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.

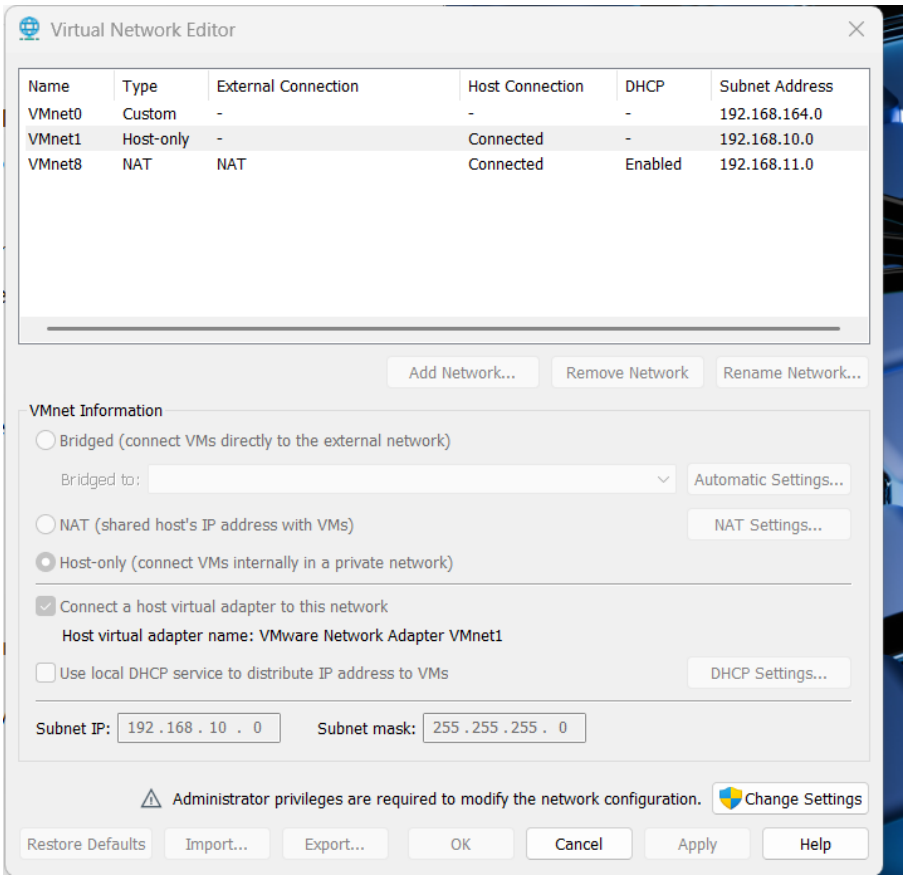
4. VM Set Up and Network Configuration

4.1 Virtual Machine Setup

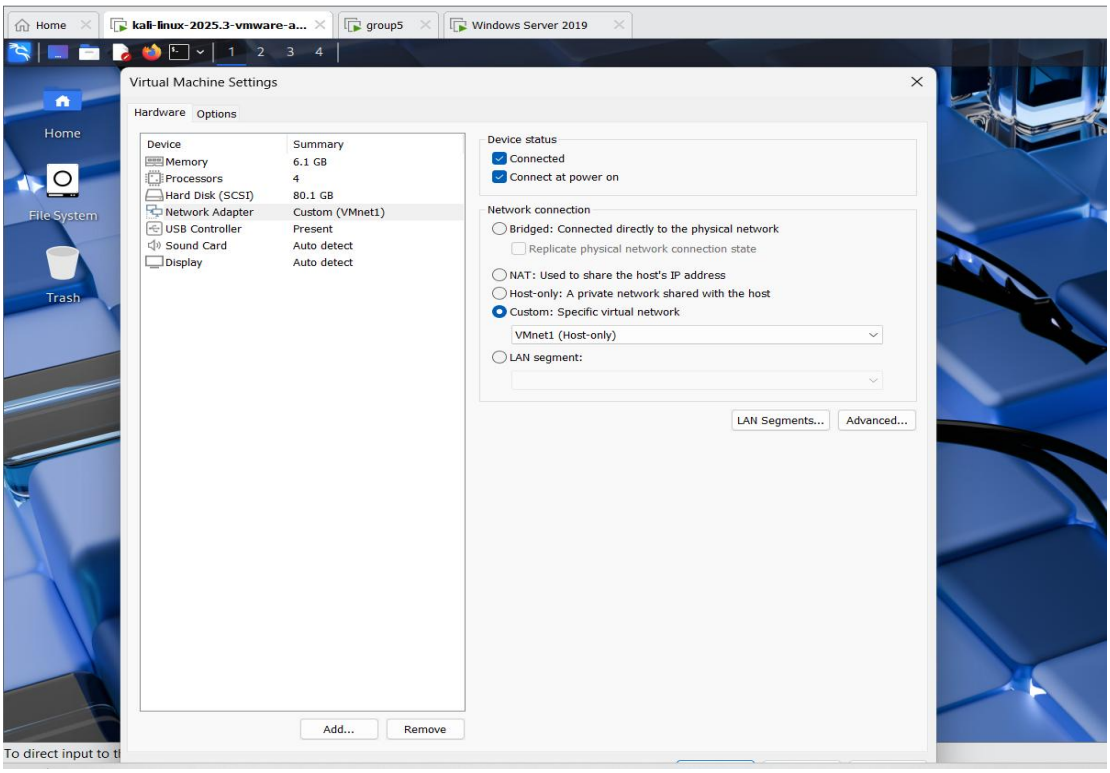
We created three virtual machines to simulate a corporate environment:

- **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.

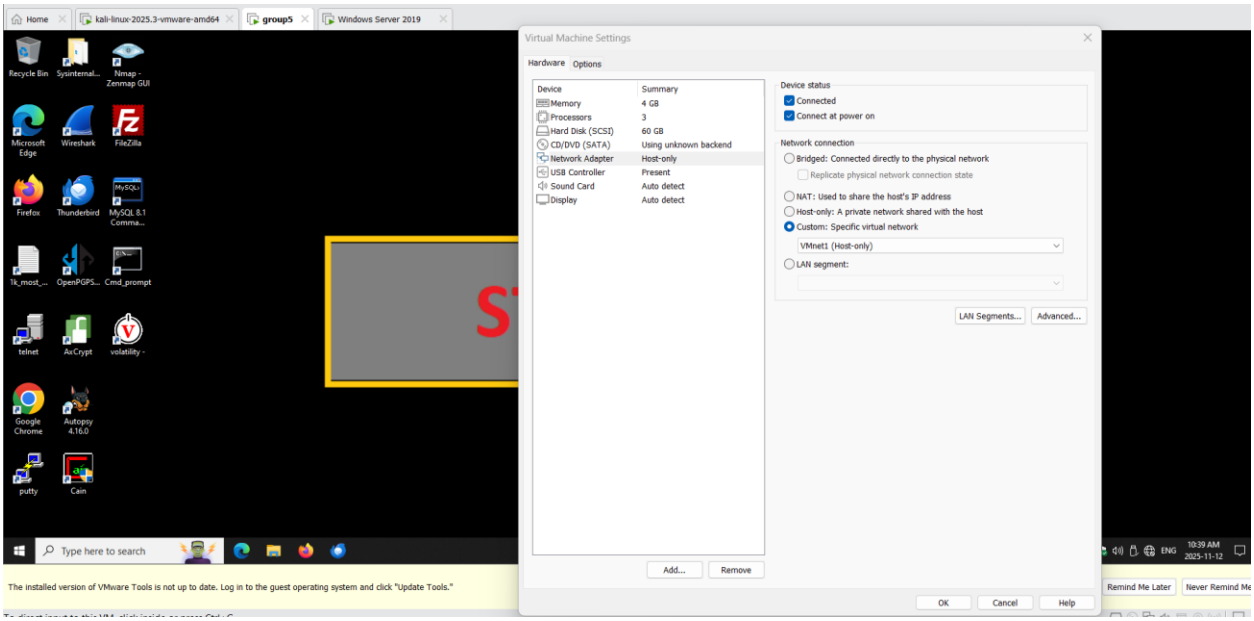
In the VM screenshots, we show the VM names, assigned resources (RAM/CPU), and the virtual network adapter bound to the internal network used for our lab.


Virtual Network Editor
Name Type External Connection Host Connection DHCP Subnet Address
VMnet0 Custom - - - 192.168.164.0
VMnet1 Host-only - Connected - 192.168.10.0
VMnet8 NAT NAT Connected Enabled 192.168.11.0
Add Network...
Remove Network
Rename Network...
VMnet Information
Bridged (connect VMs directly to the external network)
Bridged to:
Automatic Settings...
NAT (shared host's IP address with VMs)
NAT Settings...
Host-only (connect VMs internally in a private network)
Connect a host virtual adapter to this network
Host virtual adapter name: VMware Network Adapter VMnet1
Use local DHCP service to distribute IP address to VMs
DHCP Settings...
Subnet IP: 192 . 168 . 10 . 0
Subnet mask: 255 . 255 . 255 . 0
Administrator privileges are required to modify the network configuration.
Change Settings
Restore Defaults
Import...
Export...
OK
Cancel
Apply
Help

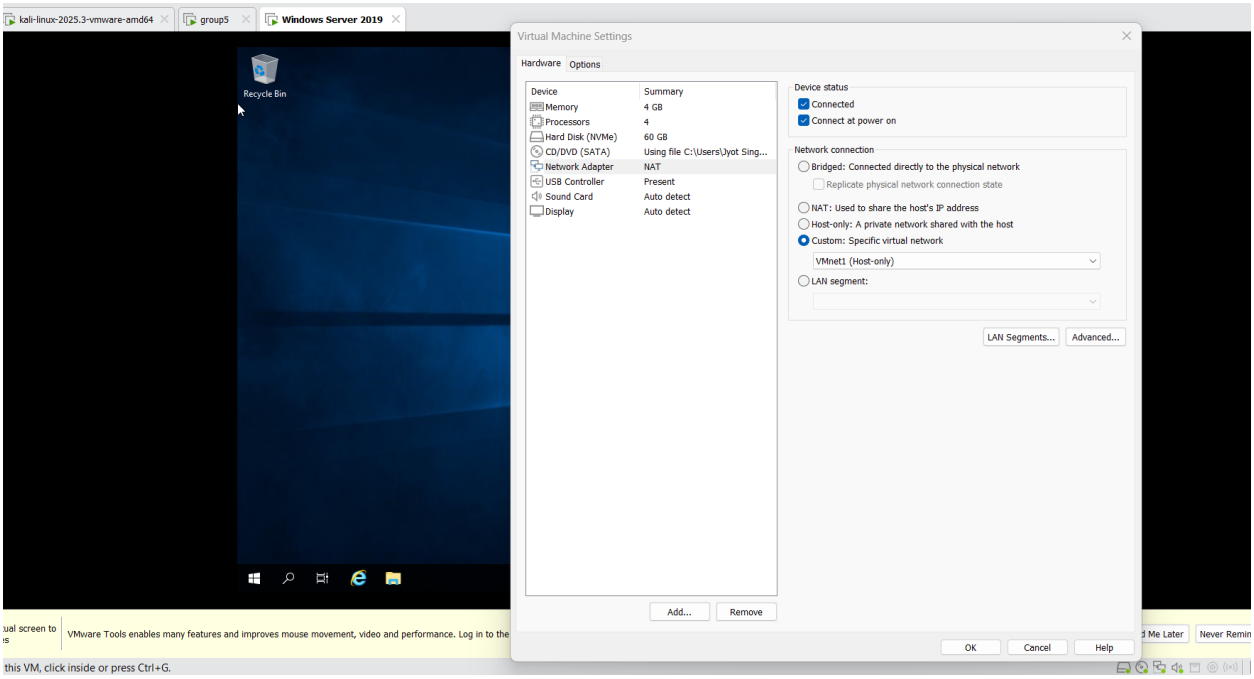
This screenshot shows us that kali linux VM is connected with with the VMnet1 (Host -only) network.



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 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`

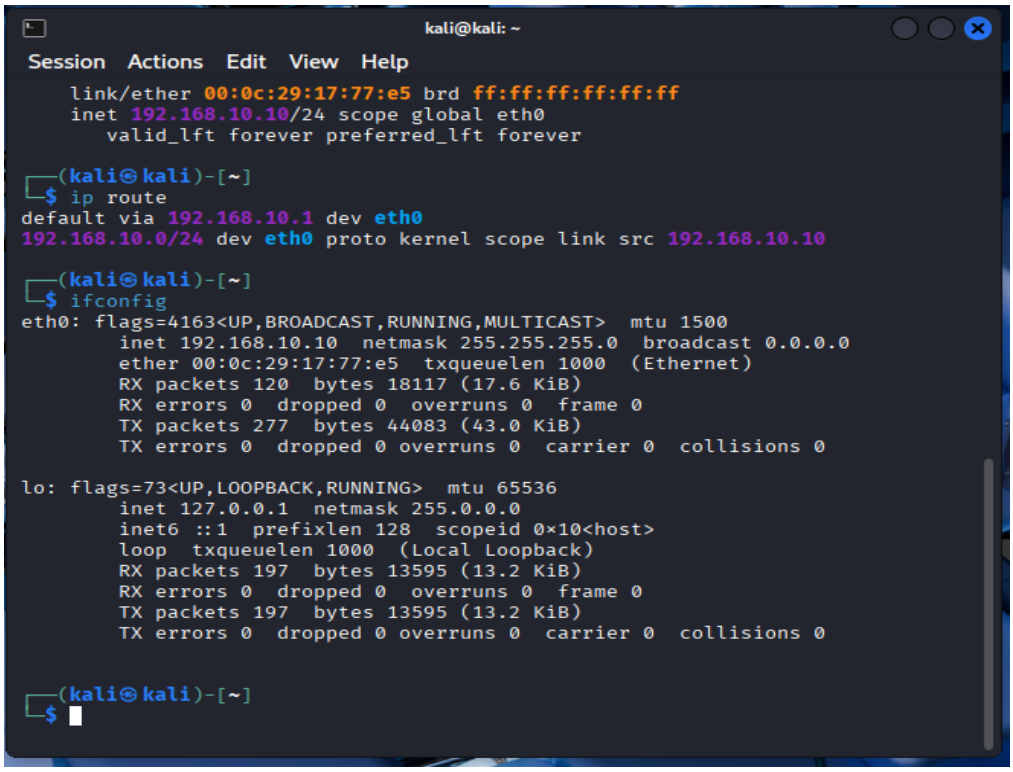
The screenshots highlight:

- 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).

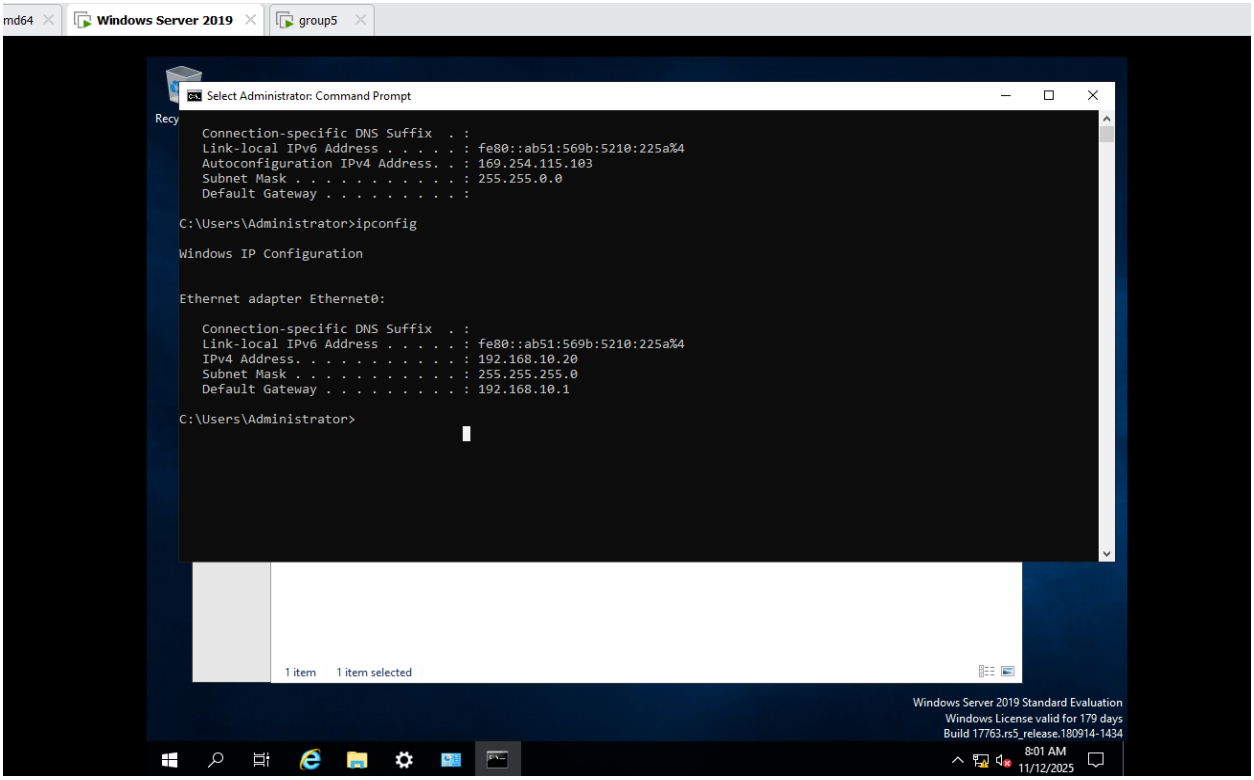
This configuration ensured that:

- 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.

