm just going to share with you the tools that I use. There are many different products out there too that you might find better for the particular device you want to root, but the concepts are the same.

It's just a matter of preference, but you first have to begin by turning the device off, because to get into it, you need to power it off and get into bootloader mode. This is device-dependent.

If you are testing this using the same model, you are going to have to hold down the up volume button and the home button and power it on. Once you hold those down, hit the power button. It will vibrate once, and you should see at the top that there is a blue text saying, "Entering recovery."

Once the little Android guy pops up, you have to edit the standard bootloader from Samsung and you will see that you have several different options here.

The Apply Update options are different ways of updating the ROM and typically, when you get an update from the manufacturer, they get scripts that not only downloads the update, but takes you through this process and automatically selects to do an update from cache.

You should also see that you can do a wipe of the data in factory settings as well as the cache partition, and you can reboot into the bootloader, which is the same environment, or lastly you can also power it down.

You can chose the stock bootloader to see what it looks like after you root this device. But before you root a device, you'll need to make sure that you get the appropriate root files for the device itself, and you can to do that by going into my settings on the device, scroll down where you have the “About Device” option and you should see that you have your current baseband, kernel version, the version of Android that currently running, and the model.

This is important because you want to find the root of this particular model. In this case, it's an N950F, and don’t forget that you can mix and match. You can

also scroll down and see your current build number, as well as with the Samsung devices; we have something that's called Knox.

Knox is a security feature so that you can support both enterprise and personal environments on the same device without it being compromised. One of the things you're going to want to do to root a device is, typically, in an Android device, you want to look at the build number, tap it several times and you should see that it gets into developer mode.

What that means is that it will give you access to the resource through a USB cable, so if you hit “back”, you now will have the developer options available to you.

You can also turn on USB debugging while you can use Odin to connect to the device to root it. Next, you can hit back and look at download mode. To get into download mode, it will allow you to download ROMs, especially customized ROMs, onto a device.

To get there, you can do the same thing as before. Power it off, and if you remember, to get into the bootloader, we used a Volume Up, Home button, Power button key combination.

All devices have their way of getting in. Therefore you will need to do some research to figure out what you need for your particular android device. But in this precise device, I use a Volume Down option, Home button and turn on the power and I'm going to get a warning here that I'm about to add a custom ROM.

It warns you that you're going to take responsibility for what you're about to do and it tells me to continue, I have to hit the up arrow button. Once you hit the up arrow button, it will now be going into Download mode.

In the upper left-hand corner, it will show you information such as the product ID; the current binary is Samsung official, also shows you that the Knox warranty is currently set for zero.

Knox in the Samsung environment, it’s a chip that's on this device, and if I do a custom ROM, it modifies that chip. The microchip can't be unmodified; therefore, after you flash and do my own custom ROM on the device, your warranty's going to be void.

This is the one way that the vendor can determine if they know what they look for if you've done a custom ROM or not. While you are this mode, you want to hook your phone to your computer using a USB cable.

Your computer should already have the drivers loaded up, as it’s also an

important part of the preparation for your device to receive a root and the new ROM.

I do want to share with you a few resources that you can use as far as rooting is concerned. One of the most famous sites for rooting devices is called XDA Developers. You can visit the website on https://www.xda-developers.com/

This is a website that has everything about almost every device out there. You can visit the site and select forums and then select from all sorts of devices that they support.

You can also look at the top devices, the newest devices, but if you select All Devices, you'll see there is an overabundance of devices that are going to be listed in there.

You can do a quick search on the website for a particular device you have, and once you found it, you will see that the site describes the device itself, gives you some specs, and then they break it down into things like a Q&A where you can figure out how to root it for example.

Further down at each device's page, you can also find the custom ROMs. For example, Sun Engine is a very popular ROM that's out there. It's gaining a lot of attraction and some of them will be based on its distribution.

The way you can choose a ROM for your device is to see which one has the most activity. You can also look at the number of views on the forum where they might explain that they've combined some of the best of the latest devices.

The important that you read everything that developers are posting. For example, it'll tell you that if you have a particular model, there are some additional things that you need to pay attention to.

For example, in some cases, for a flawless experience, you should be on a particular baseband and therefore, you would want to find that baseband or make sure you updated to that.

In other cases, they may warn you that the initial boot might take at least 15 minutes, so don't get impatient. Therefore, make sure you read, and you don't just start going at rooting devices and see what happens.

Some apps might be advertising that you can download them on your device and they are going to root the device for you. Be careful with those. Not saying that they're terrible, but if you are unsure of the source, as well as loading an app to root your device, it does sound reasonable that they want to automate the process but still have to be careful.

I have heard before that if you use these types of services can be dangerous simply because the superuser account has total access to your system.

Moving on, now that you know about different modes and settings, let’s discuss Odin. There are several different versions of Odin out there, and you can download it by visiting https://odindownload.com/

The older versions have been around for a long time, so make sure that you have the latest or even check with the author of your ROM or the author of who's telling you how to flash a particular device because they need to tell you to use a specific version of Odin.

In my case here, I'm using Odin3, version 3.10. This is the utility that will brick a device faster than anything else, so you want to be very careful using this file, and you want to read about how to use it. IT guys tend just doing and then reading later, but that's going to get you into problems with this one.

To use Odin, you have to put the phone into download mode and plug in the phone, and then you'll notice that you have a little blue icon saying com 3. That tells you that Odin is talking to your phone and it is currently in download mode.

The first thing you want to do with Odin is to member your recovery mode. Install your custom recovery, so change your bootloader. The plain text bootloader gives you the ability to wipe the cache or do a factory reset.

This particular bootloader for my device is called TWRP, so I have to go to the file called Open Recovery TRWP. Go ahead and hit OK, or “Open” and it's going to verify it then hit Start.

This will upload the information, and if you look at it, you will see that this is your new recovery mode. Now you can Install, Backup, Mount, Advanced options, Wipe, Restore, Settings and Reboot.

Here, select Install a new kernel. The new kernel is going to allow you to install custom ROMs as well. You should see here that you have almost like a directory structure, so go ahead and navigate if you have saved your file to your external SD card that’s called Emotion kernel.

Go ahead and swipe to install that, and then you also want to install your SuperSU to give you superuser accounts that have rights to everything, then reboot the device.

Once your phone is back, you'll notice that nothing's changed except for the fact that under my apps, you have something called SuperSU, so the application is running, and now you know that you have superuser rights.

Knox is going to fail because Knox was the security mechanism that you just broke and that's the only time you're going to see that little error message pop up. If you go back into your download mode, you will see that it voided the warranty.

Next, restart the device, and get back in download mode. Here, you'll notice under Knox warranty; it now tells you that you are void. From there, you can download your own custom ROM and install it. The reason why you had to do the custom recovery and the custom kernels is that the stock recovery doesn't allow you to install any ROM, unless it's been digitally signed by the manufacturer, which would be, in this case here, Samsung.

But now, you can load up a ROM that doesn't have a whole bunch of bloatware in it, that's a bit more secure. You don't have to wait for your ISP or Samsung to come out with a fix for a particular issue. The other downside of rooting and doing your custom ROMs is that once you've rooted, you will no longer get over the air updates from your vendor.

You might get the updates, but it won't let you apply them, because it will see that the security has been broken. You can always unroot your device and there are steps on how to do it, but now that the device is rooted, let’s discuss different types of attacks that can take place against Android-based devices.

Chapter 9 The danger of Free Apps

First of all, most people keep on installing applications from sources or developers they don't know. Granted, some apps come from major suppliers and vendors such as Microsoft or Google, but what about John Smith and the game called Tetris right?

Maybe he's trying to copy of a popular game. But, John Smith can decompile an existing game or create one from scratch and just simulating an existing one. Let's say that you want to take a popular Tetris game, and you are going to pay for it.

Then, you are going to decompile it. Decompiling can also help an attacker to see what vulnerabilities are accessible through the application. Maybe you don't have to do anything special to it, because it's just writer poorly. Next, you can download a program such as dex2jar from https://github.com/pxb1988/dex2jar , that Dalvik VM uses, and it converts it into Java or a JR file.

You then download a Java Decompiler from https://github.com/java-decompiler/jd-gui/releases , and you can start to inspect different parts of the application. Next, you will take your mischievous code, such that you can track passwords that users may type in via web browsers. As a result, you can disassemble the APK on the fly.

Then, you simply repackage the now infected program, and distribute it through unofficial marketplaces, or maybe through a torrent. Relying on the social Engineering aspect of greed, many people will download and install it.

One of the ways that you can implement a countermeasure against these types of attacks, especially if you're developing your own apps, is to use something like a Pro Guard from https://github.com/Guardsquare/proguard , which is a program that obfuscates the Java classes by remaining the classes and field to methods.

This isn't a best way of completely defending against everything, but it slows the attacker down. There are another types of network attacks that you can implement in different ways too. But as a countermeasure, the first thing you can do is start looking at network traffic to find out which applications are pulling http or https traffic.

Most applications will advertise themselves that they are protecting you because they're using TLS to mitigate the risk of you being involved in a man-in-the-middle attack.

But, we can quickly get around this because all we have to do is add our own trusted CA or a private certificate to the Android Device. Then implement that on a proxy server. And you just got hocked, because at this point, I'll be able to intercept all your https traffic the application is using with a product such as Burp Suite from here https://portswigger.net/burp/communitydownload

At this point, I can then start manipulating the HTTP requests and the response between the application and the endpoints of its contact. Once you set up this type of environment, you're able to bypass the client-side validation.

One of the most significant issues that developers have is the tendency to disable certificate verification and validation so that they can use self-signed certificates for testing. Then they forget to remove this feature, which leaves their application open to man in the middle attacks. Therefore, do not trust data from anybody or anyone by going to websites.

Chapter 10 Android Attack Types

We already discussed NFCs in the aspect of how great they are. As an attacker, you might be looking out for legitimate NFC tags. They could be in the mall or at the grocery store, and you can create your tag which allows the user to tag or tap that tag to get information.

Well, all you have to do is create your tag. Place it on top of that tag, or tear off their old one and put yours there in return. Then you send them off to a website that has an exploit to a vulnerability in the WebKit.

So, any end-user who scans the tag would find their device rooted and compromised. Rooted, I mean not in the right way. Another attack, that's an interesting one that NFC opens up, is something called a relay attack.

To execute a relay attack, all the attacker needs is two devices which act as a token and a reader. They create a connection via a proxy channel, typically Bluetooth, but could be Wi-Fi as well.

This allows relaying the information over a greater distance. The proxy reader is then used to communicate with the real token, while the proxy token is placed near a real reader.

The token assumes it's talking to the reader and responds accordingly. That response is then relayed back to the token proxy, which will transmit the information back to the reader. You'll never guess what information we end up with on our systems.