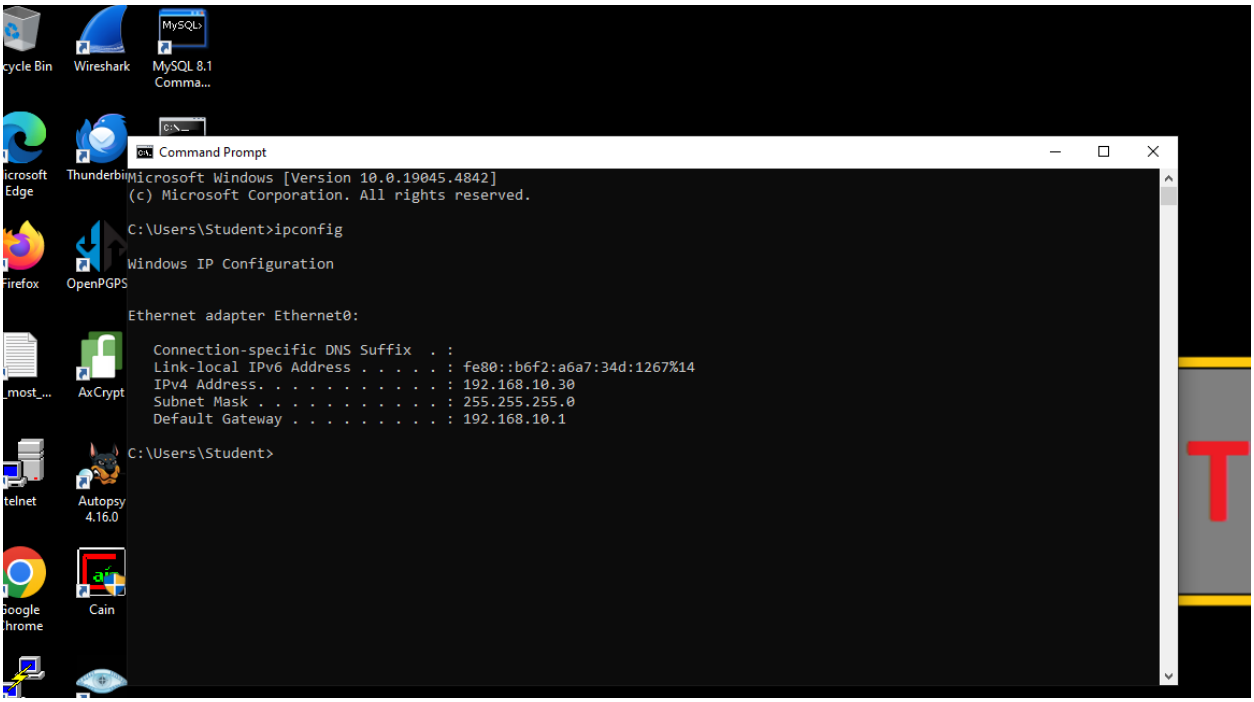
This screenshot provides us the information of IP address and subnet mask of Kali Linux.



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



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:

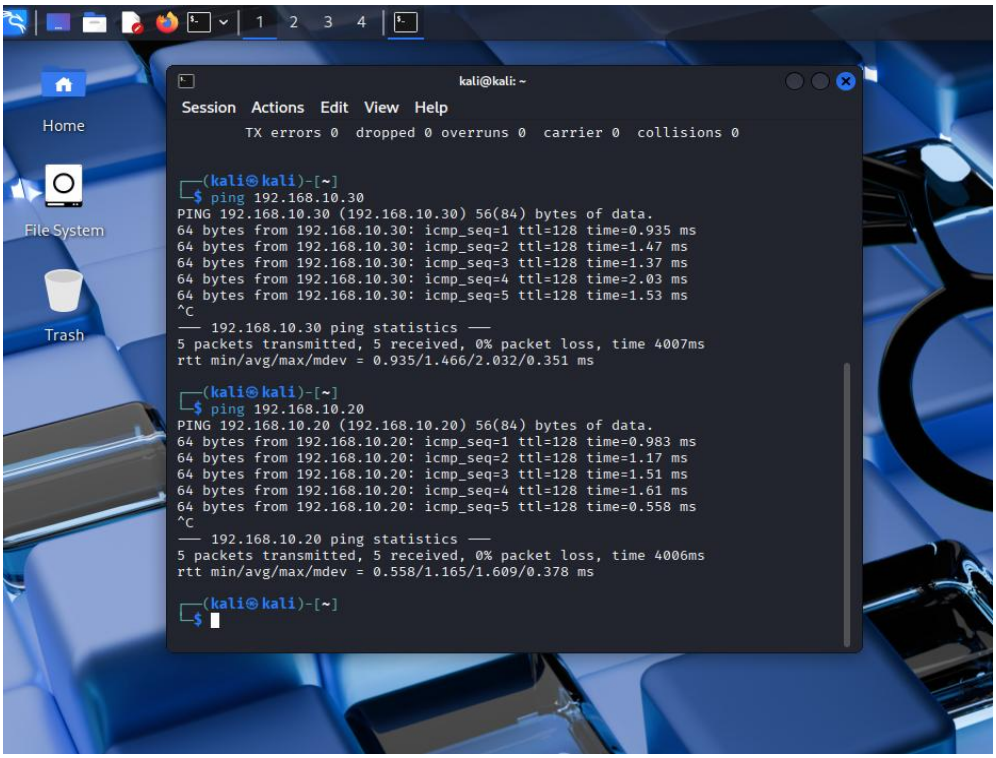
- From **Kali**, we pinged `192.168.10.20` (Windows Server) and `192.168.10.30` (Windows 10 client).
- From **Windows Server**, we pinged the Kali machine and the client.
- From **Windows 10**, we pinged the server and Kali.

The screenshots show successful ICMP replies, confirming:

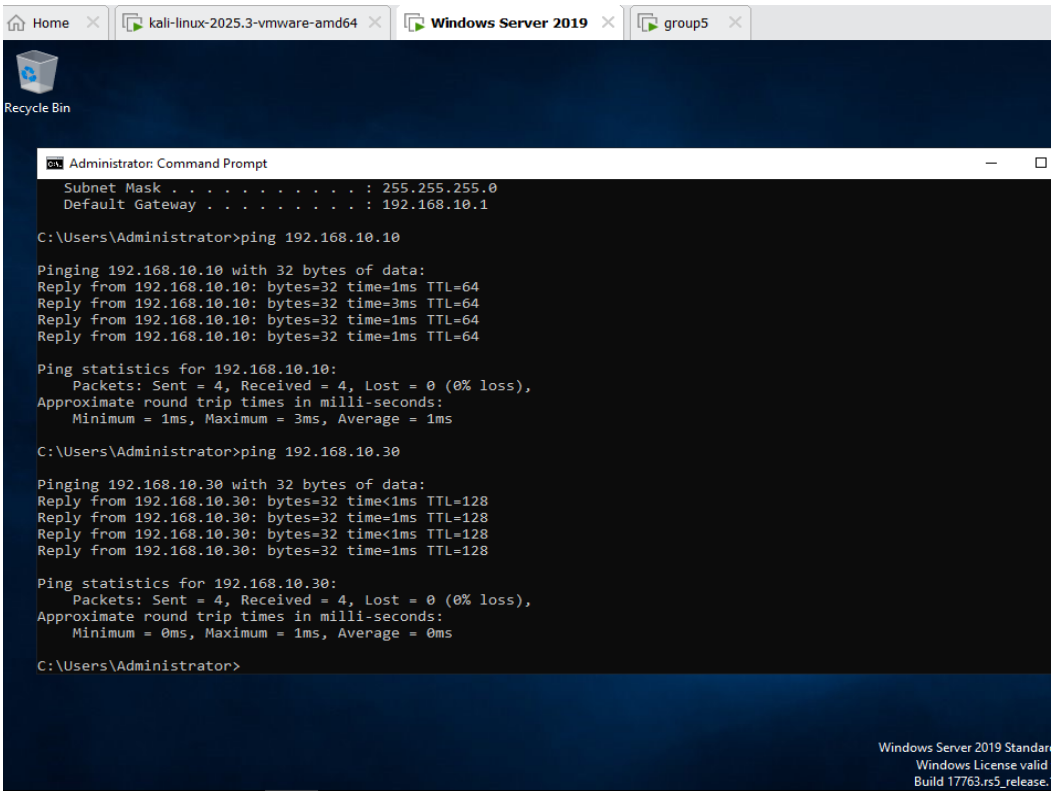
- Each system was correctly configured on the same subnet.
- The internal lab network was operational and ready for Active Directory and further testing.

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

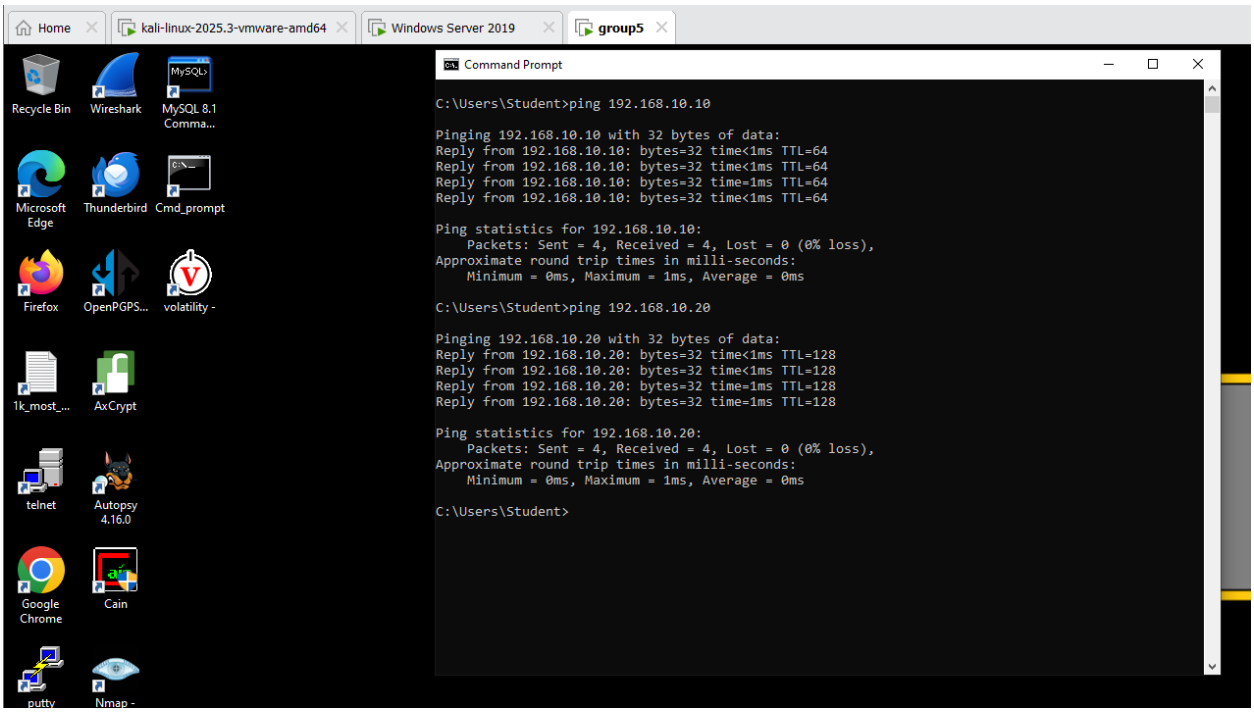
This is the proof of connectivity of Kali Linux with other VMs.



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

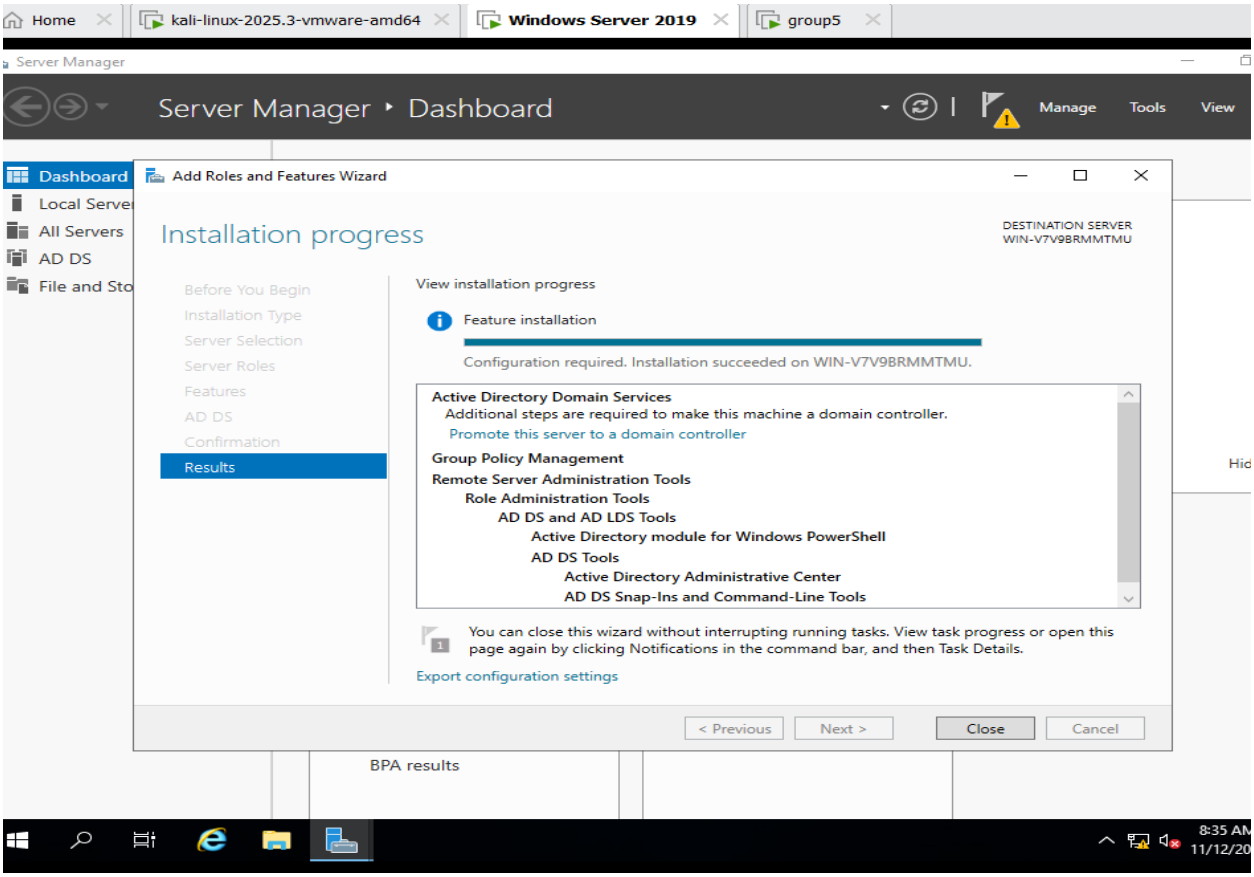
The screenshots illustrate:

- 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.

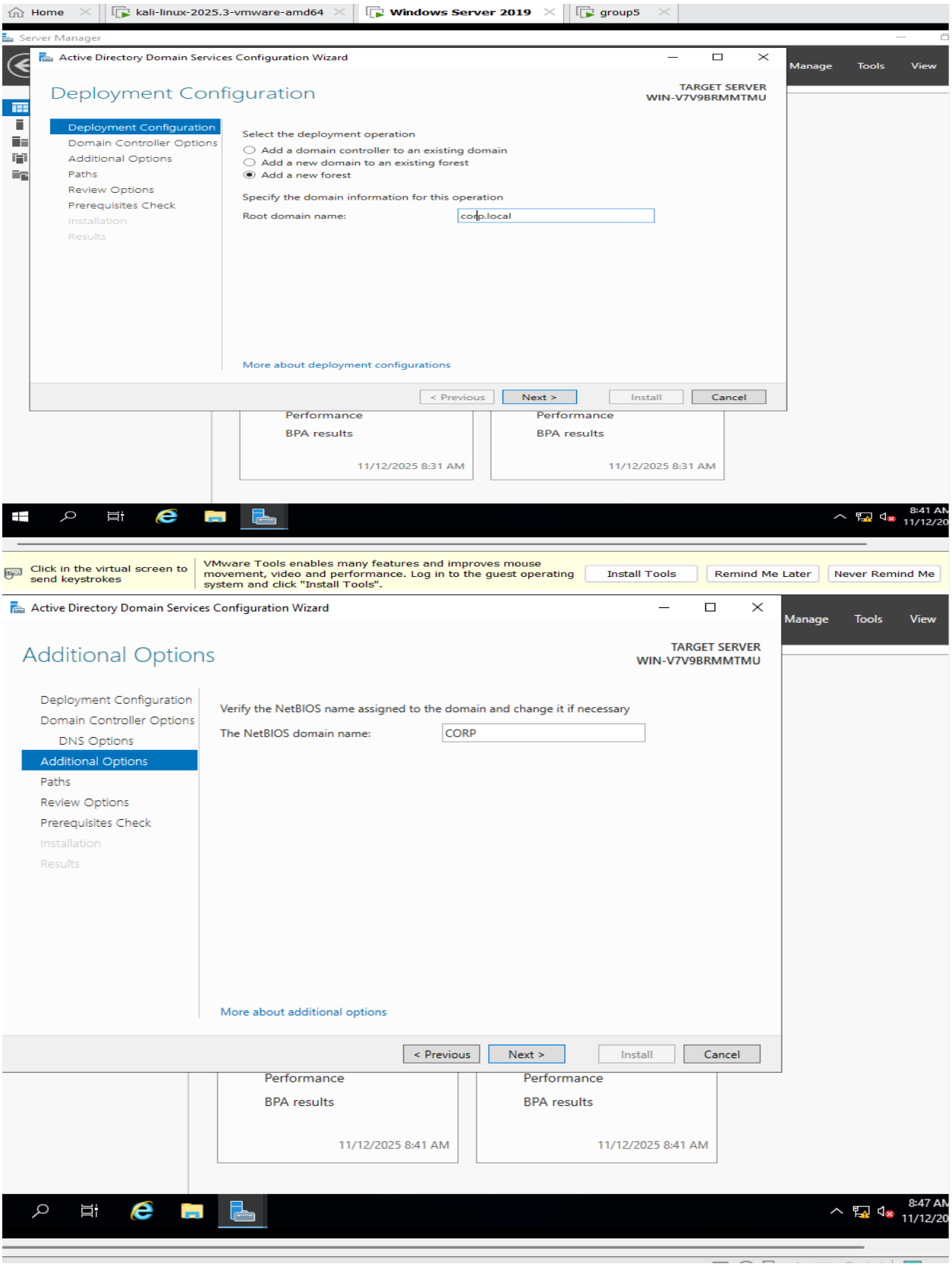
As part of the configuration, we:

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

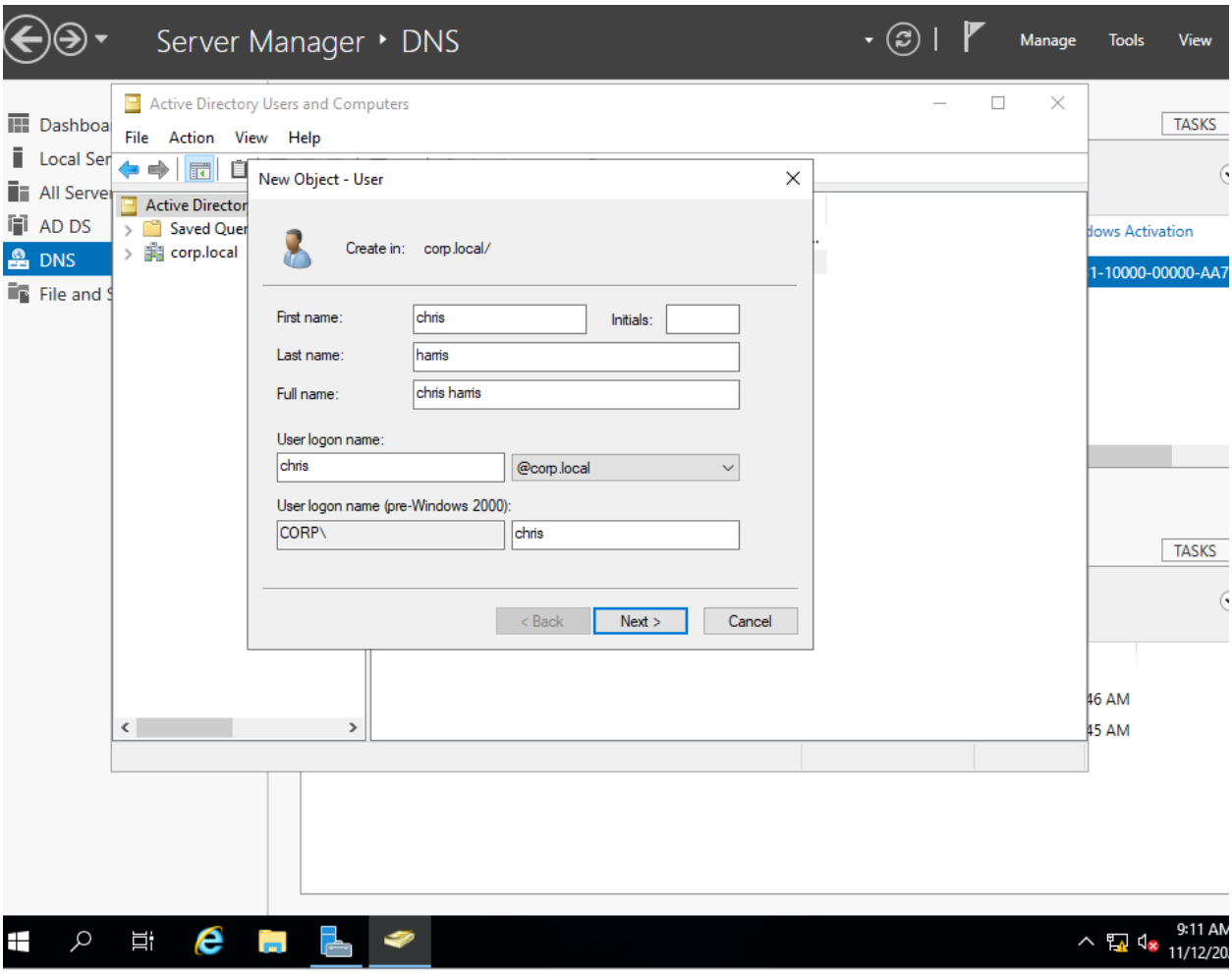
The below screenshots provide us the steps how to create active directory in windows server.



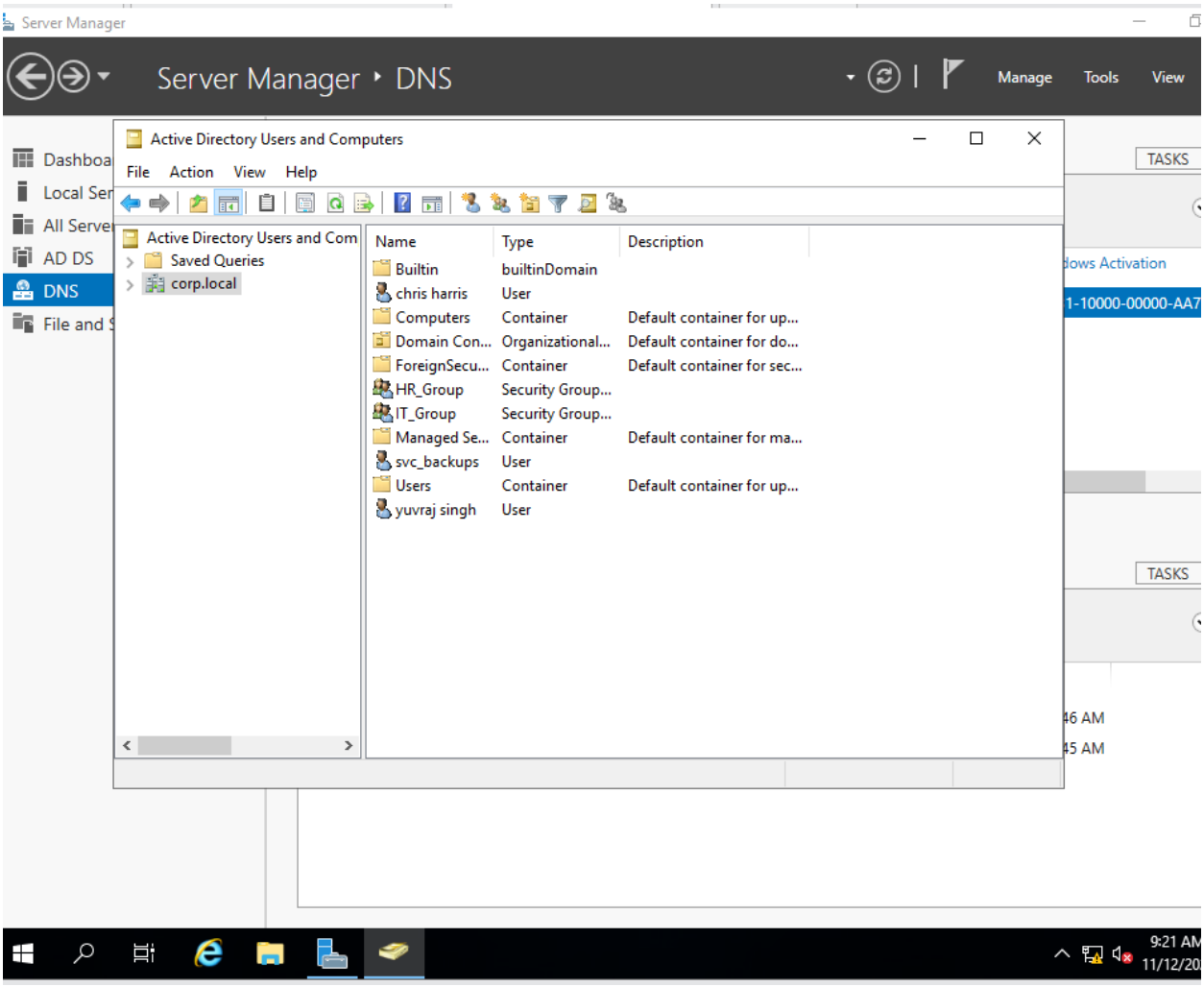
This is the step to set a domain name.



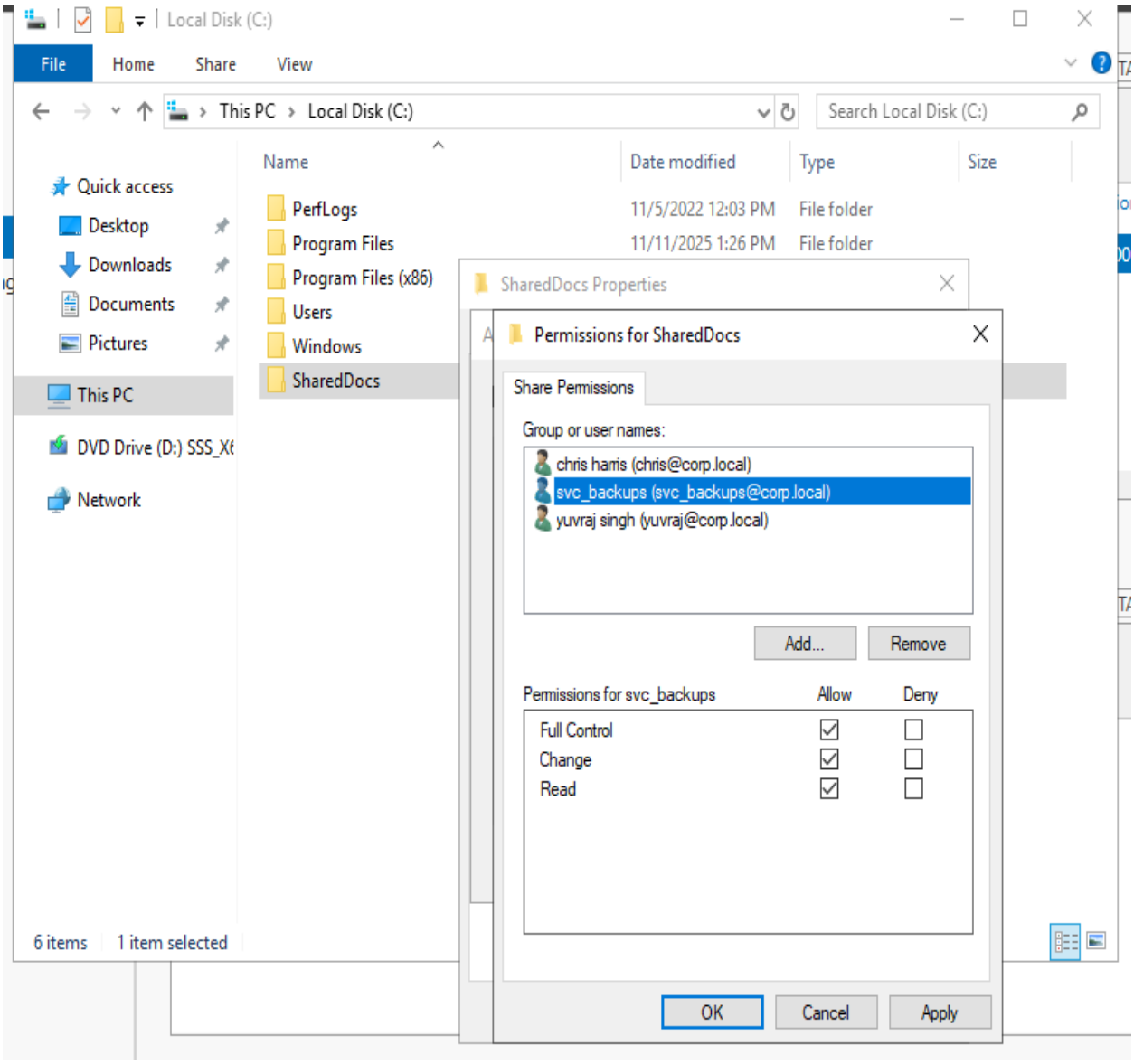
This is how we create different users in active directory domain.