So, when it comes to data leakage, normally the Android operating systems stops applications from getting access to other application files or their internal files, because each application is assigned a unique user identifier, and a group identifier.

But, an application could create a “word readable” or a “word writeable” file. What this could do for example, is that imagine that you have an application like Twitter that stores credentials to talk back to it’s server.

In a meanwhile, a malicious application on the same device could read that file and send your login information back to my evil server.

Another data leakage area would be external storage. Remember that any file that's stored on your external SD is globally readable and writeable to every application on the device.

Therefore you should stop storing sensitive information like your passwords, server names on IP addresses on external SD cards. Another data leakage location could be in the fact that SQLite is supported, or it's implemented throughout Android.

Anytime we see SQL, it means that we have to be worried about SQL injection attacks. As far as the developers are concerned, they have to avoid storing information in unencrypted formats of any type.

Also, you should never create globally readable or writeable files, and you must also encrypt your phone. Lastly, we can look at logs. Information always leaks out via logs.

Developers document way too much during the debugging process, so they can test to see what's going on, while they forget to turn those off. Some Android applications will search the device or gain access to these application log files during installation time.

A great example of this was when Samsung stored touch coordinates inside of a log file, which would allow an application to determine the user's pin, based on the log touch coordinates.

Moving on, we also have a URI scheme. This is when there is a malicious web page loaded into the Android browser that can get hold of the contents of a file on the SD card.

One way to protect yourself is to disable JavaScript on your Android browsers. Also, you could look at using a different browser that might prompt you before downloading a file, or change the settings of your browser, so that it prompts you.

Many times if you use the default browser on the device, you're relying on the ISP to release a patch. Technically, you could unmount the SD card. There are also insecure components, such as Wi-Fi. What if there's a vulnerability that allowed any application permission to acquire your Wi-Fi credentials right?

This vulnerability had modified a function in the Wi-Fi configuration class that included the password to be stored. That allowed you to recover a list of all the objects that the Wi-Fi configuration object had taken a look at and seen.

That would have the passwords in clear text instead of the standard storage of asterisks, or an empty string. Maybe you can see why custom ROMs are great because you don't have to worry about certain vendors having code that makes you worry when you think about what they have access to.

The problem is that the attackers are getting very tricky in terms of how they to deploy malware, that it's hard sometimes to determine what is real malware and what is an application that's just been modified by injecting their piece of malware inside the application.

As far as the application developers are concerned, is that sometimes we start with an application that doesn't seem too bad, but then it’s all go wrong.

When you think about all the different applications we have installed on our mobile devices, it's probably more than what average people have on their PC-s.

Applications on the mobile platforms do help us, as far as making sure we're not susceptible to some vulnerabilities that a website might give us. We might go to a website to book flights, instead of going to the site and hoping they're not compromised, we can go to Flight Company’s app, and because their app is sandboxed, hopefully, we are secured.

Applications that we get through the app stores are typically safe. But, there was an app called DroidDream, that was distributed through third-party application marketplaces.

DroidDream allowed the user to go through the install process, and you got the application. But, you also got apps that were repackaged with the DroidDream malware on it. So whenever you are installing an application, watch the permissions.

For example, if you are going to visit the Play Store and check if you have any updates, you might see several updates. Some of them are Google updates, but many times people will visit, and what will happen is that when we see updates, people select “Update All”, and they don't pay attention to what's happening.

Well, the application itself, or the vendor could start all innocent, but over time through updates, they might require more permissions, just because they want to be a little deceitful.

Therefore you should always look at permission details. That will tell you that it needs Wi-Fi connectivity, as well as full network access. Well, you have to think about what they want or why would they want full network connectivity.

If it makes sense, you might go ahead and update because some apps don't require any special or additional permissions. This helps to warn you when an application is changing the permissions that you gave it initially.

This part of the social engineering aspect of hacking is that we tend just hitting

accept. Other apps might want to have access to some Identities.

For example, they want to find accounts on your device. Perhaps they want to do that because your account is related to your email account, so it's looking for those accounts.

Regards to location checks, it could be because you are subscribed to a service, and the App providers want to tell you where the closest facility is available to you.

Yet, if they want to have access to your Bluetooth connection, that should concern you unless it’s a big company that you already trust. Big businesses will not be going to do anything to put their whole company in jeopardy. At least that’s what we all hope unless they are compromised right?

Many apps have ads, especially the ones that are free versions. They're going to have some ads popping up, and they want to know where you are located so that they can focus those ads based on your location. The other issue that if they would need more than just your Wi-Fi connection. Wi-Fi only makes sense because maybe they're going to download those ads to you.

But maybe they also want to see when you are online and when you are offline. As a result, I recommend you not only watch your permissions on your applications, but do a little research, and find out what other people are saying about the application. Not necessarily on the review page for that app, because they have people that post positive comments and they're paid to post those comments.

Chapter 11 Securing Android Devices

When it comes to locking down Android devices, there are a couple of things we need to take a look at first. Some people still can’t decide (myself included) which is better, an Android or Apple device. Well, it depends on your background.

If the user has more experience or a technical background in IT, probably going towards Android rather than Apple is a better choice. But, if it's just an end-user or a family member, who wants a device to work, then going towards the Apple platform is more reasonable. At the same time, there are many technology professionals just vale Apple.

The biggest reason that you want to look at these two different platforms, or consider these two distinct platforms, is because of the attacks that are thrown at them. Let me ask you a question. Who has a more significant market share?

Because whoever has a more significant market share is going to have more attacks associated with it. In this case, Android does have a more substantial market share. The fact that Android is so fragmented, with so many different vendors, they have a more significant market share compared to Apple.

The other issue is the approval process for applications. this is going to be where Android suffers, because what happens is that developers who create an application typically pay a one-time fee, and they can upload their application to the Google Play Store. After that, within an hour, the applications start to appear.

Versus Apple, that goes through and does statistical analysis to detect any improper usages of their API. This then causes the application to take about a week to be approved.

Also developers are required to pay like about $100 for a developer fee, which does raise the bar as far as entry into the App Store. In a nutshell, Apple can decide whether or not your app makes it into the store. To get into third party application stores, Android is your choice because it does support installing apps from unknown sources.

This means that the user can install from these third-party app stores. Just because the app says that it's signed, doesn't necessarily mean that it's approved, because Android doesn't care who signs the application.

Versus Apple, which only allows users to install iOS applications from its App Store. There's no side-loading, natively. So, the iOS kernel enforces the security

mechanism by only executing signed code by an approved party.

While Apple's approach is a strict control, it does give you the added benefit of a reduced target, as far as malware is concerned, not that it's invulnerable. Versus Google's approach which provides you with a lot more for and a little bit more freedom too. It is up to the end-user really.

Personally had both Android and Apple products, and they all useful as well bad features, or I should say not as user-friendly on one platform, while it is on the other.

Back to the topic of protecting ourselves, the first rule is this: No side-loading. Some could even argue of possible security issues that it's more than likely that performance or hardware related issues are also important, but you will never go and download an APK file, or a collection of APK files from a torrent and side-load them.

Also, make sure that you keep on eye your updates. When it comes to updates, not only should you make sure that you are updating the applications themselves, but the OS as well.

If you're rooted the phone, you 're going to have to come up with an alternative way of updating. Also, make sure that you use some security mechanism for locking out the screen.

Don't do the standard swipe to unlock, because if you lose the phone, anybody can undo that. Certain of the newer technologies that we're seeing include fingerprint scanners or facial recognition.

At this point, most of these are not 100% foolproof, but at least it slows attackers down. Also, make sure that you use a legit App Store. Stay with the leading vendors such as Amazon or Google. Those two are being the primary sources for Android, at least in the early 2020s.

Also, get yourself an antivirus. A quick Google search for the best antivirus for Android including the current year will give you some results. Remember, even that it's a mobile device, it's still a computer system, so we need to protect it.

Also, don’t forget to protect the apps. For example, if you have some apps on your mobile device that used for personal information, so when you tap on the app, it should ask you to type in a separate password.

This way, you can make sure if you lose it somewhere, people don't start transferring money out of your account. Anything you wouldn't want a stranger, who picked up your phone to be able to see, you need to make sure that you

protect it.

For those who don't know much about technology, I would recommend No Rooting.

In summary, we have looked at the architecture of the Android platform. We talked about its security model, and if you recall, that architecture starts with the Linux kernel, which is there for support of your display drivers or communication between the device and the operating system.

Then we also discussed different libraries that are in charge of handling 2D and 3D graphics. We also covered the Service Manager and SQLite. Next, we have discussed the Free Type, which handled our fonts, as well as the core libraries and the Dalvik Virtual Machine.

We also discussed what an Application Framework is, where the applications help support the default apps, and the applications themselves. We then looked at understanding how the device worked and talked about the security model that they have as far as permissions required for applications.

Likewise, we have discussed the application modules and data, and how data is stored on SD cards as well as what problem we have with external SD cards.

Next, we have covered NFC, aka near field communications and the problem that it can create for us. We then took a look at the concept of rooting and the bootloader.

Next, we have installed a custom bootloader via the root process, and install the SuperSU user on the device, so that we can do whatever we want. Lastly, we went through different attacks, such as internet-based attacks, network traffic attacks, information leakage, and the NFC relay attack.

Chapter 12 IOS Architecture Basics

Once you get into an Apple environment, it's very restrictive. Yet, Apple controlling both hardware and software, it does lead to better controls all around.

First, we are going to make sure that you understand the IOS architecture, and how Apple is designed. We'll also look at understand the device itself, and then, instead of talking about rooting, in the Apple world, we do something called jailbreaking.

Lastly, just like in the Android environment, we will discuss how we can secure the IOS or Apple devices. To begin with, Apple is exceptionally unique. It has some similarities as far as it’s layered to Android.

There are several different layers that we'll take a look at, but this particular operating system is called IOS, aka Integrated Operating System. It is the platform that Apple uses for iPhones, iPads, iPods.

Apple does not allow iOS to be implemented on any non-Apple devices that are out there, and just like what we have with Android, IOS is simply an intermediary between the hardware and the applications.

Applications are not supposed to try to access hardware directly. Instead, they're going through several different layers of the architecture. The first one is the core OS.

The system-level involves the kernel environment, the drivers, and the low-level UNIX interfaces of the operating system. IOS provides a set of interfaces for accessing many of the features of the operating systems.

When devs create applications, those features are handled through the lib system library, and the interfaces are C-based, which gives us the ability to control things such as networking or BSD sockets, locale information, also networking components, as well as the file system access.

It's also in charge of things like DNS services and Bonjour. It also has a core Bluetooth framework, which is responsible for what Apple devices sound like. It's a framework that allows developers to interact with the Bluetooth accessories, or to create accessories via Bluetooth, so we're able to scan for those accessories, then connect or disconnect from them.