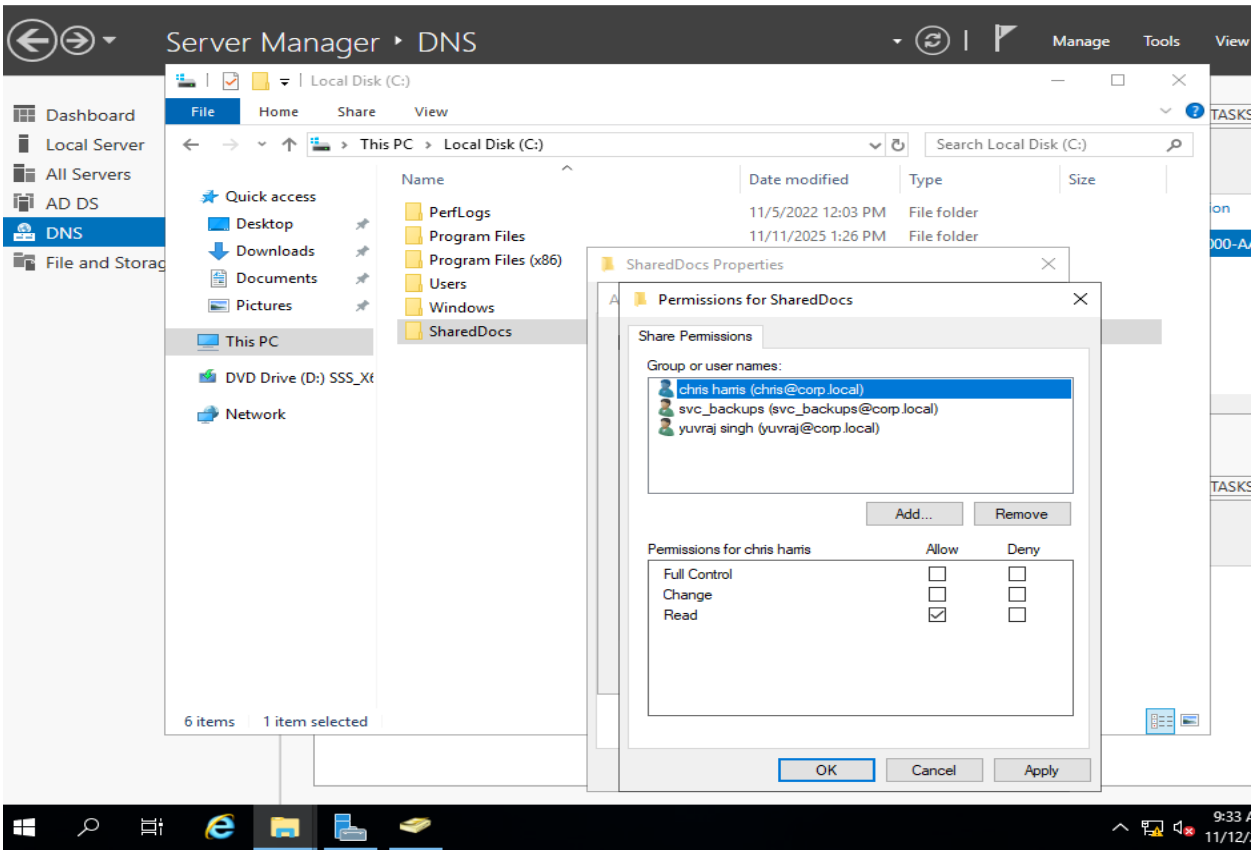
This screenshot proof of differnet users and groups in active directory.



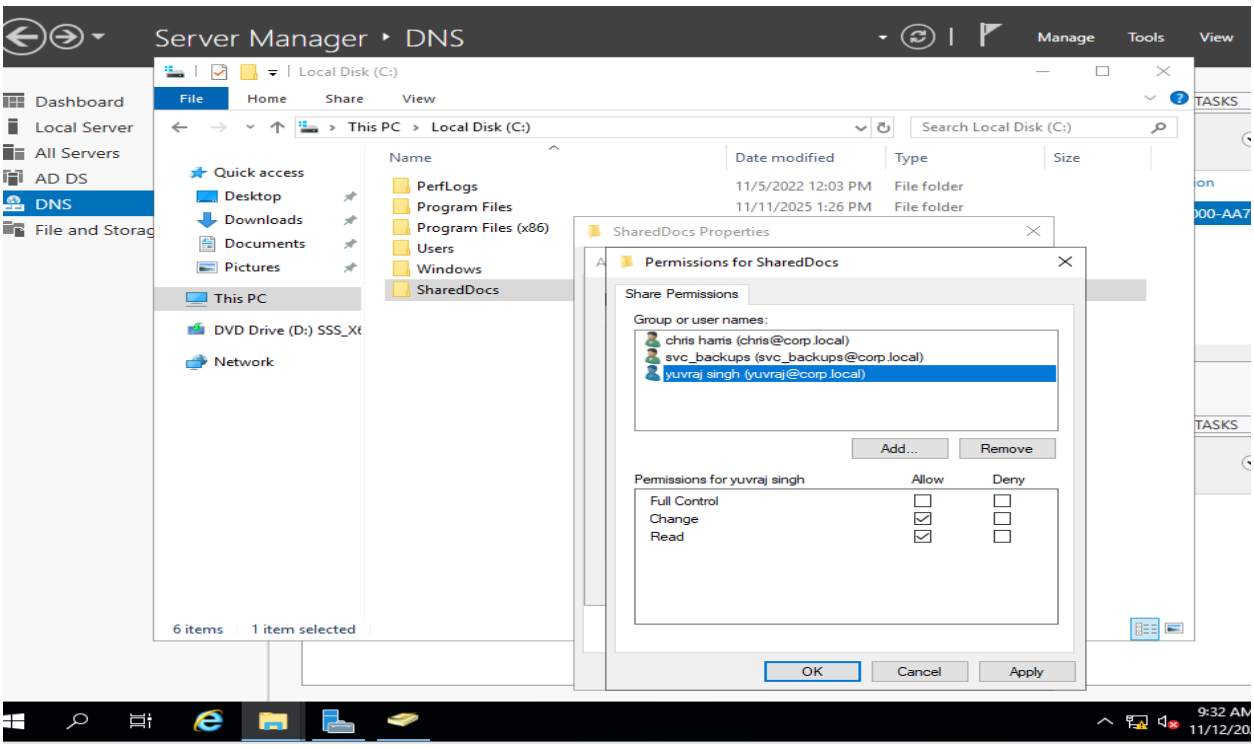
This screenshot provides the information that this user has full control on permissions.



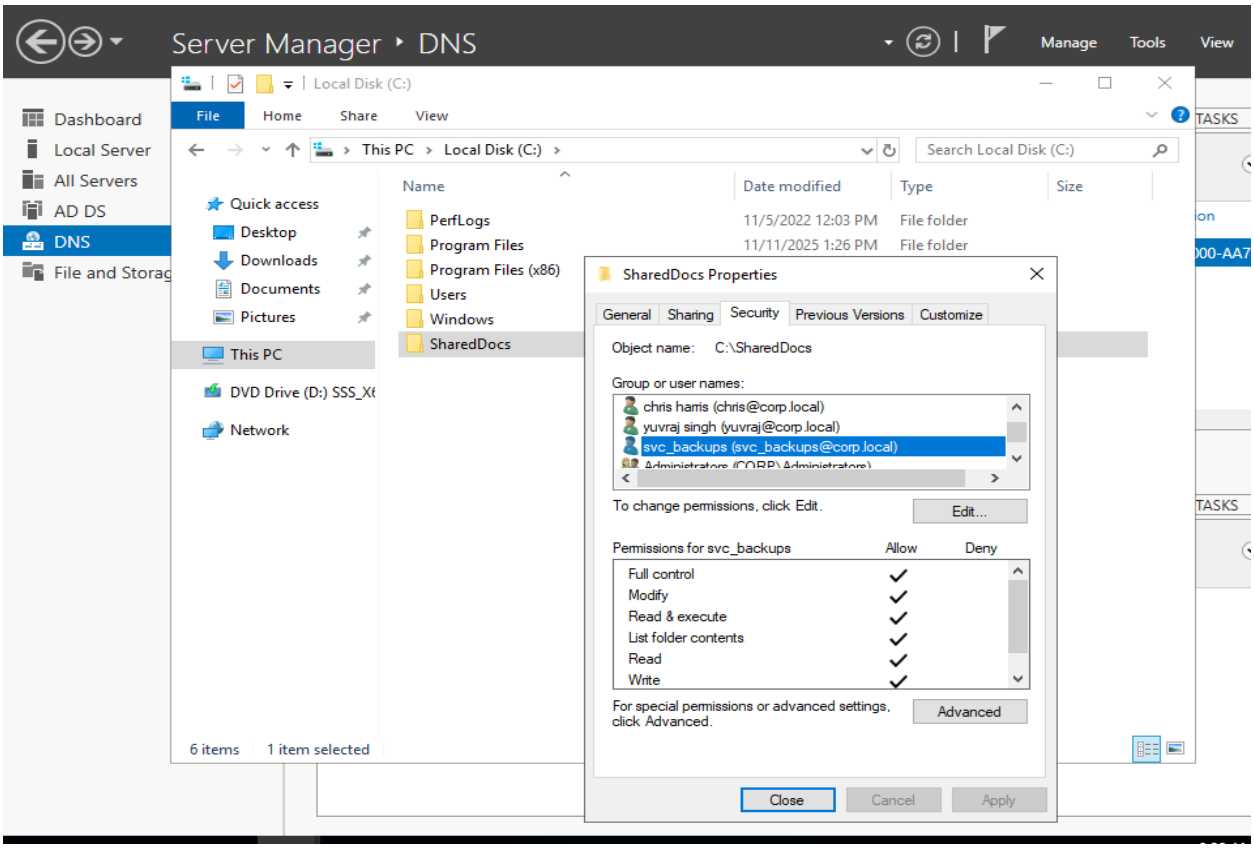
This user does not have full control on permissions. It has ability to read and change in files.



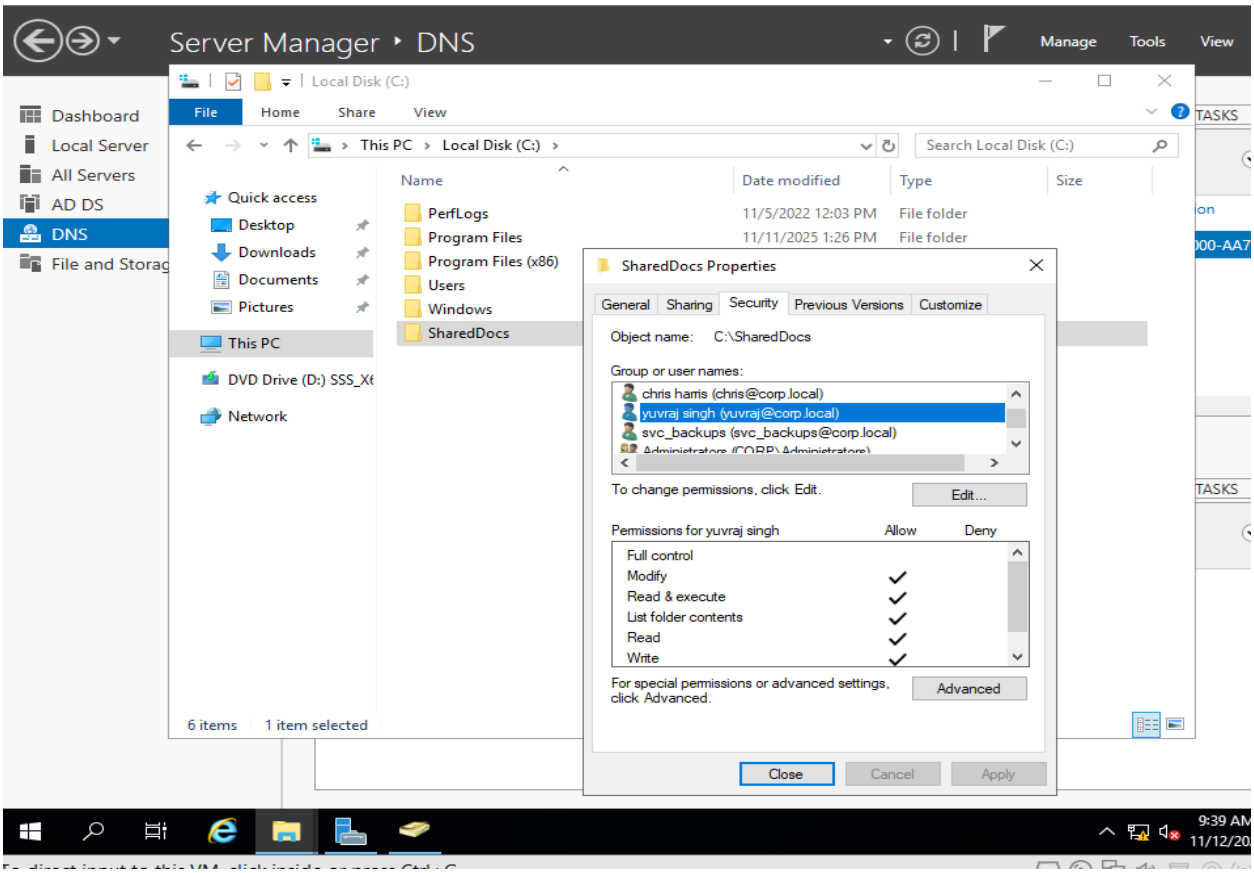
This user has only permission to read the files.



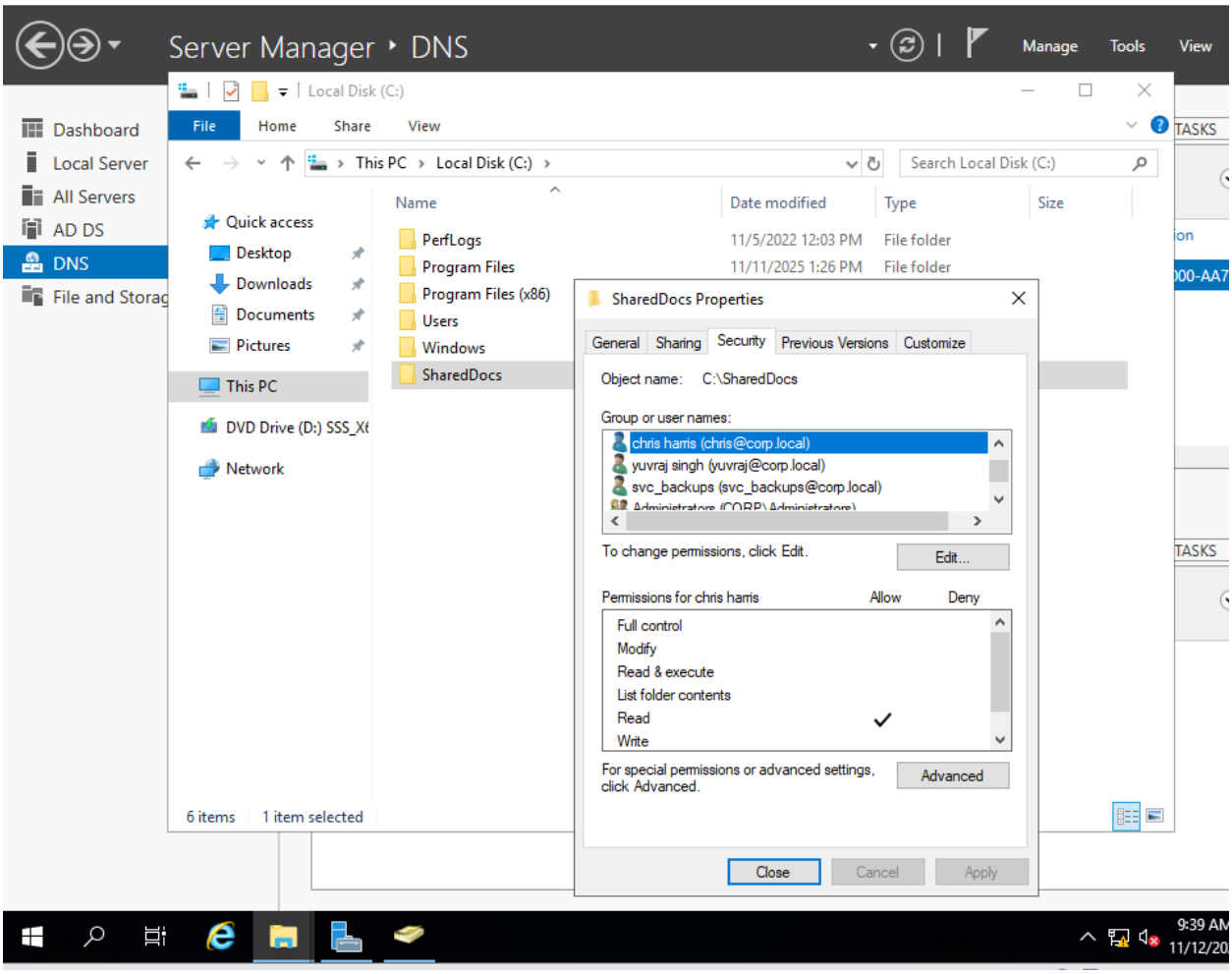
This screenshot shows that this user has full control, it can read, write, modify and execute the files.