We're able to broadcast using iBeacon. We can preserve the Bluetooth connections and restore the links when your app needs that Bluetooth device.

There's also a generic security service framework or also known as GSS, which gives you a standard set of security services for IOS applications. Besides the regular interfaces that it supports, IOS can also include some additions for managing credentials that are not part of the standard but may be used by specific applications.

We also have the framework for the external accessories, which gives us the ability to communicate with hardware accessories over the 30 pin connectors and lightning connectors, or even using Bluetooth.

Then we also have the standard security framework which Apple uses to guarantee the security of data that applications utilize. This interface manages the access to certificates, public and private keys, trust policies, as well as the storage of certificates and cryptographic keys in a key chain.

Without getting in-depth with key chains, the key chain can be used among multiple apps that you create. This way it’s easier for applications to interoperate.

Above this layer, we have the core services layer. This layer has different features associated with it, such as the peer-to-peer services which your device uses to connect or initiate a connection to other devices that are nearby. This is mostly used in games.

There's also the iCloud storage. This feature allows you to write applications so that the user's documents and data are pushed to the iCloud. There's also the block objects feature, which will enable developers to incorporate their Object C and C codes, which is a developer's concept as far as a language construct is concerned.

It allows them to implement callbacks, delegation methods, and other mechanisms that they can use to combine the code that needs to be executed with the associated data that it needs to be performed with.

There are a bunch of services that developers use, but we also have data protection features that allow the apps to work with data that needs to be encrypted.

This way, if the device is locked, the contents are inaccessible to the application or possibly any other app that may be considered an intruder. When the device gets unlocked by the user, a decryption key is created, which allows the app to access the file.

The core services also include the SQLite. This allows a developer to embed an

SQL database within the app without having to run separate remote database server processes. Therefore your app could create databases.

It could manage tables and records. There's also the massive core service framework. The framework itself has several sub-components, things like the account framework for signing in.

We also have the address book framework, the ad support framework, the core location framework, the media framework, the motion framework, and the telephony framework.

Above this particular layer, we have the media layer. This layer is in charge of several technologies, and those technologies would include anything that deals with media such as the camera, audio, as well as both 2D and 3D.

It uses OpenGL and GLKit that are in charge of both 2D and 3D rendering. There's also the core graphics framework, which is also known as Quartz. It's not as well-performing as OpenGL, but if a developer needs to do a custom rendering in 2D, they can use Quartz.

Then there's also the core animation feature, which does exactly what it sounds like if you want to change the animation process of even if it's just displayed for an app, how the app zooms in and out as you touch things.

Above this level, we have the Cocoa Touch. This is a critical layer because it's the first layer that most developers deal with. It's responsible for the interface that we see that's standardized throughout the Apple platform. Our buttons. It's also in charge of the Text Kit, which is what is in charge of handling text and the way the text is formatted; paragraphs, columns, and the pages.

Cocoa Touch is also in charge of our push notification services, so we can push text notifications or add a badge to your icon that says that you've got a notification, such as a new text message.

There's also the auto-layout feature, which allows developers to create interfaces with a tiny amount of code. You could say that an interface would always be 10 points away from the right edge of a parent's view, so it's in charge of our layouts.

There's also the gesture recognizer, which picks up the pinching technology that Apple is famous for. Then there's the standard system view controller. This way, Apple can make sure that applications presented in a very consistent experience, whether they're in your application or a different application. It's just one environment.

There are also other types of Cocoa Touch frameworks, including, there's an Address Book UI, there's an Event Kit UI, a Game Kit, a Map Kit, even a Twitter framework, and the UI Kit framework, which handles copying and pasting, multitasking, PDF creation and so on.

Chapter 13 IOS Hardware Security

Let's discuss an Apple device in detail, and in this case here, we'll try to understand where it came from and how it operates. Back in the 1980s, Steve Jobs had gotten forced out of Apple and he went and founded a company called NeXT.

NeXT was focused on creating powerful workstations for both business and educational purposes. The company wasn't as successful as Steve had thought it would have been, but it did sell over 50, 000 computers.

When it comes to the framework of NeXT, it was used to lay the foundation of the first web server and web browser software. The NeXT environment was built off of Objective-C.

In 1996, Apple purchased NeXT. This was when Steve Jobs made his comeback into Apple, and with that, they introduced the technology called OpenStep. This was the new operating system for NeXT, and technically they renamed it to NEXTSTEP.

NEXTSTEP was then utilized as a replacement to the Mac OS or the classic macOS environment. In 2007, Apple announced that they are getting into the new mobile platform via the iPhone, and this did change the electronic industry.

Apple has been very inventive when it comes to pushing things to the next level. For example, the iPod was instrumental in getting us into digital recording, where the iPhone was the next step to creating mobile devices that are so powerful today.

When they first released it, they called it the iPhone OS, and only later renamed to just IOS. Because of the success of the iPhone, as Apple released new products such as iPad or iPod Touch, they used the same operating system.

When it comes to Apple TV, it's different than some of the other products by Apple, because it's an embedded device. It's not designed for mobility, but in its heart, it still runs a version of iOS. When it comes to security, Apple's been able to control the safety of their devices, because they manage not only the operating system but also the hardware itself.

At the heart of the hardware for the iOS devices, we are utilizing ARM-based processors, and that would also include the current A series of CPUs by Apple. Apple devices have many similarities, but each device is relatively different.

They do share some of the same frameworks and layers that support the iOS

platform. Apple also releases major updates yearly. They push out security updates all the time, but significant updates are typically done every year.

As far as its footprint is concerned on the devices, it usually takes up about 1.2 GB of space, which is low for an OS, but there are so many different versions of these products that some are different, and it also depends on when you are reading this book.

When it comes to the bootup, this is where the security of Apple shines. During the entire startup process, every component that's involved in the bootup process or the startup process has been cryptographically signed by Apple to make sure that the device is what it is, and it hasn't been modified.

Those things include the bootloader, the kernel extensions, the kernel itself, and baseband firmware. When an iOS device is turned on, the application processor immediately executes a code from read-only memory aka the boot ROM.

This boot ROM is laid down on the chip during fabrication. Therefore it is implicitly trusted by the device. The boot ROM code comprises the Apple Root CA Public Key, and it is used to verify that Apple signs the low-level bootloader before it continues to load.

This is one of the first steps that's used in the chain of trust as the machine boots up and lays our foundation for us. After the low-level bootloader fires off, it verifies and runs the next stage of the bootloader which is iBoot, which in turn then verifies and runs the iOS kernel.

If the device is a cell phone or has cell phone service, the baseband subsystem also utilizes its boot process with signed software and keys that have been verified by the baseband processor that has been loaded by the cell phone provider. This is how they used to test and made sure that you had an iPhone on the ISPs network when you had to be on a specific Carrier.

It's still utilized today because the cell phone vendor wants to make sure that you're running the device, and it hasn't been compromised on their network. If you have a newer version of an iPhone, it must have a SE aka Secure Enclaver Co-Processor that ensures its software has been verified and signed by Apple, and it hasn't been modified.

If one of these steps in the boot process fails in any way whatsoever, then you're going to get a warning saying that you need to connect to iTunes. This is called Recovery Mode. If the boot ROM is unable to load or verify the low-level bootloader, it enters into DFU, which is short for Device Firmware Upgrade

mode.

In either case, the device has to be connected to iTunes to restore the factory default settings. Sometimes Apple releases software updates to the platform to address more current security concerns, and when they do these updates, if you've jailbroken the phone, it typically tends resetting the phone.

Chapter 14 IOS App Security

When it comes to the application security, once the iOS kernel up and going, it controls which user processes and applications can be run. To make sure that all apps come from a known or approved play store, such as the Apple iStore, which is the only one at this point, and haven't been compromised in any way, the iOS requires that all executable code must be signed by using an Apple-issued certificate.

The apps that come with your iPhone, such as Mail, Contacts, or Safari, are signed by Apple. Third-party apps have to be validated, and that's a whole validation process they have to go through with Apple where they submit their apps to Apple, and then Apple issues a certificate.

If you work for a company that wants the ability to write your applications, you can apply for the Apple Developer Enterprise Program, and part of that process is that you have a DUNS number.

Apple then does background checks and approves you, in which case the company can register and obtain their provisioning profile, which allows in-house apps to run on their devices that it authorizes.

Once the app has been verified that it came from an approved source, it looks at its runtime to make sure the app hasn't been compromised and won't try to compromise any of the other portions of the system.

All third-party apps are sandboxed, so they can't have access to files that are stored from other apps, and one of the ways that they do this is that each app has its home directory which gets randomly assigned when the application gets installed.

If by chance the app needs information from other apps, it does so only by using the services explicitly provided by iOS. Another app security mechanism they have in place is utilizing extensions.

This means that iOS allows apps to give functionality to other apps by providing extensions, and these extensions are just special executable binaries that are packaged within an app itself.

The system automatically detects the extension during installation and then makes them available to the other apps using a linking system. The system area that supports these extensions are called extension points, and each of these points has APIs that enforce policies for that particular area.

In this case, the system will automatically launch the extension process as needed to manage the apps. As an example, you have the News view widget, which is visible in the Notifications Center, and those extension points are the News widget, the share, the custom action, the photo editing, the document provider, and custom keyboard.

Each of the extensions run in their own memory space, and communications between the extensions and the apps are maintained by the system framework that we discussed earlier.

We also have app groups. Apps and extensions that are owned by a particular developer account can share the content of their application when it's configured to be part of an app group.

For example, Twitter may have the Twitter app, as well as the Messenger app. It's up to the developer to create the app group, and they do that in the Apple developer portal, where they say these are the apps and extensions that they want within the app group.

Once placed inside of an app group, the apps have access to a shared location in storage, and that information stays there as long as at least one of the apps in the group has been installed.

Therefore, if you install Facebook and you install the Messenger service but then you uninstall the Messenger service, the information is still there as far as storage is concerned until you remove all the apps in the app group.

They also will be able to share preferences, sometimes settings, as well as key chain items, which deals with security, so you don't have to keep logging in and out.

When it comes to accessories, there are all kinds of them for the IOS platform. For an accessory to get made for the iPhone, iPad or iTouch logo, they have to go through the MFi approval licensing process, and this helps Apple to ensure the accessories aren't using any backdoor to get into the device.

As an example, when an accessory communicates via Bluetooth, the device asks the accessory to prove that it's been authorized by Apple by responding with their Apple-provided certificate.

Once it's been verified by the device, the device then sends a challenge that the accessory has to answer with a signed response, and this all happens without the user seeing anything. We plug in the device, and it works.

Chapter 15 IOS Jailbreak Types

When it comes to jailbreaking, this is different than rooting, and if you want to put this into terms, jailbreaking is bending the rules to overcome some of the existing restrictions.