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

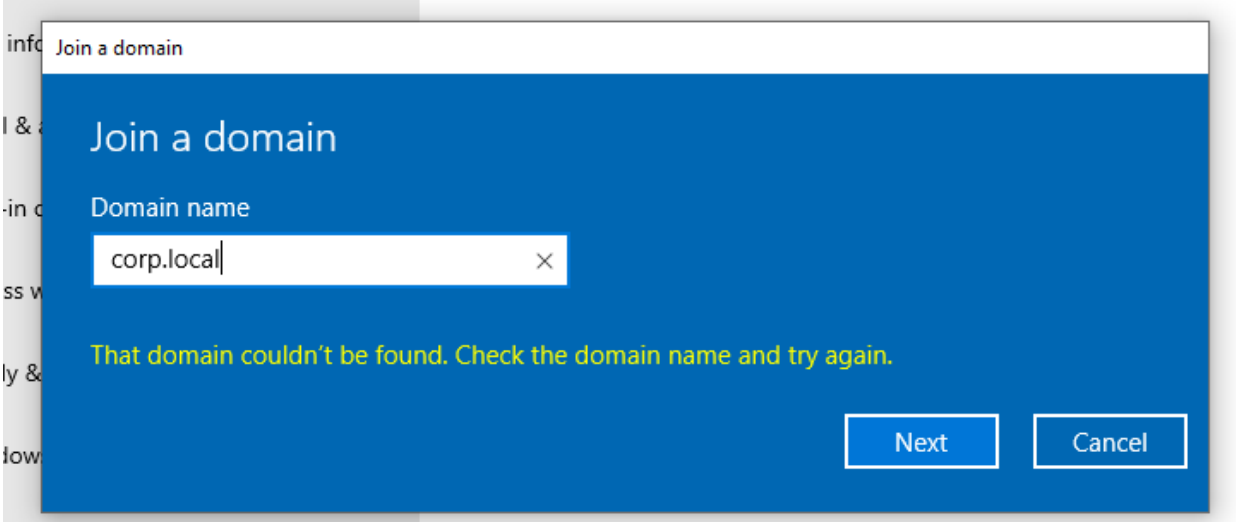
Next, we joined the **Windows 10 VM** to the newly created domain. The screenshots in this section show:

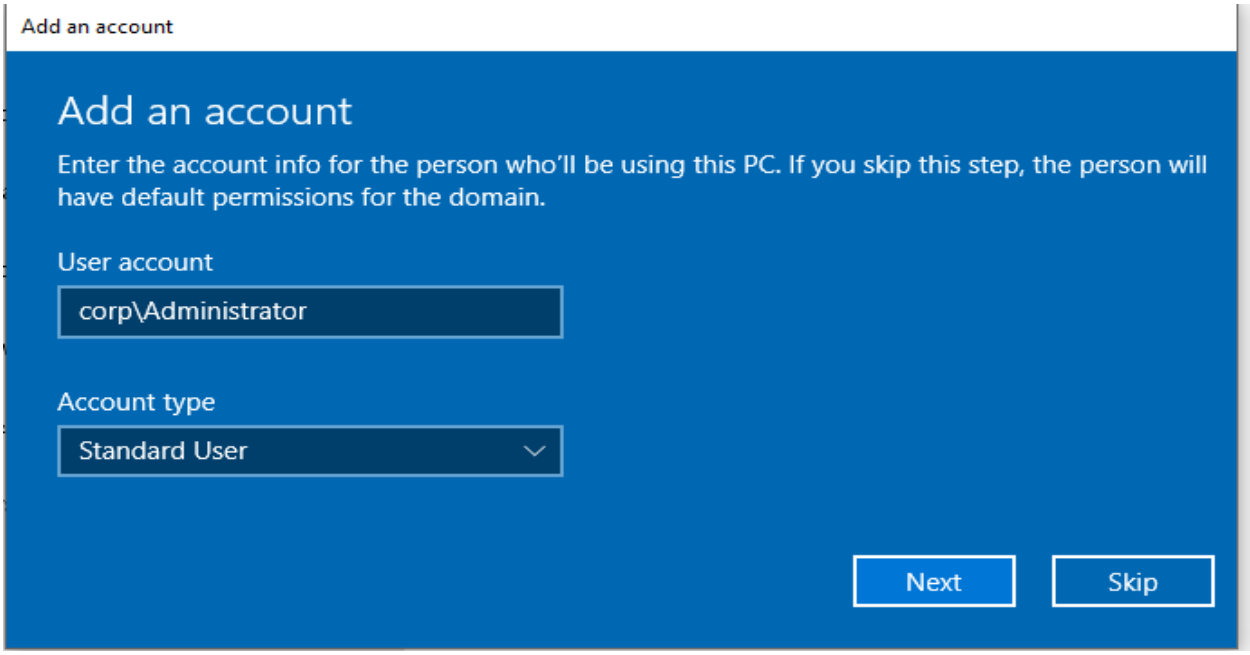
- The **System Properties** window on Windows 10 where we changed the workgroup to the domain name.
- The domain join dialog prompting for domain credentials.
- The confirmation message indicating that the computer successfully joined the domain and required a reboot.

This step is crucial because:

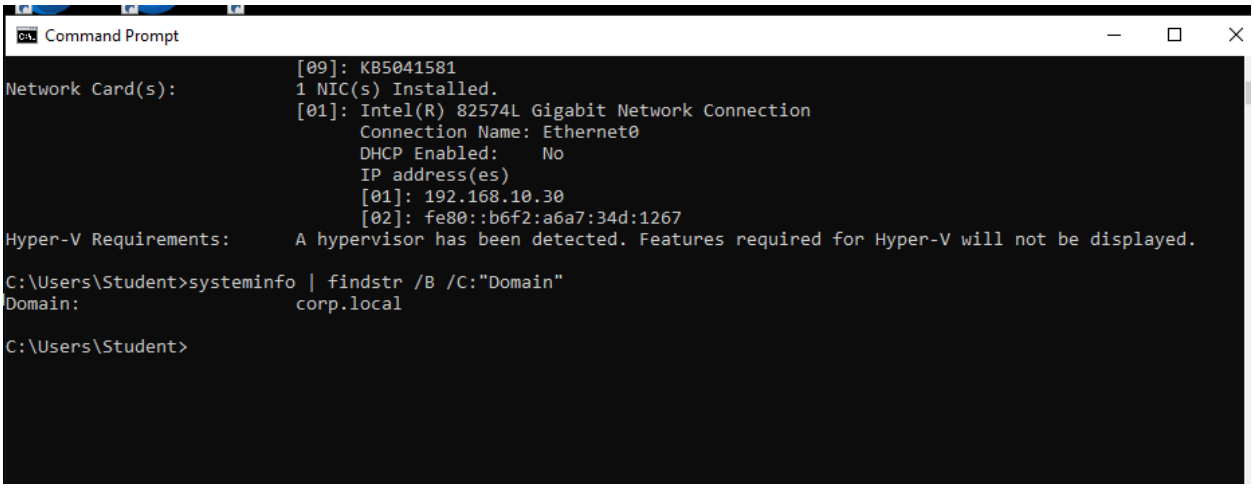
- It creates a realistic Active Directory environment that attackers frequently target.
- It enables domain authentication, group policies, and SMB shares that can be discovered and abused during enumeration and exploitation.

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



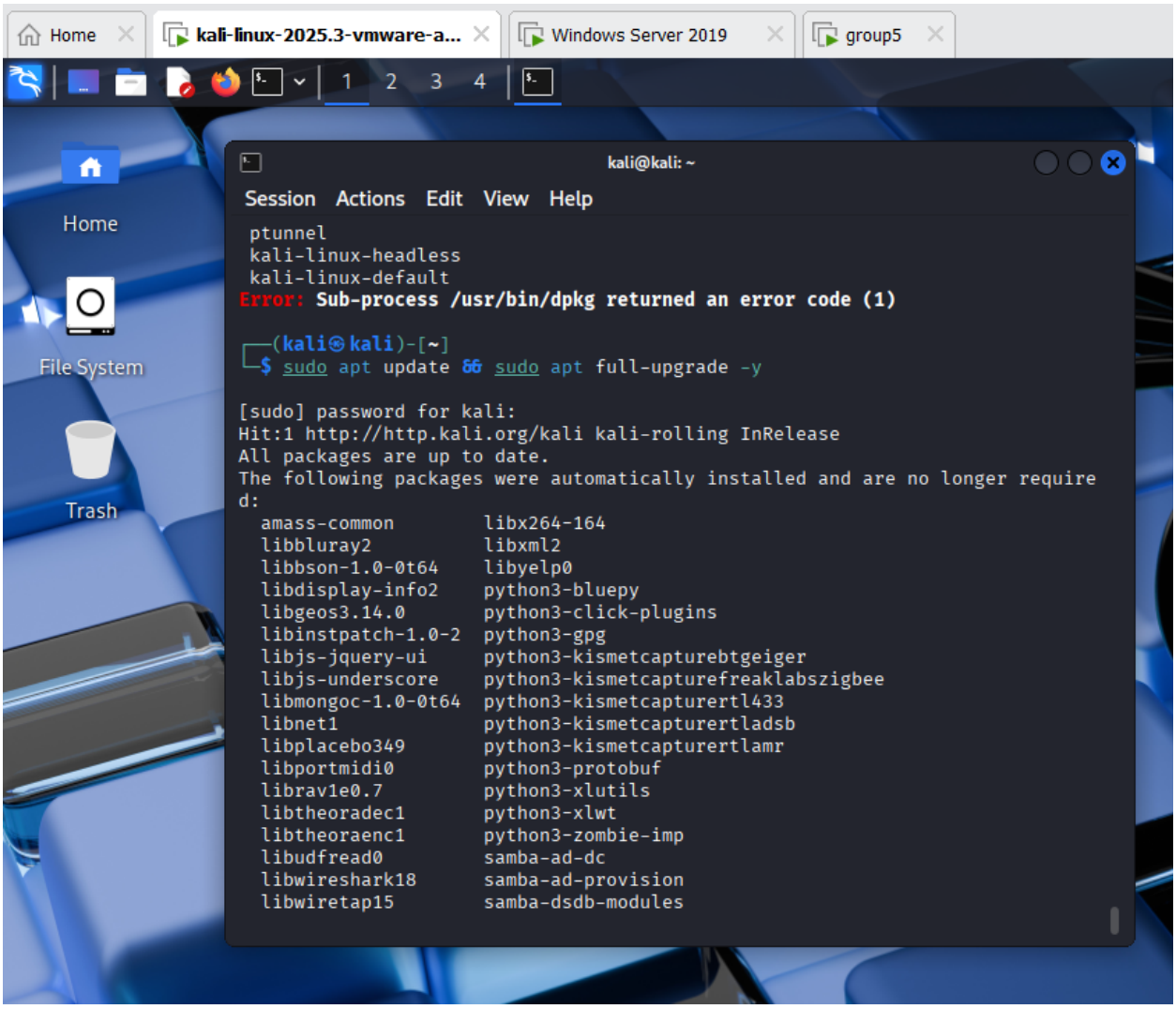


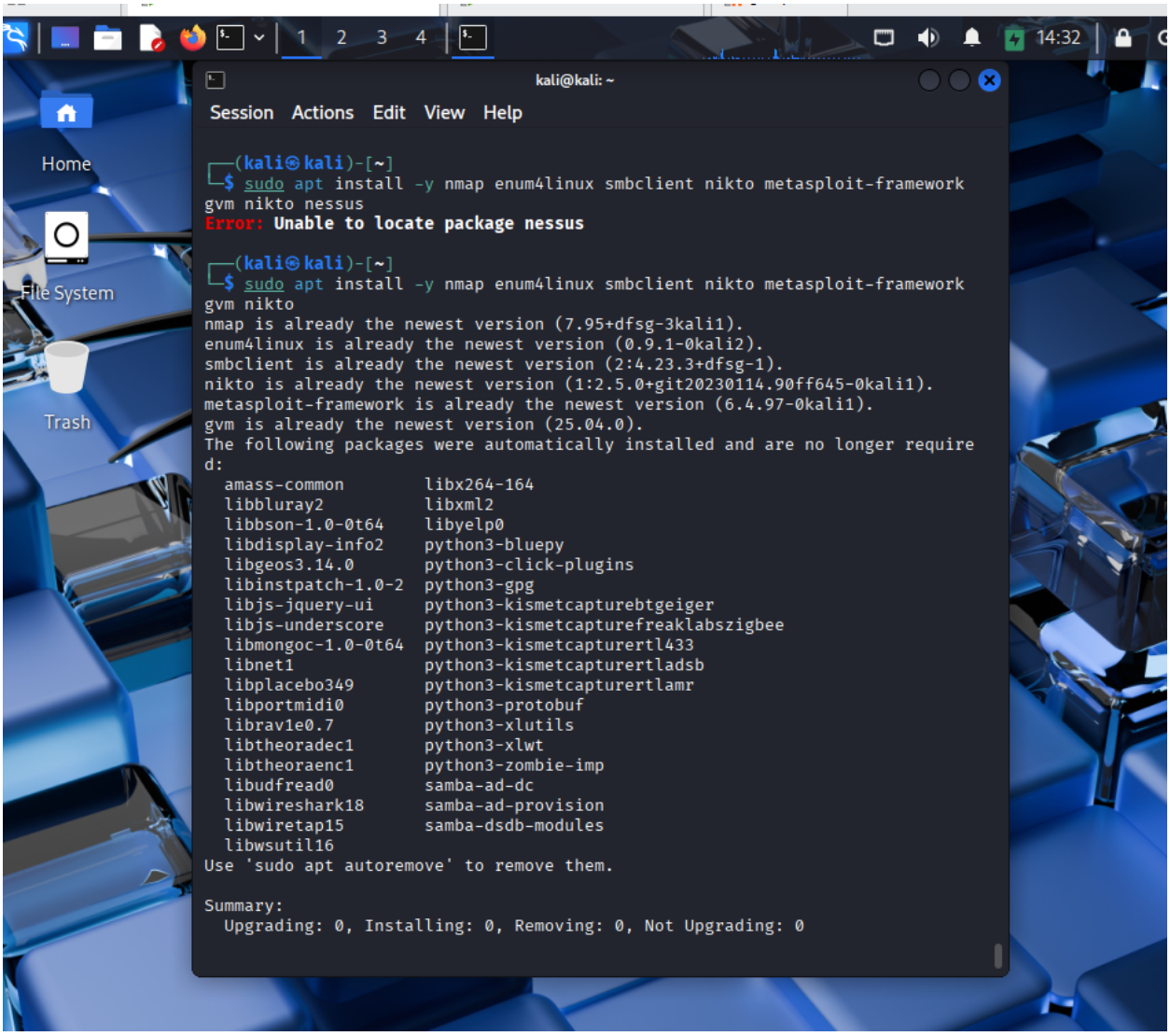
This is the proof of successfully connected client's machine with domain.



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.


Home
kali-linux-2025.3-vmware-a...
Windows Server 2019
group5
1 2 3 4
Home
File System
Trash
kali@kali: ~
Session Actions Edit View Help
ptunnel
kali-linux-headless
kali-linux-default
Error: Sub-process /usr/bin/dpkg returned an error code (1)
(kali㉿kali)-[~]
$ sudo apt update && sudo apt full-upgrade -y
[sudo] password for kali:
Hit:1 http://http.kali.org/kali kali-rolling InRelease
All packages are up to date.
The following packages were automatically installed and are no longer required:
amass-common
libbluray2
libbson-1.0-0t64
libdisplay-info2
libgeos3.14.0
libinstpatch-1.0-2
libjs-jquery-ui
libjs-underscore
libmongoc-1.0-0t64
libnet1
libplacebo349
libportmidi0
librav1e0.7
libtheoradec1
libtheoraenc1
libudfread0
libwireshark18
libwiretap15
libx264-164
libxml2
libyelp0
python3-bluepy
python3-click-plugins
python3-gpg
python3-kismetcapturebtgeiger
python3-kismetcapturefreaklabszigbee
python3-kismetcapturertl433
python3-kismetcapturertladsb
python3-kismetcapturertlamr
python3-protobuf
python3-xlutils
python3-xlwt
python3-zombie-imp
samba-ad-dc
samba-ad-provision
samba-dsdb-modules



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.