Typically, when it comes to jailbreaking, we want to remove restrictions that Apple has placed. Most people will jailbreak their device so they can load applications that Apple doesn't necessarily want you to utilize for whatever reason.

Reasons often related to marketing issues. Like being able to block pop-up ads. That's how some of the free game manufacturers and developers make their money through advertisement.

We're going to accomplish jailbreaking by loading up a custom kernel that is used that has root access. By the way, “root access” in IOS has a different term associated or a different definition versus root access in Android.

Once the user can give himself this control, he's able to download applications and software that the app store doesn't allow. It also allows for customization like adding themes or extensions that Apple doesn't support.

It also gives you the ability to unlock the phone. This is how you can get one ISP versions of the iPhone to work on the other ISPs network. Therefore jailbreaking allows you to use software that Apple doesn't think you should have access to.

One of the biggest ones here is ad blockers. There are other apps that allows you to hide apps, hide badges, get rid of certain elements, customize the date text, change the lock screen and things like that.

Another great item out there was one that's called Activator, which allows you to assign custom actions to gestures and button presses. So if you wanted to launch Twitter for example, you could do a three-finger swipe and launch Twitter.

Or you could short-hold the Home button to fire up the camera. You can also change your toggle settings so that you could have a toggle for a VPN or personal hotspot.

You could even bypass the tethering option so that whether your carrier allows you to tether or not, you turn your device into a hotspot, and as far as your carrier is concerned, it's the iPhone that's requesting the data.

When we are jailbreaking, we are modifying the security of the device, and

therefore we have to be careful. As long as you know what you're doing, it’s ok. But, for example, don’t jailbreak a device and hand it over to someone who doesn’t know what you have done with the device right?

There are different types of jailbreaking that you can implement. The first one is called untethered, which has the attribute that if a user turns a device off and back on, the machine starts up and the kernel will be patched without any help of a computer.

The other type is called a tethered jailbreak. In this case, a computer is needed to turn the device on each time it's rebooted. If the machine starts up back on its own, it's no longer a patched kernel, and it gets stuck in that partially started state. By using a computer, the phone is essentially re-jailbroken each time it turns on, which sounds like a hassle.

There's also asemi-tethered jailbreak, which means that when a device boots, it will no longer have a patched kernel. It won't be able to run any modified code, but it's still usable for standard functions such as making calls and texting.

To use any features that require the running of modified code, the user must start the device with the help of a jailbreak tool for it to start with a patched kernel.

Besides the types, we also have the exploits that can be used to root the phone. The first one is called the Userland Exploit. This is where software running on the iOS device after the kernel has started.

This particular exploit uses a loophole in the system application. It allows user-level access but does not allow iBoot level access. These types of exploits cannot be tethered, because nothing can cause a recovery mode loop, and these types of exploits could and have been patched by Apple.

We also have the iBoot Exploit. The iBoot Exploit allows jailbreaking, and it also allows for user-level access and iBoot level access. It's an exploit that could be semi-tethered if the device has a new boot ROM.

This particular exploit turns off code signing and runs a program that does all the work for us, and this exploit can be patched with some firmware updates.

We also have Bootrom Exploits. This exploit uses a loophole in the secure ROM, which is the first bootloader. It disables the signature checks, which can be used to load a patch or firmware.

Firmware updates cannot patch these types of exploits. The Bootrom Exploit allows user-level access and iBoot level access as well. The only way for them to repair this is for a hardware update of the Bootrom by Apple.

There are different solutions based on the devices themselves, and you'll want to do research on that. We're talking about the security ramifications of supporting these devices but shortly will discuss how to start preparing an iPhone for Jailbreaking it.

Chapter 16 IOS Jailbreaking

This chapter will focus on explaining how to Jailbreak an iPhone 5. The things you'll want to do is use iTunes to back up your device before you do any jailbreaking, just in case you mess things up.

Once you do that, you have to use an application called Cydia. Cydia is what's going to get installed after the device is jailbroken. You can use a jailbreaking app called Pangu, and they've got different versions of it even including up to version 10.3.3.

This particular iPhone that I'm using is running on IOS 10.3.3. First, you have to download the Pangu application for Windows from https://pangu-jailbreak.en.lo4d.com/windows .

This could take some time to download, so while the download is in progress, you can connect your iPhone to your laptop with the standard cable. Once the Pangu download complete, you have to launch the application.

Next, it will detect your device. It will see what version of IOS you are running, and it will tell you that you should click the “jailbreak” button. Go ahead and click that.

Next, it will say that you need to adjust the system date to be equal to July 3rd , 2015, and it will provide instructions on how to do that. The reason why they do this is that its part of the vulnerability.

Next, switch back to your mobile device and go ahead and do what it asks you to do. Go into time and date under the General tab, turn off the automatic time, and change it to July 3rd, 2015.

Next, go back to your main screen, and switch back to the computer, and you should see there that it's injecting the bundles automatically. They give you the warning that you should back up your device, and you need to specify that this is not the same for every other device that you have under your Apple ID.

Each device is going to have its jailbreaking technique. Next, it will tell you that you need to press on the Pangu icon on your device. Once you have switched back to the device, you should have an icon for Pangu.

Click on it, and then you should see a warning message saying: "Are you sure you want to open the application Pangu from the developer tool? "

Because it's not coming from the app store, go ahead and hit Continue, and now

switch back to the computer. You should see that it’s implementing the exploit at this point, while it will tell you on the screen that it's rebooting and don't disconnect the device.

You can switch back over, and you should see the phone is rebooting. Once it's rebooted, go ahead and switch back to the computer screen, where you should see that it's just waiting for you to turn the device on, or swipe.

Next, it will ask you to log back into the phone, and now it's processing the jailbreak files, and your phone is officially jailbroken. Switch back to the phone, and the last thing you will want to do is back on the phone you want to go back and change your settings.

Change your date back; otherwise iTunes will have a problem with it. You can set the time for automatic. Disconnect the cable otherwise, and it will keep on rebooting. Next, go back to the menu where you should have a new app called Cydia.

You'll see that the Cydia app is now installed, and this is where you can load up applications that Apple doesn't necessarily approve of. The first time into Cydia, it has to change up the file system because you have jailbroken the phone.

You are going to gain access to some locations that you usually wouldn't have access to, and this can take a while. You can reboot the device again, because of the changes to the file system.

Once the device is rebooted, swipe again to log in and launch Cydia. From here, you can start looking at different apps that are available to you. You can go into your featured apps.

Here, you can find many such as the one called “NoSlowAnimations”. You can read what it's going to do for you, but in a nutshell, it’s going to get rid of all your animations, and it will speed up your phone.

You can also do icon customization, or you also have “WinterBoard” that allows you to do a ton of customization to the display.

When it comes to Jailbreaking, each device is completely different, depending on the IOS they are running and the device itself. Therefore, you want to do a lot of research. Make sure you check your sources, and remember that doing something like this, you could break your device.

Chapter 17 Securing IOS Devices

When it comes to Jailbreaking an IOS device, even you open up an overabundance of options that Apple doesn't necessarily want you to have access to, and you just need to know what those ramifications can include.

Once you gain access to the device remotely, think of that from the attacker's perspective. Still, the biggest issue with security is at the physical layer. If an attacker gets physical access to a device, either if it's Apple or Android, a PC, a laptop, or a server, they are done.

With mobile devices, we need to take into consideration what risks come into play. As we're moving around and taking our devices with us, we do need to make sure that we secure down the doorways.

First of all, what do you do if you lose your phone? Well, you want to make sure that you use the “Find My iPhone” feature so that you can quickly recover the device before it gets compromised.

I say this is one of the features of the Apple product line that helped to change the security industry. You also want to make sure that you protect your data using the iTunes AppleID.

This can also be used to detect if an application or somebody has jailbroken a device. There's jailbreak detection that you can turn on. Also, your Google accounts. Make sure you protect access to those.

Moreover, you don’t want sensitive information being backed up to the iCloud, where anybody can hack your password. iCloud security has been improved significantly over the years, yet private pictures have been stolen before.

Apple has fixed a lot of these problems already, but when it comes to hacking, eventually, somebody finds a way around it. The majority of people don't know that their data is being backed up automatically through iCloud.

Most people don't remember during the initial set up of the device, because they are so excited, but this is also true with Android people too. Manufacturers want to help protect your data, and by default on many of these devices is to back things up to their cloud service.

Jailbreaking does open up some freedom, but it also makes you a possible target as well, and at the very beginning, I would never jailbreak a device for someone that has no tech knowledge.

Regards to social engineering, if you get an email from somebody that you are not expecting from, and it has an attachment, do not open it on a mobile device.

Likewise, you want to make sure that you have a passcode to your device, and when typing it, nobody can see what you are typing. It's complicated to recover sensitive documents and your personal information, both financial and identity-based than it is to type in a code.

Also, be careful about add-ons that require Java or might silently install Java. You should check out all the requirements and settings of each application before you install it.

Moreover, don't jump on any WiFi just because it's free. Make sure that you're only hooking up to WiFi networks that you genuinely do trust. Also make sure that you don’t click on any attachments, or a link in any email, or even on instant messengers that people might send you.

Many times these links are shortened, so you have to be careful with those. Also, make sure that you only use trusted third-party applications and know who they are. Do some research on them.

Don't get fooled because of how great the utility is or the application is or how popular it is. Do your research first. Just because cool kids playing with a particular app, doesn't mean that the app is good as far as security is concerned.

Similarly, if you jailbreak the device, you've to change the default password. The default password for root access on a jailbroken Apple device is alpine. In Cydia, their web page is laid out for an Apple device, where you will notice that you've to change the root password.

For most attackers, the best approach that they have is merely waiting for exploits to appear, and then they repurpose them so that they can target users during that time when nobody comes up with a patch for that vulnerability.

In summary, we first looked at the architecture of IOS, which is a little bit more compartmentalized, but similar concepts of what we discussed on the side of Android.

We also made sure that you understand the device itself, and we talked about where the operating system came from. That was through the company called NeXT, founded by Steve Jobs.

After that, we also took a look at how to jailbreak an iPhone 5, but remember that it's not a solution for all Apple devices, as there are different jailbreaking techniques used for all different devices.

Each device has its separate trick to be deployed. Therefore you have to do your research before Jailbreaking your specific device. We also looked at the possible consequences of jailbreaking and how to be able to see that a device has been jailbroken.

In this case here, the easiest way was by seeing that the app called Cydia has been installed. We also look at some best practices on how to secure the device. At an enterprise-level, we'll want to control all these devices or be warned of the possibility of a device being jailbroken.

Sometimes it may be that the user is trying to jailbreak the device, but it also is because the user has downloaded a malicious document that is executing a jailbroken technique so that the attacker can take over the device. Now that we've covered both Android and the Apple platform, we're going to take a look at other mobile devices that are out there.

Chapter 18 Windows Phone Architecture

Now that we've covered both Android and IOS, let's discuss hacking other mobile platforms, and I want to begin with Windows Phones. When we talk about Windows Phones, it does depend on which Windows Phone you want to look at because Microsoft is making a huge transition.

There are the original Windows Phones, or also known as WP, which had several different versions associated with it, including Windows Phone 7, Windows Phone 8.1, as well as Windows 10 which is the successor to Windows 8.1.

Microsoft has been marketing this as being an addition of the Windows 10 environment, and this is all part of their strategy of creating this unified platform environment where we will have one OS to connect them all.

And as far as the platform is concerned, the new architecture is dramatically different because it's following the desktop platform very closely. OEM partners provide most of the low-level hardware interaction and the drivers that are used to boot the phone in a Board Support Package aka BSP.

The core of BSP is written by the vendor that creates the CPU, and then the OEMs responsible for adding the drivers that are required to support the phone hardware and the different components.

OEMs are responsible for adding drivers that are required to help each phone that's provided by the vendor. The kernel in the OS will come from Microsoft, and since this is an iteration of Windows Phone 8.1, it's based on the NT kernel.

These devices use the core systems from Microsoft, which is a stripped-down Windows operating system that boots and manages hardware as well as resources such as authentication and different types of communication.

As far as handling the phone-specific tasks of these devices, there's a supplemental set of Windows Phone binaries that form the mobile core. Some of the great features of this environment include NFC support on Windows products, as well as the support for both C and C++ porting, which is supposed to help porting from both Android and the IOS platforms.

There's also the Wallet feature, which is supposed to be a secure place where you store digital versions of credit cards, coupons, tickets, which is similar to Apple Pay or Android Pay.

Another feature they presented was taking Bitlocker from the PC and bringing it

over to the mobile platform as an encryption mechanism. Windows Phones also support UEFI, and they also introduced the ability to scale their platform to larger screens, while they also support multi-core processors, which would include up to 64 cores.

From the security perspective, Microsoft is also supporting app sandboxing, while we know that the applications are going to run inside of a sandboxed environment, so they don't interfere.

Moreover, they also included the features of supporting both VoIP and video chat integration for any VoIP or video chat app, and this is one of the reasons why we see Skype being installed by default on Windows devices.

When it comes to jailbreaking the Windows mobile platform, there are a couple of reasons why people do this. Mainly, it's to sideload apps and to install a custom ROM. The custom ROM is still going to be Windows-based, but it's not going to have bloat associated with it. There are several methods that you can use to jailbreak the Windows platform, but we want to make sure that you're aware of what can happen in the background of these different devices.

What you need to know us that the Windows Mobile platform does a secure boot, or SafeBoot. The overall goal of SafeBoot is to make sure that the OS, as well as any application that launches, is trusted.

The trust comes from the application stores, so when you first turn the device on, the firmware starts a Unified Extensible Firmware Interface in the background, that ensures the hash of the keys that are stored, and the device is compared the signature of the OS to the Boot Manager, which confirms if OS is legit. If the signatures that are compared pass, then the Windows Phone Boot Manager is permitted to continue to start up, in which case, we then start relying on the binaries of both Microsoft as well as the OEMs.

This is because the OEMs will have their applications that are pre-loaded on the device signed by Microsoft. This is used to protect the application and the boot system from any type of malware from being injected.

This is advertised as no one can access the keys that are required to start the system up, but in the real world, that’s not always true. Microsoft has also reduced the footprint of the operating system. Therefore all the applications will execute in the same sandbox as a third-party marketplace app.

This, in turn, extends the customization of the OEM drivers, so if an attacker tries to mitigate the application with malware, it can only access the content

inside of that sandbox. Therefore, it prevents malware from gaining access to any of the core-level applications and hardware on the device.

First and foremost, you should only run apps that come from a trusted source. That might be the App Store by Microsoft or an OEM app store. Also, make sure that you lock the SIM with a PIN, and that PIN being a passcode that you type in when you turn the device on.

You also need to make sure that you run Windows Updates and have proper backups. Just like if you root or jailbreak a device, make sure that you back it up in case you mess something up, thus you can always restore the original environment.

Moreover, do not connect to wireless access points that are unsecured. Often people are socially engineered as they hook up their devices to Wireless access points because they believe that they are going to get free internet.

And the next thing they know, everything they're doing on that network is being monitored, logged or worse, if the traffic gets modified. You should also ensure that you have a lock screen, so when the device turns off or goes into sleep mode, you have to unlock it to continue to use it.

If you can utilize biometrics security such as fingerprint scanner, or facial recognition, you should consider using them. In particular, when it comes to these Windows Mobile devices, you can lock them down with the Lockdown.xml file.

This file can be used to restrict which apps users can gain access to, as well as which app stores they can go to. For example, an end-user is allowed to reach Windows App Store but able to access any social networking apps because you filter it that way.

As a corporate mobile device administrator, create your store that's used strictly within your network infrastructure, and you can copy those files from the publicly available store, and make them available in your enterprise-level store, and ensure that all the devices are only able to install apps from your store.

You can also use it for customizing the layout because perhaps you want certain apps or specific shortcuts available on Windows Mobile devices. To lock down Windows Mobile devices with the XML file, there is a step-by-step guide created by Microsoft that you can visit at https://docs.microsoft.com/en-us/windows/configuration/mobile-devices/lockdown-xml

Chapter 19 BlackBerry Architecture

To hack BlackBerry devices, it depends on which device we are talking about because BlackBerry also has gone through a transition between BlackBerry OS and BlackBerry 10. Both these operating systems are proprietary to BlackBerry devices.

As far as BlackBerry OS is concerned, it was prevalent back in the day when it started supporting corporate emails. BlackBerry's no longer dominant in the mobile market, and here is why.

BlackBerry OS has discontinued the release of BlackBerry 10, that was a complete rebuild from the ground up. It was powered by QNS, which BlackBerry purchased back in 2010. This particular operating system uses a combination of both gestures and touches for navigation control, and this is why it's popular in some of the interfaces we see in cars nowadays.

In 2014, they released Version 10.2, which gives the accessibility to download Android apps. Next, in 2015, BlackBerry said that they had no more plans to update the APIs or the software development kits, which would also include any runtimes of the operating system.

They said that any future updates of the platform are going to be strictly focused on security and privacy enhancements. Then in 2016, BlackBerry has announced that they will be releasing 10.3.3, which will be primarily targeted at providing more security and privacy enhancements.

When you look at either version of the OS or 10, we're talking about BlackBerry OS, which technically has been discontinued, but both of them are proprietary to the BlackBerry name, and they both support a Java-based framework that allows the Android runtime and Android apps to run on these devices.

When it comes to securing blackberries, make sure that you encrypt these devices, and activate the BlackBerry Protect feature, and implement password encryption.

Chapter 20 Mobile Device Management

Speaking from a traditional IT perspective, we'd like to manage our desktops and laptops with some devices. When it comes to adding in all these additional mobile devices, we need some way of managing them.

Therefore, we'll take a look at MDM, aka Mobile Device Management, what those are, as well as how to evade MDMs. Introducing MDM into our environment, it also becomes a target, so we'll also want to take a look at how do we protect that device. Next, we will also look at something that helps us to detect jailbreaking or rooting of devices, so let’s get started.

MDM, or mobile device management, it's a software used to monitor all your mobile devices, allowing you to deploy apps or secure anything from smartphones to tablets.

The whole purpose of MDM is to optimize and automate the functionality and security of your devices throughout the enterprise, whether they're on the corporate network or NPE aka non-production network.

Typically, you can control things such as allowing the distribution of applications, where data is stored, how the device is configured, which patches to install, as well as how to handle the machine that's been compromised.

Naturally, MDMs work with frameworks, and each device manufacturer has its structure that is injected inside of the MDM. The MDMs usually have their policies and features that mobile device administrators can control or enforce on these devices.

The combination of these policies and features form the frameworks such as Android, Blackberry, or Apple, provide their MDM framework that allows administrators and MDM vendors to link into their devices.

Different MDM software solutions are out there, and one of the most popular ones is called AirWatch, which is handled by the same company like VMware and System Center Configuration Manager.

The latest addition from Microsoft, supports the mobile Windows platform, Android and iOS as well. It does require an additional component called Intune, which is a obtainable with a monthly subscription to be able to manage your devices when they're not inside of your network infrastructure.

Intune acts as a proxy, while it has a connection to the internet. Your devices make a connection to it, and it relays it back to your System Center Configuration Manager on the back end.

Another product that's worth mentioning is called completely free Spiceworks. Which one is better? Well, you want to find one that supports app provisioning so that you can force the installation of applications on mobile devices.

With app provisioning, most IT guys and most MDM products will also allow

you to filter which applications are available to the devices via the app stores, or you can create your internal app store.

Another feature you want to take a look at is security, and there are different levels of security that you want to look for. First, you want to look at the safety of the MDM itself.

Also, how does it communicate outside of the network if the device is externally implemented. Some of the MDM solutions will allow the IT professional to limit the number of pages or files to be downloaded and require passwords to access online data.

We're also starting to see a lot of MDMs take advantage of the infamous TPM chip. This way, you can make sure that the users encrypt their device, and the chip acts as a decryption key for you.

You'll also want to look at other security mechanisms, such as can you control what Wireless access points your users can use. For example, maybe you want to do a blacklist on any wi-fi access point that has the word free in it.

Another feature you'll want to take a look at is the enrollment process. Technically, it would be both authentication and enrollment. This is what locks the device and the user and the organization together.

Some MDMs can be set in a way so that it responds to each user according to their role-based permissions. For example, a user typically downloads an app, which then leads to a sign-in screen for the MDM site.

At this point, the employee has to use their credentials from Active Directory to authenticate who they are. Then the device gets enrolled, and we can then control it.

The remote access feature is also trendy. This also could allow an administrator to remotely disable or even wipe the device if it gets reported stolen or lost.

Another feature is called MEM, aka mobile expense management. Some MDMs will help you compile cost data for mobile devices. For example, you can see how much data they're using in their data plans, how much voice traffic, how many text messages, so an administrator can help keep costs down for the company.

Most of your MDMs are also include their own set of policies. Moreover, when it comes to provisioning a device in which you need to deploys and enforce policies, as well as restrictions you may have for mobile devices, and MDM can help you.

The MDM deploying a client application on the device through some mechanism, and once that’s complete, the app compares itself as it reports into the MDM to a policy file.

Once you have a new app on the device, that will create another attack vector. There are two different files that most MDMs will install or utilize by most MDM products.

It's called the “EffectiveUserSettings.plist” and “Truth.plist”. Both of these are system files that help determine the IOS device's security settings. The “Truth.plist” has information in it, such as PIN and passcode policies, as well as device timeout settings and restrictions.