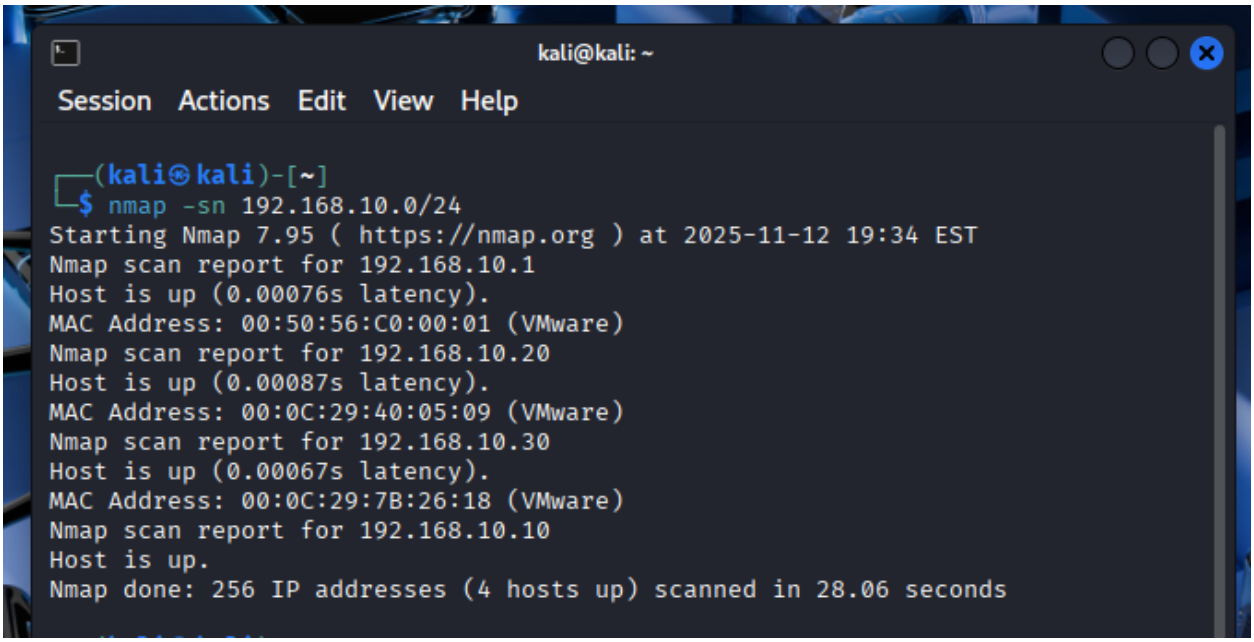
The screenshots show:

- 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),

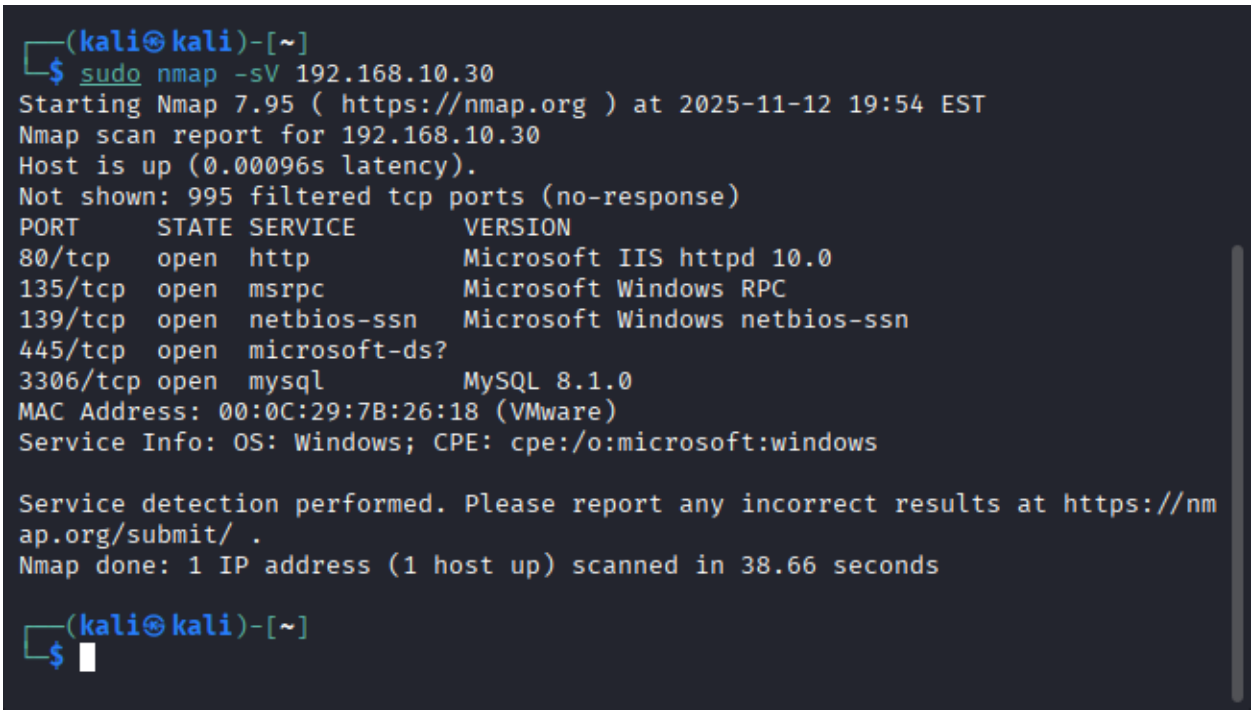
- 3389 (RDP) or others depending on enabled services.

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

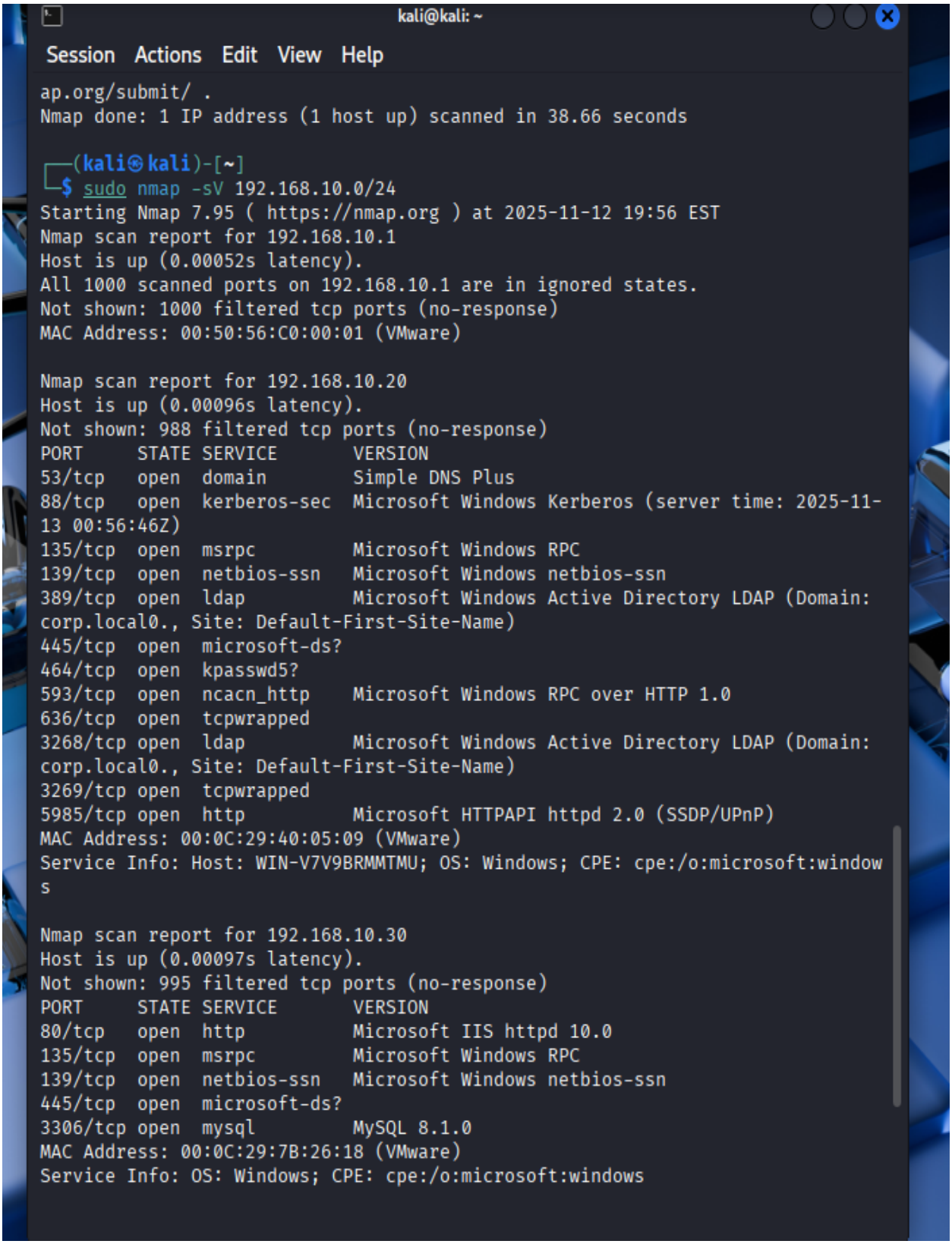
The purpose of this command to check which hosts are up on a network without scanning ports.



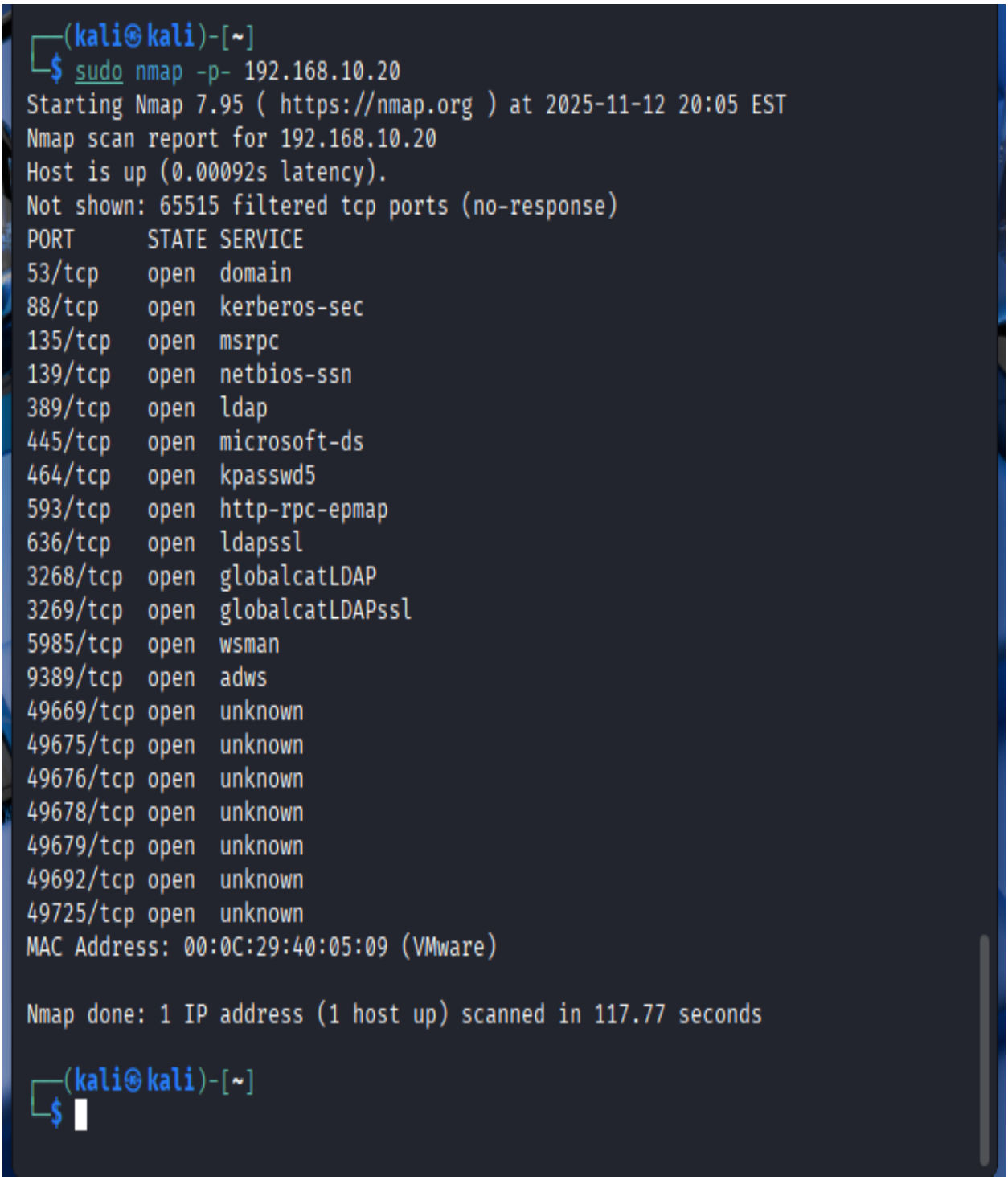
We use this command to identify services and their versions running on open ports.



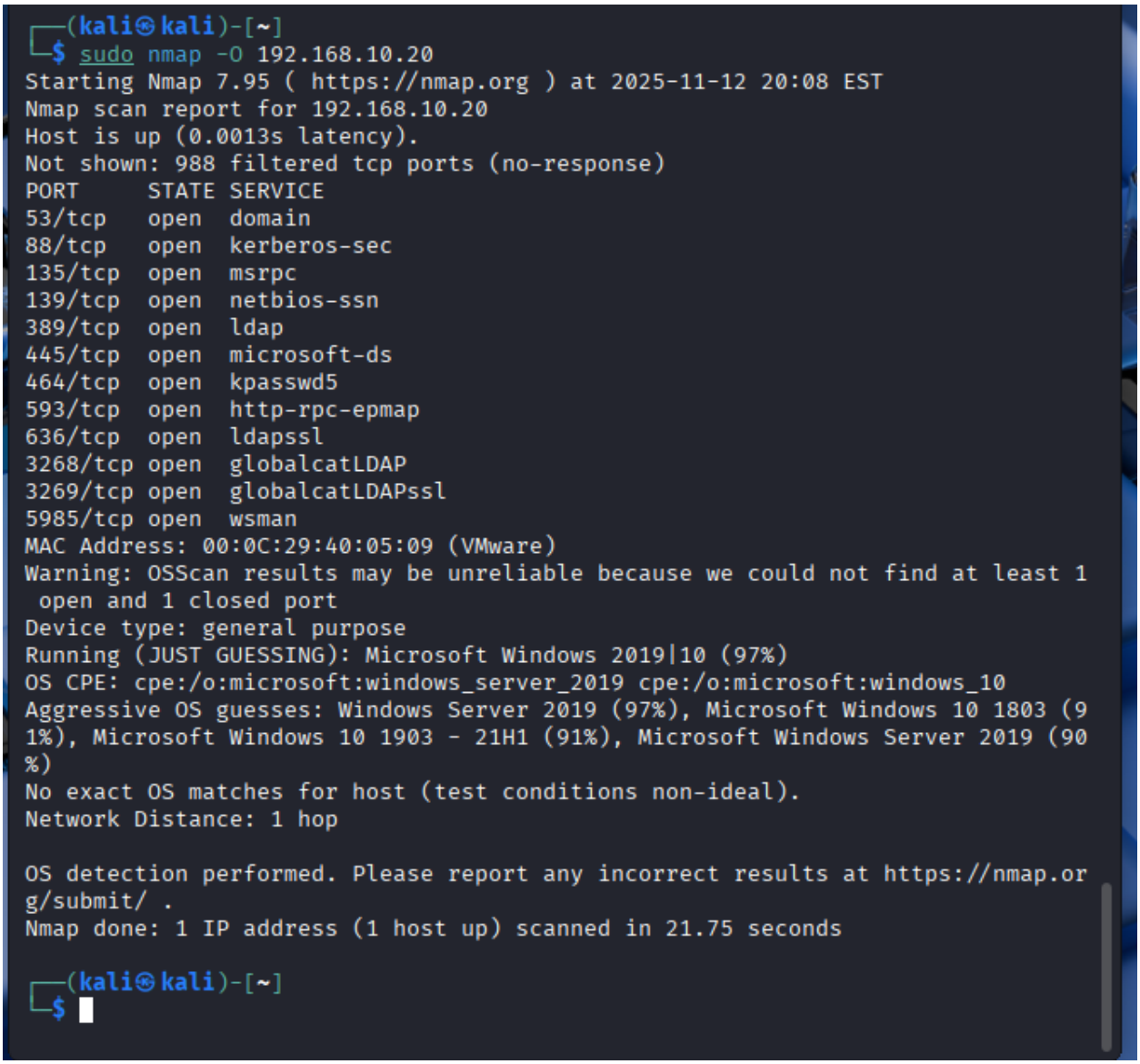
We use this command to identify services and their versions running on open ports without choosing any specific host.