For example, you could disable alphanumeric enforcements of a PIN, or the maximum grace period so that you don't get locked out after four PIN attempts. Additionally, you can set the Maximum inactivity setting. For example, when you don't use your phone, it locks out in a couple of seconds.

You also want to ensure that your MDM supports the ability to detect modifications.

This is going be done by the policies of the MDM software, and in some cases, some of the solutions are if you detect a change in the XML file on the device, you re-push the master XML file back to the device. You can also set up automation, so if these files have been modified, you lock down the device or wipe it.

Another thing that you want to make sure is that your MDM protects you from patching and modification attacks. These are platform-based attacks, so there are different types of attacks for different kinds of devices.

For example, Android uses self-signed certificates, so an attacker could easily modify the binary of the Android application to patch existing functions, or even inject a new code.

An attacker can also execute the Android package manager command on a device that installs or uninstalls applications. When it comes to the IOS platform, attackers use “injection”. A mobile substrate does this.

The mobile substrate is a common framework that allows applications to do runtime patching in IOS. Using these types of structures, an attacker could patch the jailbreak detection feature of an MDM, so it would never report that the device had been jailbroken.

There is also an issue with decompiling. Applications, especially on the Android

platform, can be reverse-engineered. Both; Java and Dalvik have tools that allow you to decompile or disassemble the representation of the application that is stored in the “Dalvik byte” code or inside the Java code.

By doing so, the attacker can get a high-quality version of the source code because it's been installed on the device. An attacker can use several different tools to accomplish this, such as DEXtoJar or APKTool.

On the IOS, mobile applications are written in Objective-C, which is a relative of C. Therefore, we have programs such as class-dump,class-dump-x, and class-dump-z that we can use to reverse engineer our iOS apps.

Because IOS is restrictive when it comes to applications, using these extractors to look at the application binary, doesn't give you any insight. It provides a debugger with the ability to correlate and understand what the author was doing as far as the logic and the function of the application is concerned, which is the first step in any ethical hacking.

How do you protect from this? Well, you can use obfuscation. There are several programs out there such as ProGuard that obfuscate the code that an author has created, so it can't be reverse-engineered or at least slows the attacker down.

When it comes to detection, you want to make sure that you can find out if the devices are being jailbroken or rooted. MDM solutions have built-in client applications that detect devices if they are jailbroken or rooted.

Typically, this coding is done very basically, because in the case of Apple, most people like to jailbreak their device so that they can run the Cydia app store. Therefore, most MDMs look at if an alternative app store has been configured.

Chapter 21 Security Recommendations

One of the most noteworthy issues with the mobile platform is that people load up too many apps. Therefore you should be careful about automatically uploading photos to social networks. People often share way too much information.

Sometimes people don't even realize it. Every time we take a picture, we post it up on Facebook. Well, did you look at the photo? What's in the background? Are you exposing your address?

So be careful that you don’t share too much information publically. Also, make sure that you can maintain control of mobile devices. If you lose control of them, it's time to get them back in the office and reassociate.

Make sure that you are also doing security assessments on both; the devices themselves as well as the application and its architecture. Make sure that you only install apps from trusted locations.

Likewise, don't add any location-based features, unless there's a component that supports the application. For example, don't load Google Maps if you're not going to need Google Maps.

Typically, a lot of the Android devices come with those applications pre-installed, but on the same aspect, you also need to be careful of apps that want to look at your location.

Also turn your Bluetooth off if you don't need it, especially while traveling. Only turn on your Bluetooth when you need Bluetooth, plus you'll get a little bit extra batter life too.

Similarly, don't connect to two separate networks such as Wi-Fi and Bluetooth at the same time. Furthermore, make sure that you do backups, and check how often your devices are synchronizing, especially if they're synchronizing outside of your network.

You also want to ensure that you're using strong passwords. When it comes to passwords, ensure that you set the idle timeout to automatically lock when the phone is not in use. When you hit the power button, it should automatically locks your phone. Make sure that you take advantage of the “lockout and wipe” features. If somebody types in your code incorrectly four times, your device should get wiped.

When it comes to jailbreaking and rooting devices, the best practices are that you

should never allow rooted or jailbroken devices in the corporate environment. Also, make sure that you keep every application up to date, as well as the OS itself.

This is relatively easy because most of our app stores tell us that you need updating. Here, you should review the changes that the updates are wanting to accomplish, instead of just hit “Apply All”, because you could be lowering the security profile of your device.

Also, take advantage of hardware encryption. Hardware encryption is like encrypting the hard drive on a PC. If somebody steals your PC, and if they don't know the encryption key to type it when they boot it up, they're not getting the data.

Moreover, review your MDM policies, and this comes down the road of change management because when you come up with plans, you need to make sure there's a complete agreement across departments of what is allowed and what's not allowed on mobile devices.

Some IT guys think that they know everything and know what's best for everybody, but we all work differently, so you want to discuss with your manager or other team managers what to include in these policies.

Moreover, make sure that you filter email forwarding barriers, because most of our malware coming across email, and you don’t want to head out to your devices.

With some of the newer email servers out there, you can allow only emails that are generated internally, so only those get forwarded to corporate mobile devices. Also, look at permission rules, so when somebody is trying to visit a particular page on these devices, they're not getting injected with a code that's going to jailbreak them or root them.

Also, make sure that you are only using applications that are signed by the app stores, or you've approved them for internal use only. Likewise, make sure that you set up erasing data to keep people from guessing passwords on the devices, and ensure that you have auto-locking turned on.

When it comes to backups, ask yourself these questions: Where are you locating those backups? Is that place is in a secure location? Require a passcode where you swipe a pattern? This is because studies show that people are not using patterns.

Also, make sure that you have a software maintenance plan in place. For

example, if there is an app installed on a device and it’s not been used since 90 days, most probably don't need it.

Also, make sure that you continue to sandbox the data and the app. This way you don't have to worry about any cross-infection.

When it comes to Wi-Fi, make sure that the person has to ask to join. This is because one of the new features of Windows 10 is to automatically join known Wi-Fi access points, not the ones the end-user is aware of, but the ones that Microsoft is aware of.

When it comes to emails, don't allow for the caching of emails because even you've deleted the email, it's still in the cache and if someone is going to get a hold of that device, they can read that information.

You also, as a company, need to sit down and decide if data can leave the environment. For example, do we allow people to send the payroll spreadsheet via email? Why not use the newer technologies such as ShareFile options, where you don't send you a copy of the file.

Instead, provide a URL link back to your SharePoint server so that users can read the files. Then you don't have data leaving the environment.

You may also want to consider Citrix from the mobile device. Instead of putting the data on your mobile device, the user opens up a Citrix session back to their PC.

When it comes to Android, check what Google is backing up, what they're storing, what they're caching, and what data that they're collecting. Also, make sure that notifications are turned off on the lock screen, so people can't see the notifications. Notifications should only be visible to users if they unlock the device.

Make sure you turn off “AutoFill”, and train your users on how to use these devices. What's acceptable and what's not acceptable? Make sure that there are written policies in place, so they understand what the consequences are and you hold them to those consequences if they break them.

You also want to consider when users are getting in remotely, how long do you allow them to stay inactive before the session times out. They might have to type in a username and password again, but do not cache domain passwords.

Chapter 22 Spiceworks & Solarwinds

Spiceworks is an ad-supported application aimed at the Small and medium-sized business aka SMB, designed to perform inventory and necessary monitoring of Local Area Networks. The product is excellent for about 250 employees or less, which is vital to remember when evaluating the product's functionality and features.

The software is targeting system administrators who may not have a deep appreciative of technologies or who may lack proper education, training, and certifications.

Therefore, the product's feature set, while being productive, offers much less functionality of products such as Solarwinds Orion. Still, Spiceworks is free software that you can try out by visiting their website at https://www.spiceworks.com/downloads/

The software is a desktop application, and the installation is very easy for a Windows app. Once installed, it requires a username and password for an admin account.

The detection feature across the network is very fast, but the system is slows down during the network device detection process, but it’s not too bad, and the load is not noticeable on the client devices.

The detection procedure can be scheduled to be rerun intermittently as well, so you can keep the info database always up-to-date. The end user can access the desktop via a Web GUI, and the underlying process can be run as a service.

This enables the software to be used on an ad-hoc basis to get a snapshot of the network condition. You can also run it a server and access it remotely. The app itself, while is resource intensive, is not as resource hungry as more enterprise-type software.

It operates by examining a range of IP addresses, and it utalises WMI to log on Windows devices and SSH aka Secure Shell to access Linux systems.

It can also monitor routers, printers, switches, and many other networked machines. The way it performs the detections and monitoring the machines, it requires no additional software installation or physical contact with the devices.

Moreover, you don't need to spend hours on complex configuration, such as with other traditional monitoring and management apps and software.

Once it detects devices on the network, it uses standard WMI queries and UNIX

shell apps to determine the system info and their status. This method esteems the security settings on the network machines in an easy-to-understand fashion.

But, if something prevents this from operating, Spiceworks marks the machine as “Unknown”. It does an excellent job cataloging and inventorying the software, hardware, services, and necessary status information as well.

One issue is that it detects only McAfee and Symantec AV products.

It includes a basic trouble ticket system as well, and alerts can be sent via SMS, and e-mail, or viewed via the Web-GUI. The Web GUI, in overall, is an easy to understand platform.

Spiceworks also has a reporting system that can create reports based on monitoring and cataloging functions to allow you to find printers low on ink, PC-s low on drive space, or where new machines added to the network.

The GUI also allows end-users to look at the network, and apply filters to locate a device or a group of devices. The GUI function also allows you to search for anything that you can monitor.

Another great thing is that Spiceworks, as a company, is forming communities with the end-users. So, for example, you can submit a question to the community. As you see, Spiceworks is very comprehensive and contains information from various vendors and end-users.

Moreover, they have a forum where devs can request new features, submit bug reports, or vote on feature ideas.

Because Spiceworks is free, it requires minimal configuration and does not install any additional software on the devices it manages, and you should give it a try. In case you find a feature that’s missing, you can request it, and I am pretty sure that they will include it in the future.

Yet, in case you need real-time alerts, a highly customizable system, or proper device-specific information, you should try out Solarwinds. Solarwinds Free trial can be downloaded at https://www.solarwinds.com/downloads

Still, if you only need an MDM to monitor mobile devices or just for inventory purposes, Spiceworks is an excellent choice for you.

Chapter 23 Malware & Spyware on IOS

There is no difference between what we know about malware when it comes to the desktop platform versus the mobile platform. Yet, it's interesting how Malware getting around some of the security mechanisms that are in place.

Typically it's being done by social engineering. As far as attackers concerned, when it comes to malware on the mobile device, they are just computers. Since mobile devices are only computers, it's the same type of environment for attackers too. We need to understand that a lot of the malware that we get is just a repackaging of older attacks on our systems. But, let’s begin with Malware on IOS.

You might be thinking, “WHAT? malware on IOS?” Well, there's malware out there for IOS. Apple is considered more secure than other platforms. Still, we open ourselves up to different types of malware attacks.

The first actual malware attack was called the iPhone firmware 1.1.3 prep tool. This malware disguised itself because it acted like this was a preparatory upgrade before you updated the device to Version 1.1.3.

Once you install this particular malware, several common utilities got installed on jailbroken devices that would stop working, and the fix for this was to reinstall those utilities.

This malware only infected jailbroken devices, but you have to remember that jailbreaking has been very popular throughout the Apple community. On jailbroken phones, the default password for SSH is “alpine”, and the reason why it's “alpine” is that that was the code name for the IOS version 1.0.

Well, a year later, someone ended up doing a full scan on T-Mobile's 3G IP range, and exploited this vulnerability and installed ransomware on those devices.

In 2009, a teenager released a worm that exploited the vulnerability called “iKee”. Yet, another attacker took “iKee”, modified it, and released it as “iKee.b”.

In this particular case, the purpose was to “Rickroll” the target, while the same default password was used again. “iKee” used the same SSH vulnerability by running a series of commands, so that it would propagate itself throughout the host.

First, it would delete the “sshpass utility”, and then would copy the “sshpass

utility” and the “worm” itself from the current mobile device file system to the remote file system.

During that process, it copied an image and replaced the background image for your locked screens on iPhones, and it was a picture of the 1980s British pop star “Rick Astley”. Here's is where things got interesting. After that, the scan began on the victim's IP address range that it was hooked into, to propagate out to new hosts.

In 2012, we ended up seeing a piece of malware that hit the “AppStore” called “Find and Call”, that automatically uploaded users' contact lists to the company's server. It then spammed those contacts with a link to the app, saying it was from your friend.

In 2013, a developer, who went by several different names, a more famous name that he was known by “Felix the Cat” included adware in his tweaks and these were sold through “Cydia”. This was known as “Nobiniz” and some of the apps that he put them inside of, included “Better Chrome” or “Chrome Downloader Enabler”, and many more.

He had a bunch of different apps or tweaks that he supplied, so his products were top-rated. He also had tweaks for Instagram, Twitter, iOS 6 Photos, Menu, and he added some new features to it too. He was popular, and he ended up getting into advertising before his apps got blocked.

In 2014 things started to get interesting, and it's because mobile devices started getting more popular. We first saw something called “Unflod”, which tried to capture users' “AppleID” and passwords. Once it captured those, it sent the information to Chinese IP addresses. This was also known as “SSLCreeds”.

We also had “AdThief”. This particular malware, which tweaked a developer's ID that was intended to tell ad developers who were presenting their ads, such as affiliation fees, it redirected those instead to the attacker's account. This went on over 70K devices at the time.

We also have “AppBuyer”, which was a piece of malware that would connect to a “C&C” server. It downloaded and executed malicious executable files, hooked network APIs to steal the device's users' AppleID and password, and then upload those to the attacker's server.

It also simulated Apple's proprietary protocols to buy apps from the AppStore by using the victim's identity. All three of these were targeted towards jailbroken devices.

We also have “WireLurker” and “Masque”. This malware targeted not only IOS but MacOS as well. This was one of the first pieces of malware that installed third-party applications on non-jailbroken devices, and they did that through enterprise provisioning.

“Masque” allowed an IOS app to be installed using enterprise provisioning, and it could replace another official app installed through the AppStore, as long as both apps used the same bundle identifier.

“Xsser mRAT”, was another piece of malware that would target jailbroken devices. This app installed a rogue repository on Cydia. Cydia, even though it's an AppStore, had different repositories that you could pull apps from, and this was a rogue repository that got installed.

Once it was installed, it gained persistence, meaning you can't delete it. “mRAT” then makes a server-side check and starts to steal data off the user's device. It also executed remote commands that were issued by the C&C server.

In 2015, we saw even more Malware come out for IOS. First, we have seen the one called “Lock Saver Free”. This was a free app that installs an extra tweak that hooks into ad banners and inserts it's own identifier into the banner so that the author of the Malware would get revenues for referrals instead of the actual app developer.

Next, we have seen another Malware called “XcodeGhost”. This malware was found in some re-distributions of “Xcode” that was targeted at Chinese developers. “XcodeGhost” infected the apps compiled with the versions of “Xcode”, and apps that were published on the Apple AppStore had to be pulled, once Apple became aware of this.

This also uninstalled the apps from your devices at the time. This malware added code that could upload device and application information to a CNC server. This was capable of creating iCloud password sign-in prompts that were faked and was able to read and write data from the copy and paste clipboard.

After that, another powerful Malware comes out called “KeyRaider”, which was used to steal Apple account usernames and passwords, and the device grid by intercepting iTunes traffic on the device. There were over 200K stolen accounts via “KeyRaider”.

“YiSpecter” was the next famous Malware that used its private APIs on both jailbroken and non-jailbroken devices. This particular piece of malware would download, install and launch arbitrary iOS apps, replacing existing apps with

those that it downloaded. It also changed Safari's default search engine, bookmarked and opened pages and uploaded device information that was sent to their CNC server.

“Muda” aka “AdLord” was also spread through Cydia. Therefore it only affected jailbroken devices. This one was setting up display advertisements over the notification bar and then asked the user to download ISO apps that it was promoting. Therefore the author would make money off the referral fee.

“Youmi Ad SDK” was another software development kit that was used by a lot of Chinese AppStore developers to take advantage of private APIs to get more personal information.

Personal information that Apple allowed, including a list of the apps installed on the device, serial numbers of the device, the user's AppleID, email address and so on. There were over 200 apps that got into over a million devices.

It's not limited to just attackers, because even governments are using pieces of malware to target individuals. One of these tools was called “FinSpy Mobile”. “FinFisher” is a suite of commercial surveillance tools that are sold to governments as legitimate programs, which have been used to target different activists and other people, so they could see what they're up to.

“XAgent” was another Malware. This also came out in 2015, which is also a surveillance tool that targeted specific people, other government individuals, military individuals, journalists, and so on.

This one could be installed on both jailbroken and non-jailbroken devices. This particular suite of tools would do a combination of a phishing attack called “Island Hopping”, so phones of friends and associates of the actual target were first infected, and then used to pass on the spyware link.

This was based on the social engineering aspect that a target is more likely to click on links from people they know. Once installed, it would collect text messages, contact lists, pictures, GPS locations, a list of apps installed, a list of processes that were running and WiFi status of the device and so on.

This is also the piece of malware that gets a lot of people concerned because it could switch on the phone's microphone and record everything that is heard.

We also have “DropOutJeep”. This is a tool that the NSA uses, and this malware can pull or push files to a device, look at text messages, contacts, voicemail, GPS locations, turn on the mic and so on.

There's been some speculation that this particular tool had some help from

Apple, but Apple has denied it, saying that they've never worked with the NSA to create a backdoor to any of their products.

We also have the “Hacking Team Tools”. This one's interesting because the Hacking Team itself got hacked. They sell offensive intrusion and surveillance components to governments and law enforcement agencies, or at least they used to.

Either way, this particular piece of spyware was designed to target specific people, instead of looking at broad surveillance of the public. Their primary tool was a remote control system, did require a jailbroken device, but they were looking at implementing this on non-jailbroken devices too.

There's also an attack framework that's out there called “Inception”. This particular framework was designed to steal information using phish emails. These phish emails were saying that it was an update from WhatsApp, and if you clicked on it, it triggered a download of a Daemon installer package. This was focused on jailbroken devices. They made it look like it was coming from a “Cydia” store, and once installed, the malware would steal your address book, phone number, MAC addresses and so on.

Other people have taken this same framework, and they've repackaged it and referred to it as “Cloud Atlas”. There are other targeted tools out there as well, such as “Copy9” and “Copy10”. These two are both spying tools that allowed parents to check on their child or what their child was up to.

Similarly, there's also 1mole and FlexSPY, and most of these again were advertised as “keep an eye on your kids”.

Then we have the traditional type of malware coming from the desktop side, such as the “iKeyGuard” and “iKeyMonitor”. “iKeyGuard” is a keylogging tool, and “iKeyMonitor” would keep track of what your kid was doing on their cell phone. There's also “InnovaSPY”, “mSPY” and “ownSPY”.

Moreover, there is also “MobiStealth”, “SpyKey”, “StealthGenie”, and “TrapSMS”. All these products would spy on either application, the full device, or look at specific applications like text messaging.

There are even tools out there for research, meaning proof of concept, looking at what else could be done. For example, we have “iSAM”, which affects both jailbroken and non-jailbroken devices, and it scans for jailbroken devices that have SSH running and the default password, and it can also jailbreak a device that hasn't been jailbroken.

There's also a tool called “Instastock”. This has demonstrated that there are flaws in Apple's restriction on code signing on IOS devices. The app is initially accepted and then pulled from the AppStore automatically.

There's also a device called “Mactans”, which was presented back in 2013 at BlackHat. This device looked like a charger, and if you leave it around, somebody will pick it up thinking it’s an Apple charger, plugging it in, it injects a malware on the device.

There is also “XARA”. This is also an interesting one because security people went through and found methods to do cross-app resource access, meaning getting past the sandboxing, and they were able to make attacks on the IOS platform. They have submitted some malicious proof of concept apps to Apple and the AppStore, and Apple approved it.

The researchers immediately removed the apps from the store, to see if it would get passed, which makes you wonder what the attackers are doing, right?

There's another tool called “Jekyll”. This is for researchers, so they can demo or show a method of getting malicious apps approved for the AppStore by masking legitimate features to evade the detection of the malware, but the malware would be triggered once the app was installed on the IOS device.

Apple got this app approved through the AppStore with this method, but later it was pulled by the researchers. “NeonEggShell” was another similar tool. This particular proof of concept was an interesting one because the researcher demonstrated how easy it was to take over an entire device with a piece of code that was no bigger than a Twitter post, which is no more than 120 characters.

Chapter 24 Malware & Spyware on Android

If you think IOS is terrible when it comes to Malware, well, it's even worse on the Android platform. Still, you have to remember why it is worse. The answer is because of the market share. There are many iPhones out there, but Android has more. Currently, at the beginning of 2020, 87% of the market share is owned by Android.

When it comes to Android, Malware doesn't require the device to be rooted. Instead, there is a tool called “AccuTrack”, which turns your smartphone into a GPS tracker. We also have “Ackposts”, which is a Trojan that steals contact information, and then uploads them to a remote server.