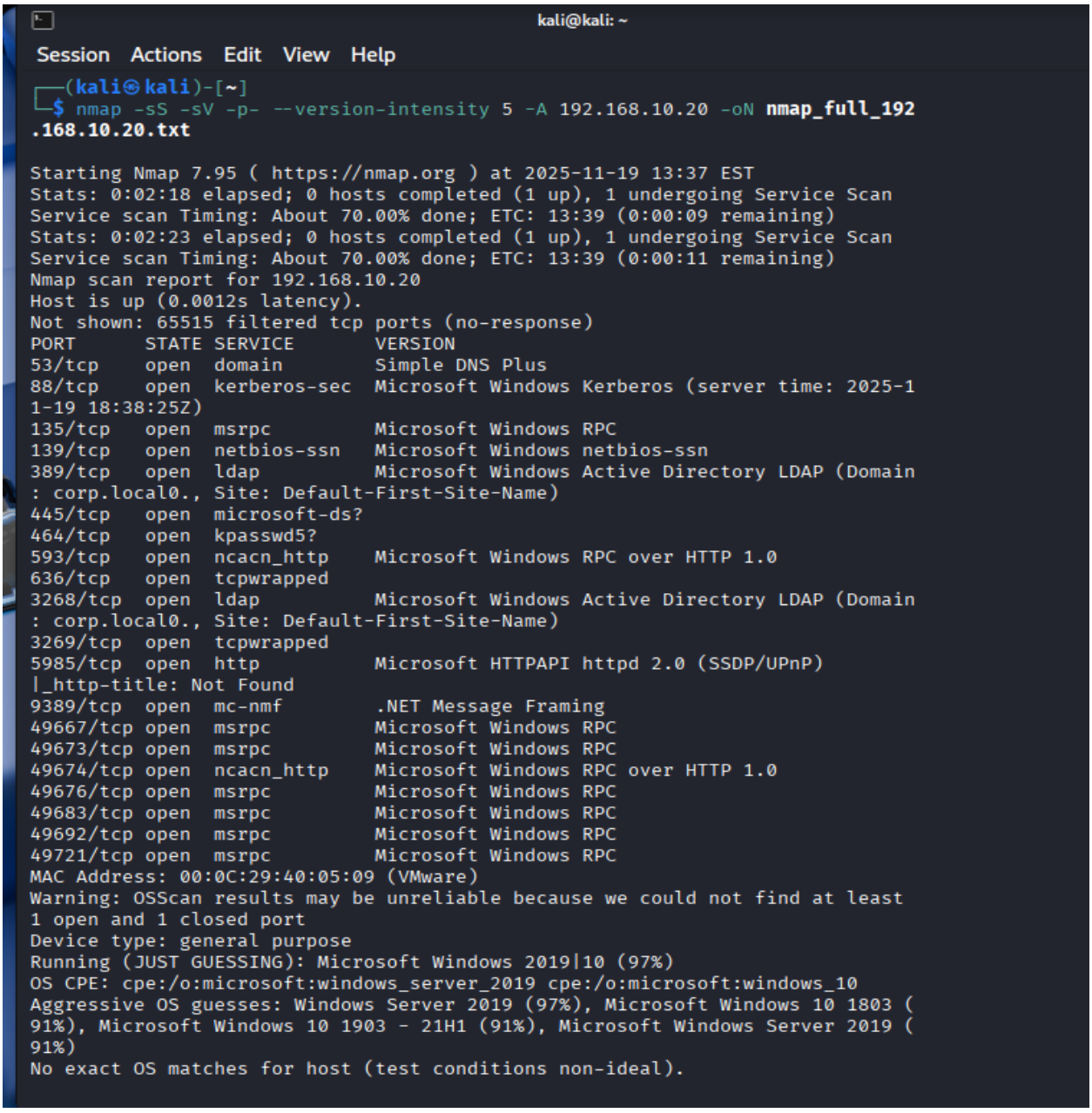
The purpose of this command is used to scan all 65,535 ports instead of just the default 1,000 ports.

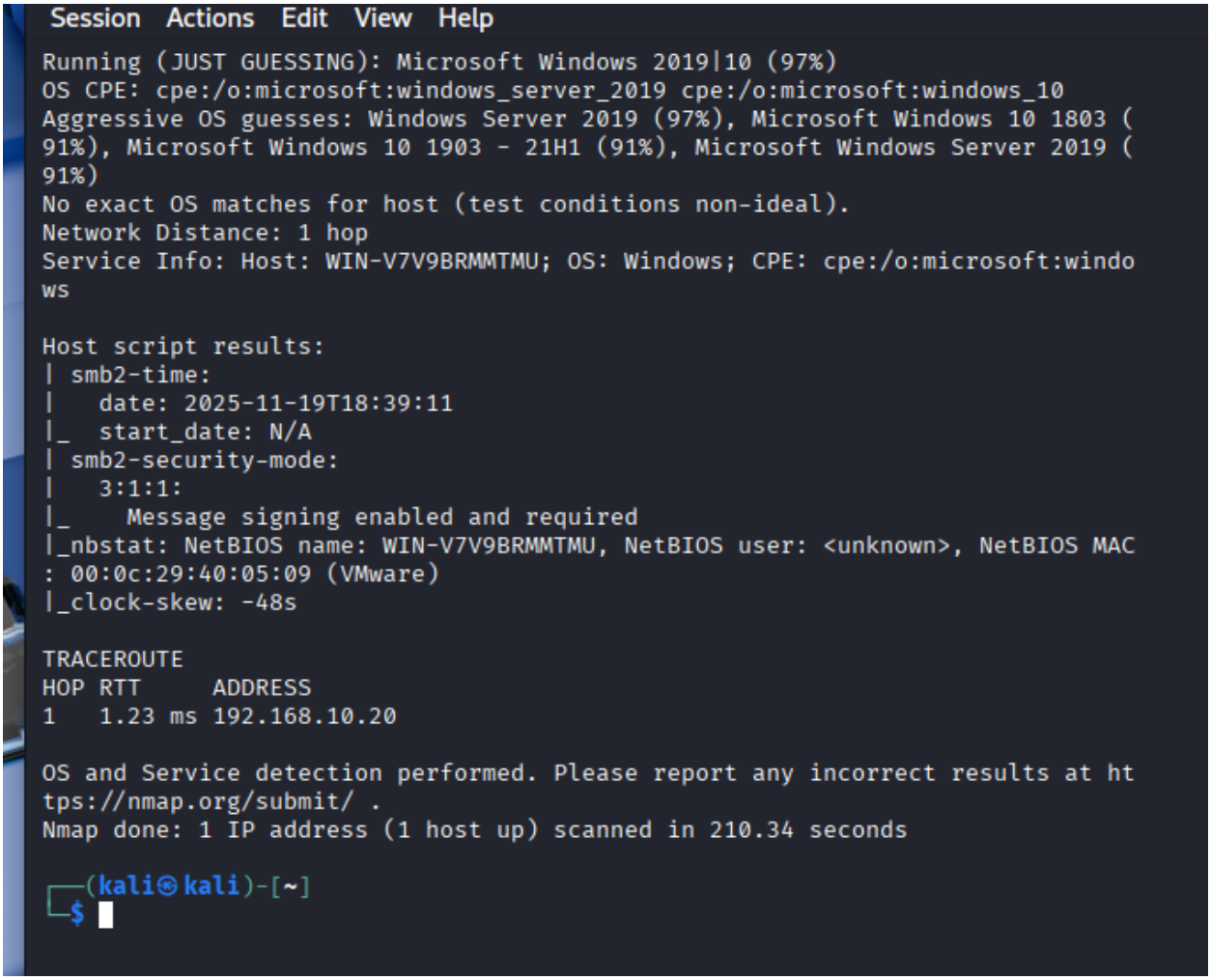


We use this command to guess the operating system running on the target.

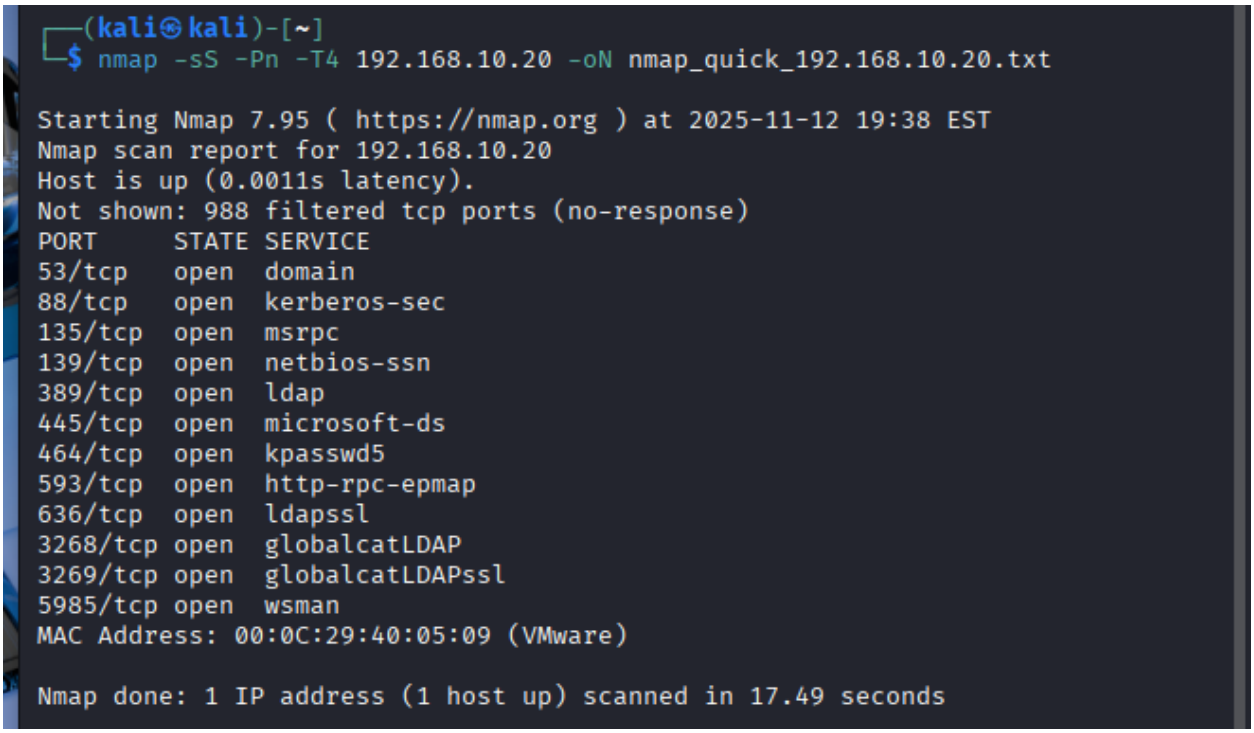


This command performs a full stealth scan of all ports, identifies service versions, runs OS detection and scripts, and saves the detailed results to a file.




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.



7.2 Nikto – Web Service Assessment

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

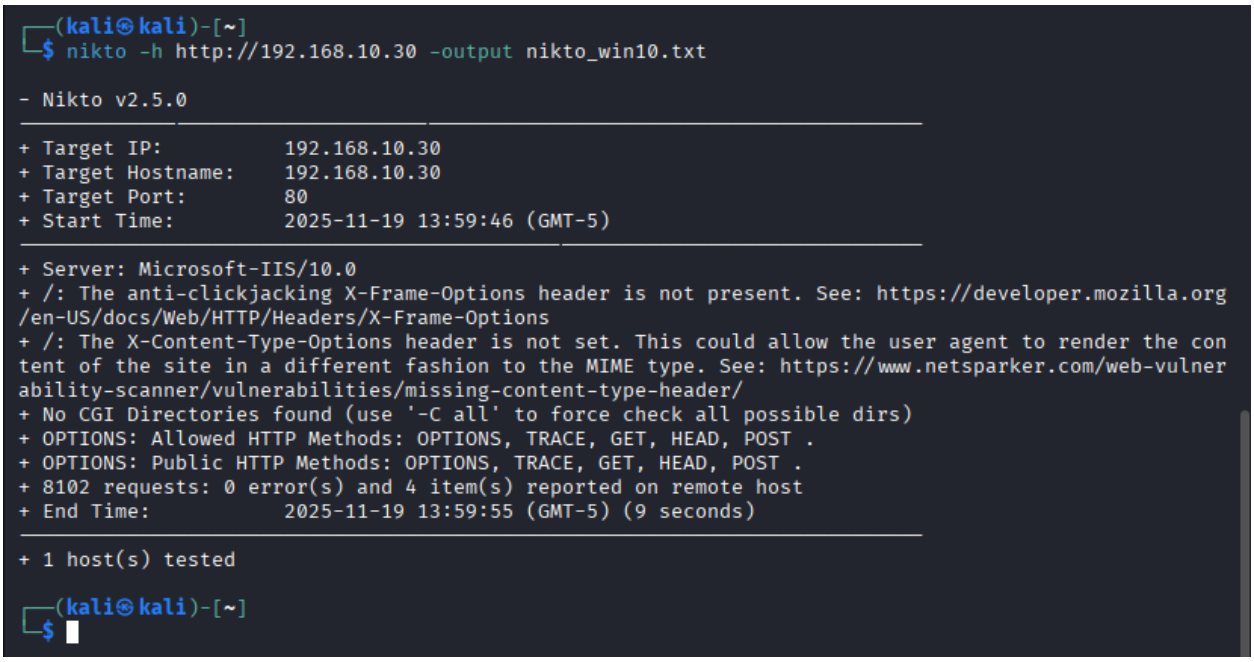
```
nikto -h http://192.168.10.30
```

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.