There is also “BankBot” that steals user's banking information and mobile accounts from devices. There is another tool called “BinV” which is also a banking Trojan, but it targets Brazilian users only.

Another tool called “Boxer” that’s also a Trojan that would send SMS messages to premium rated numbers so that it can inflate your phone bill. There is another one called “CounterClank” that had an extremely aggressive ad network, and it did have the capability of stealing privacy-related information so that they could target ads at you.

We also have “Dogowar”, which was a spam SMS messenger malware that would send out to all your contacts and result in you having fewer friends.

There's also “FakeAngry” which was a backdoor Trojan that could receive a connection from a CNC server. There's also “FakeAV”, which was a fake anti-virus product, but this one too could intercept incoming and outgoing phone calls as well as your voicemails.

There is also a tool called “FakeBank”. This opened up a backdoor and stole information. This was so bad that it also could infect a connected PC. So if you connected your phone to your PC to download your files, it would trick the users into exchanging their bank information, like their login information via a fake site.

Another tool was called “FakeNetflix” which was stealing user credentials from Netflix. “FakePlay” was an application that runs in the background and gathers SMS information, and then sends it to a proxy address that runs at a device administrator privilege level.

Android does have device administrators, and your applications can request that

level of authority. A good example is an application that stores passwords and generates passwords for different sites. This is for making sure you don’t have the same passwords on all the websites that you are using.

Well, one of the features of that application is that if you make it a device administrator, it can then monitor your device, so for example, when you launch an app for internet banking purposes based on your thumbprint, it will auto-fill in your information for you to log on.

There's also another tool called “GingerBreak”. This particular piece of malware would root your device without your knowledge. There are also tools like “MMarketPlay”, that was focused more on the Asian market because it was a Trojan that automatically blocked applications on the Chinese Android Marketplace.

There's also “Obad”, which was a multi-functional Trojan Malware. It could send SMS messages to a premium-rate number, and it could download other malware programs, install them on the infected device, or send them further via Bluetooth, and remotely perform commands from a console.

“NikiSpy” is another piece of malware that would gather information such as the IMSI or IMEI number of the device, GPS location and then would upload it to a URL.

There's also “SMSZombie” that was extremely stubborn and hard to remove, but its primary function is that it was used to exploit a vulnerability in the mobile payment systems used by Chine Mobile.

We'll cover mobile payments shortly, but some of this Malware I mentioned are more current, and a lot of them deal with banking or mobile payment applications.

There's also “SlemBunk” that pushes out a fake login interface when the application's running in the background, and it does that in the effort to phish for credentials for your banking institutions.

The app stays incognito after running for the first time, but it just sits there and monitors the activities that the device is experiencing. It can detect when certain legitimate apps are launched, and it displays the corresponding fake login interface.

From there, those credentials are then transmitted to a remote CNC server. This particular malware is disguising itself as a Flash player or a WhatsApp update on the infected system. There are over 70 legitimate apps for financial companies

that can emulate its login interface.

There's also “Cetifi-Gate” which is a vulnerability that allows an application to gain access or privileged access rights on a device. This allows the attacker to steal personal information, track device location, turn mics on or off, and much more.

What's happening here is that the attacker is using a vulnerability that allows them to take advantage of insecure apps, certified by OEMs and carriers to gain unrestricted access to any device that includes screen-scraping, keylogging.

There is many more Android Malware that I could mention, but I am sure you get the picture of how bad it is when it comes to the Android platform. Instead, let’s move on and discuss Mobile Payments.

Chapter 25 Android Pay & Apple Pay

I want to discuss mobile payments in terms of you as an end-user making payment via your mobile device. When it comes to security, if you look at the technology that's being utilized in the back end, it looks better than any of the credit card technologies that we're still using.

Still, we are about to take a closer look at what's happening when we try to pay with our devices. We'll look at Secure Element versus Host Card Emulation, as well as Android Pay and Apple Pay.

Then we'll take a look at other mobile payment options that involve our mobile devices, which would be our mobile credit card interfaces. Let's begin by looking at Secure Element and Host Card Emulation.

They are also known as SE versus HCE. Let's first talk about what Apple does with the secure element. When it comes to the secure aspect, you need to understand that Apple wasn't the only player in this particular technology.

Google used to use this as well, but they've moved on. The way this works with secure elements is that on your device, you've got several different mechanisms in place that help with the communication when you're trying to pay for something with your phone.

With SE or secure element, it stores the credit card and cardholder data and does a cryptography process. During a transaction, what happens is that the SE emulates a contactless card using industry-standard protocols to help the whole authorization of the transaction. The SE itself could be either embedded in the phone or could be embedded inside of the mobile provider's SIM card.

HCE is a hard-based card emulation. This is where the card emulation and the secured element are in separate areas. When it comes to host card emulation, you need to understand a couple things.

First of all, again, Google used to use SE but they moved over to HCE. In fact, Microsoft announced that Windows is supporting HCE. What happens is that there is no more secure element involved.

We're going to separate this into different areas. When an Android phone is tapped against a contactless terminal, the NFC controller inside the phone redirects the communication from the terminal to the host operating system and the phone's HCE implementation guarantees that any NFC data received by the processing app that was obtained directly from the controller.

Therefore, there's no way to spoof a payment app with data from another source because they are sandboxing it. Because the app is sandboxed, it ensures that its data isn't available to the other apps on the device.

Performing all those functions of a secure element in software still has a security risk associated with it. But to help us out, Visa and Mastercard are implementing cloud-based stable features, and what this means is that some of the functions that the physical secure element provide will be performed on Visa and Mastercard servers over the internet.

Android Pay

Android Pay is the replacement for Google Wallet, not that Google Wallet is wholly gone. When it comes to Android Pay, it supports fingerprint scanning, but it's not necessarily required. If your device doesn't have it, you can use a PIN. If possible, you want to use the fingerprint feature because biometrics is much safer, but if you do have to use a PIN, don't use a pin like 1-2-3-4.

Another feature that Android Pay has is the ability to support dead zone transactions, and what we mean by this is that you might be in an area where your phone doesn't have service. How do you go through an authorized payment?

Well, you don't want it to go unlimited so that Android Pay can do a limited number of transactions in these dead zones. How does it do that? Well, you can still use secure elements if you need to, because it's backward compatible.

Whenever you make a purchase, a token is created, which replaces your real credit card number with a 16-digit number, and this is called tokenization. The other advantage is the ability to have reward cards associated with your Android Pay applications.

Android Pay isn't just for buying things in shops because you can use it with apps as well if the developer puts in an Android Pay button. Google Wallet also reintroduced as a peer-to-peer payment app, where you bump phones together or you even put your phones next to each other to make a payment. In terms of security perspective, most of these apps have security mechanisms to require you to verify either through fingerprint or PIN that you want to transfer that money.

Apple Pay

Apple is using the NFC chip inside the iPhone. It's also inside of the Apple Watch. It does rely on fingerprint scanning. Apple Pay currently accepts payments over 2M retail locations, and Apple gets a cut of all the transactions.

When you set up Apple Pay, a lot of people get concerned because the first thing you do is you take a photo of your credit card. You don't necessarily need to worry about this because according to Apple, whether you take a picture or you type in your credit card information, it gets encrypted and it's sent to the Apple servers.

Apple then decrypts the data, figures out who your payment system is through, such as Visa or Mastercard, then re-encrypts the data with a key that was provided to them by the credit card company.

It then sends that encrypted data along with some other information like your iTunes and App Store account activity, information about your phones such as the phone number, the name, the model, and your geographic data, and the bank then looks at that information, determines if that's you, and issues back a DAN, aka dynamic account number.

This gets issued back to Apple. But Apple doesn't have any way of decrypting it, so they create a dynamic, secure code which is then stored on your SE.

Mobile Credit Card Interfaces

There are several mobile credit card interfaces out there. One of the more popular ones all has mobile card readers that we plug into our mobile devices, but the more popular ones are called Square.

Since “Square” became so popular, other companies have released their versions of readers, such as Gopay, which is done by QuickBooks. But, there is also PayPal, Amazon Pay, Local Register and there's like many more.

The downside is that some of these are hardware-based. Often these devices don’t know what app it's talking to. Therefore, during a purchase, a malicious merchant or third party can record several extra encrypted swipes of a credit card.

Provided that the swipes don't get sent to the Square servers, they can then play these back into the Square Register app at a later time to modify the transaction.

In summary, we looked at mobile payments and talked about the difference between secured elements versus host card emulation. Both of them have their advantages and disadvantages. Apple has invested in the secure element, but they could quickly move over to host card emulation, or Google could move

over to secure emulation. It just depends on which one's going to end up being better for everybody all around.

Due to rapid changes in technology, in the upcoming years, we might see better payment options with even better security, but time will tell for sure.

Conclusion

I hope this book was able to get you started on your pursuit of becoming an Elite hacker and hopefully you will choose to become an Ethical Hacker.

In case you found some of the techniques and strategies I have demonstrated being advanced at first, it's ok, however repetition and on-going practice will help you to become an IT Professional in no time.

If you wish to check out my previous books, feel free to look up:

- *Volume 1 - Hacking – beginners guide*
- *Volume 2 - 17 Must Tools Every Hacker Should Have*
- *Volume 3 - Wireless Hacking*
- *Volume 4 - 17 Most dangerous hacking attacks*
- *Volume 5 - 10 MOST DANGEROUS CYBER GANGS*
- *HACKING WITH KALI LINUX: Penetration Testing Hacking Bible*
- *HACKING: Social Engineering Attacks, Techniques & Prevention*
- *HACKING: Hacking Firewalls & Bypassing Honeypots*
- *HACKING: Denial of Service Attacks*
- *HACKING: How to Hack Web Apps*

…or better yet, you can check out my 3 Books and 5 Books Bundle by visiting my Author Page on Amazon.

Thanks again for purchasing this book.

Lastly, if you enjoyed the content, please take some time to share your thoughts and post a review. It'd be highly appreciated!

About the Author

Alex, originally from Germany, currently living in the UK. Alex is a Network & Security Engineer, Ethical Hacker having over 10 years of experience within the IT field.

Alex already written 8 books, all revolving around Ethical Hacking, Cybersecurity and various Linux Tutorials.

Occasionally filing contract roles and helping large Companies to identify Security Vulnerabilities, in European Countries as well Honk Kong, Singapore and sometimes in the US.

From Alex:

"I do my best to keep myself up to date with Technology and innovation of Hacking methods by studying every day, but I also would like to share my experience and knowledge with the world, especially those are new to IT Security, and Ethical Hacking.

My intention is to teach you what I have learned for years by explaining Hacking methods I have experienced and the reason why these activities has become and formed such a large Cyber Criminal Organizations.

I will be providing Hacking strategies and step-by-step implementations in order to provide you with some homework by building your home lab and start to practice how so you can start your career and become an Ethical Hacker."