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
```

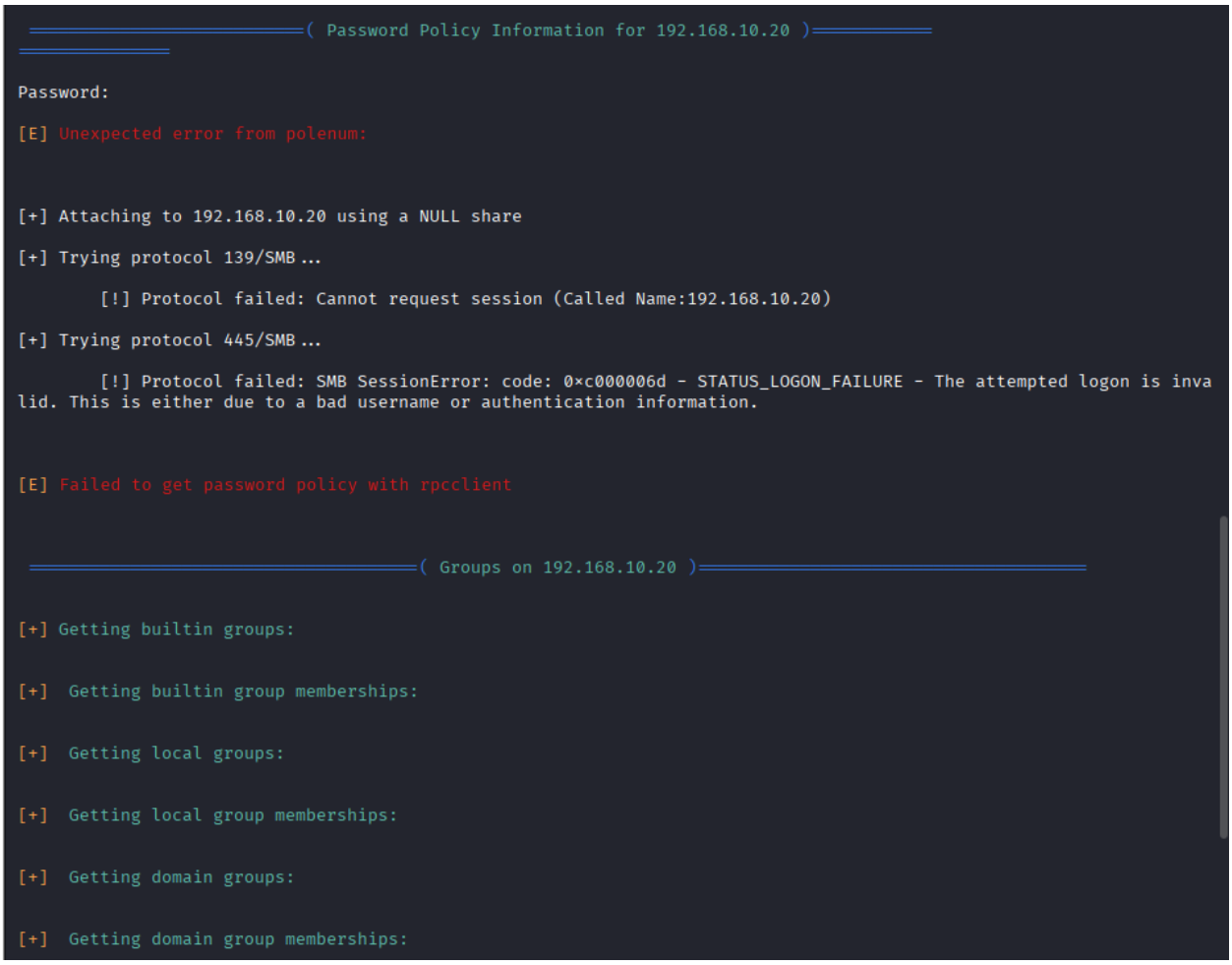
The screenshots for this part show:

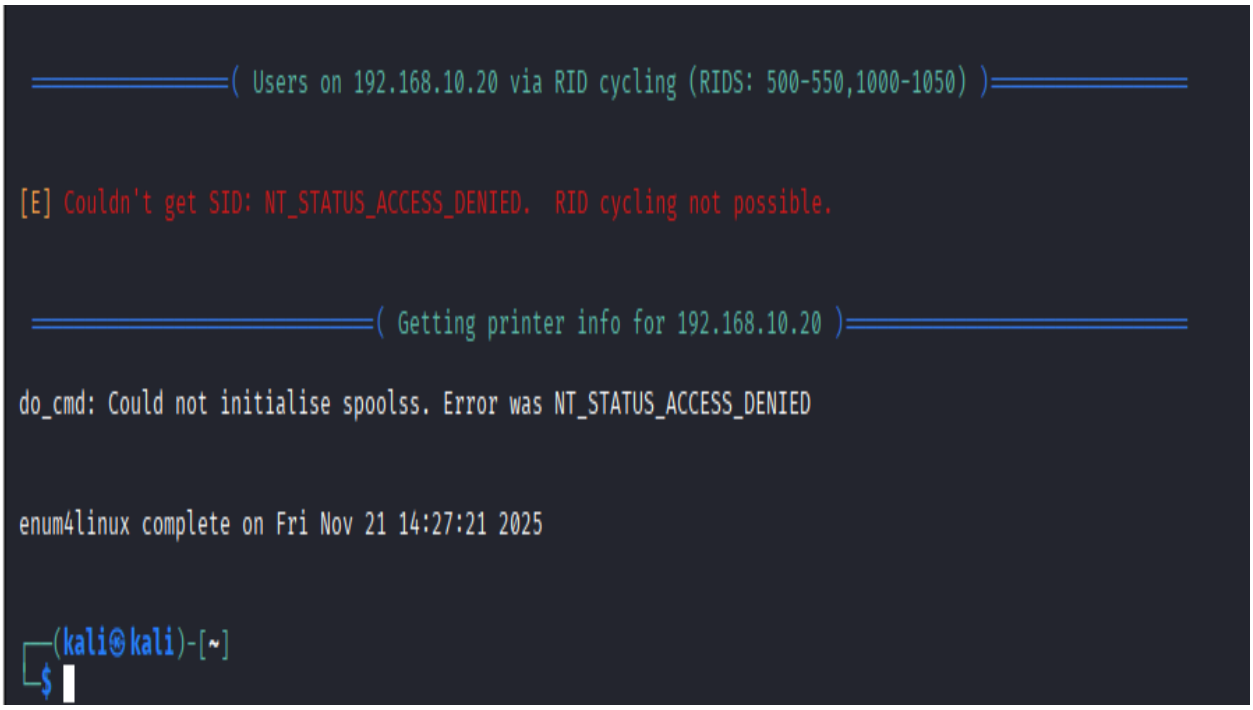
- Retrieved **NetBIOS names** and **domain/workgroup names**.
- Lists of **user accounts** and **groups**, where accessible.
- **Shared folders** and other SMB-related information.

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

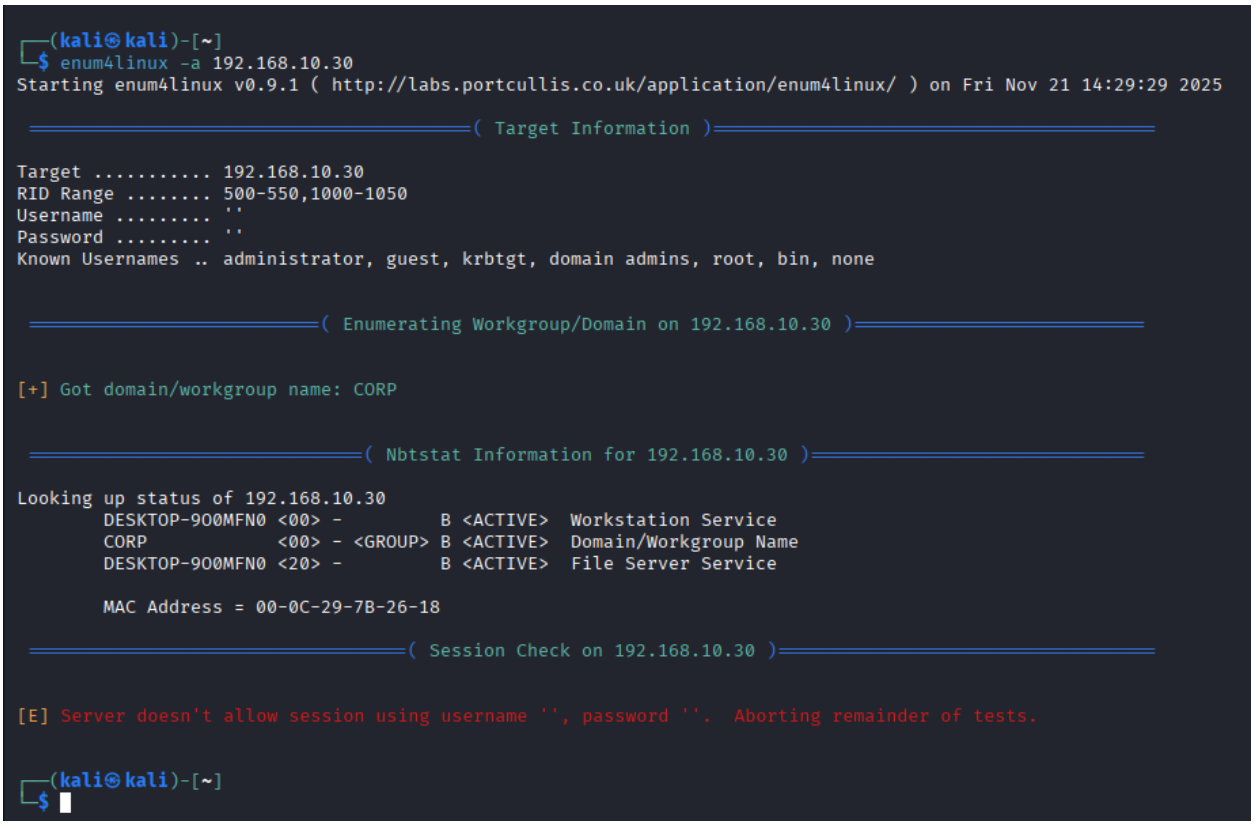
This command performs a full SMB enumeration on the target, gathering information such as users, shares, OS details, and domain data.

```
[+] 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 srvsvc. 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:
```


(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: 0×c000006d - 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.



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.

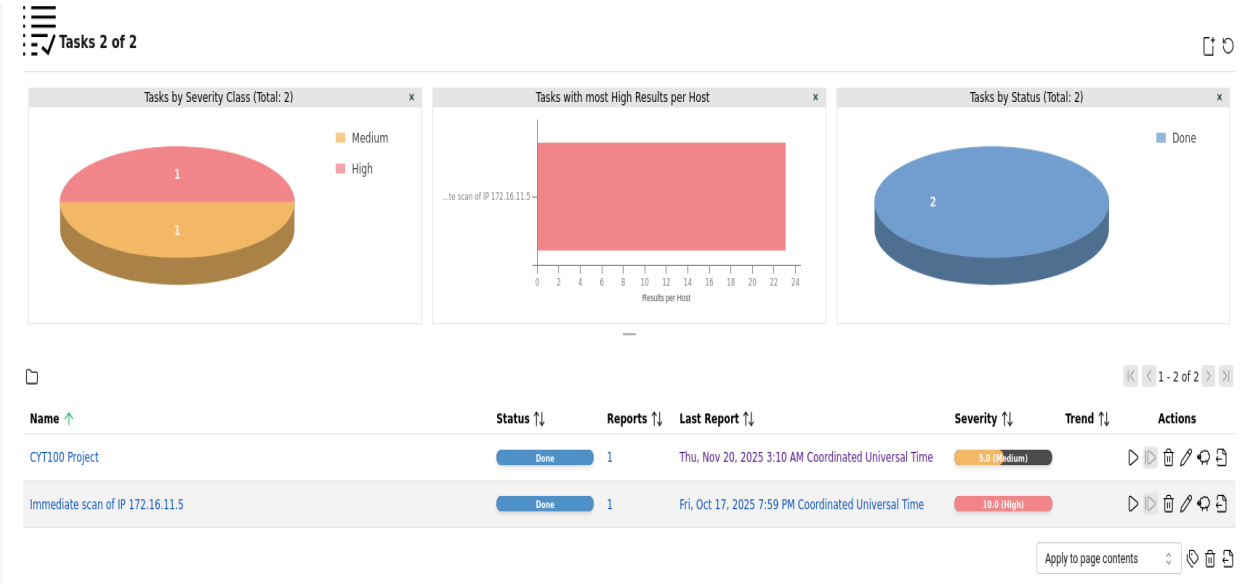
8.1 Scan Configuration

The screenshots show:

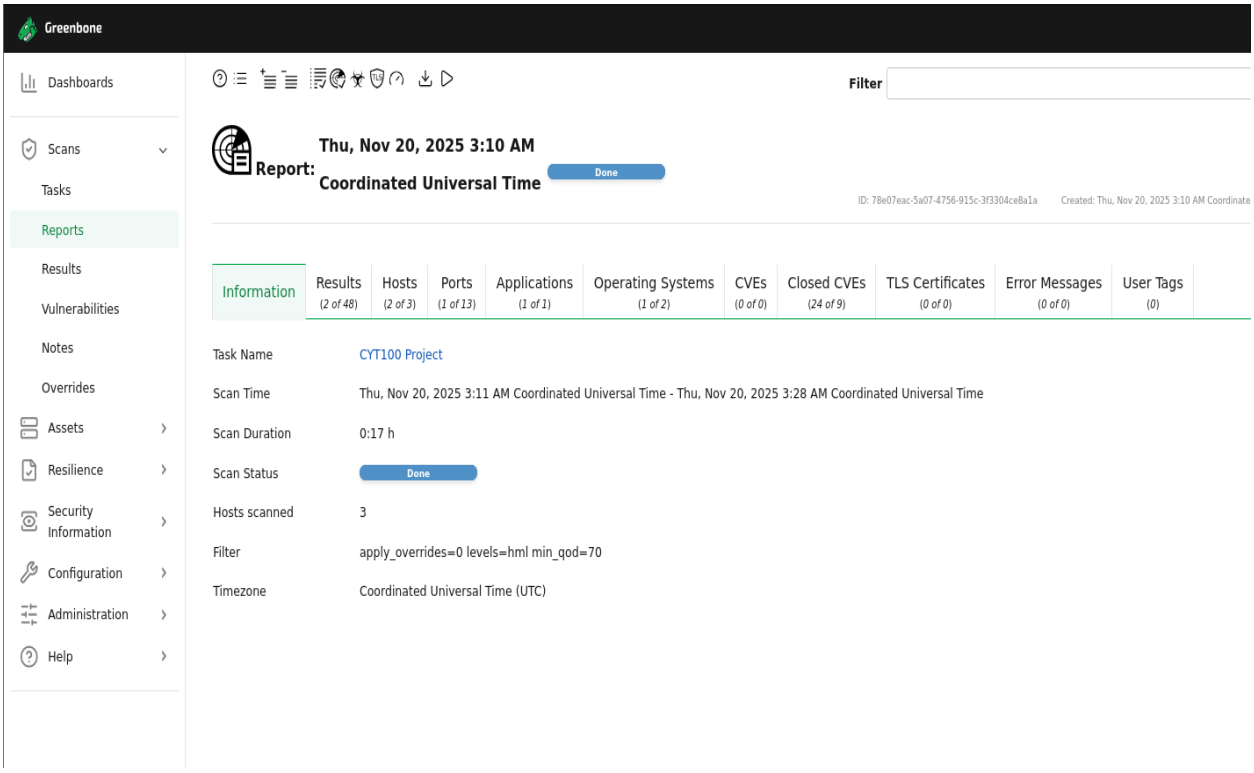
- The **OpenVAS web interface**, including authentication into the console.
- Creation of **targets** for each host (e.g., `192.168.10.20` and `192.168.10.30`).
- A **new task** configured to run a full or default scan profile against these targets.

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

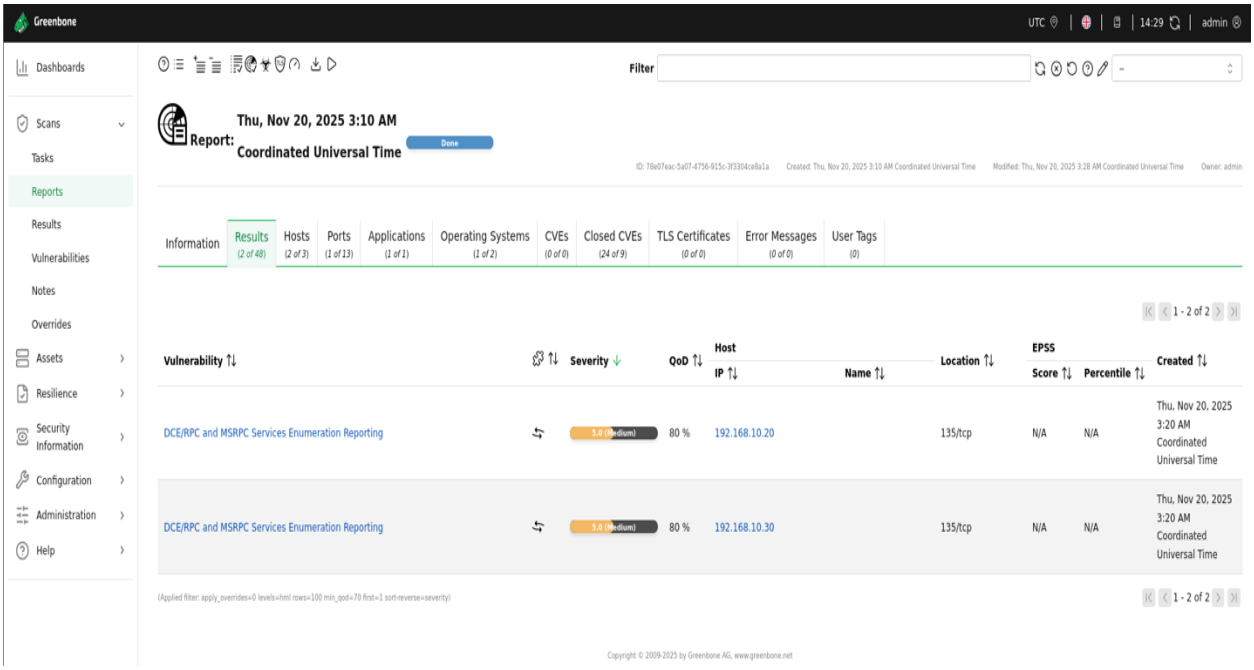
In this screenshot, we created a file under a name of CYT100 Project for gathering all the inforamtion of vulnerabilities of targeted machines and it also shows us the severity level of it.



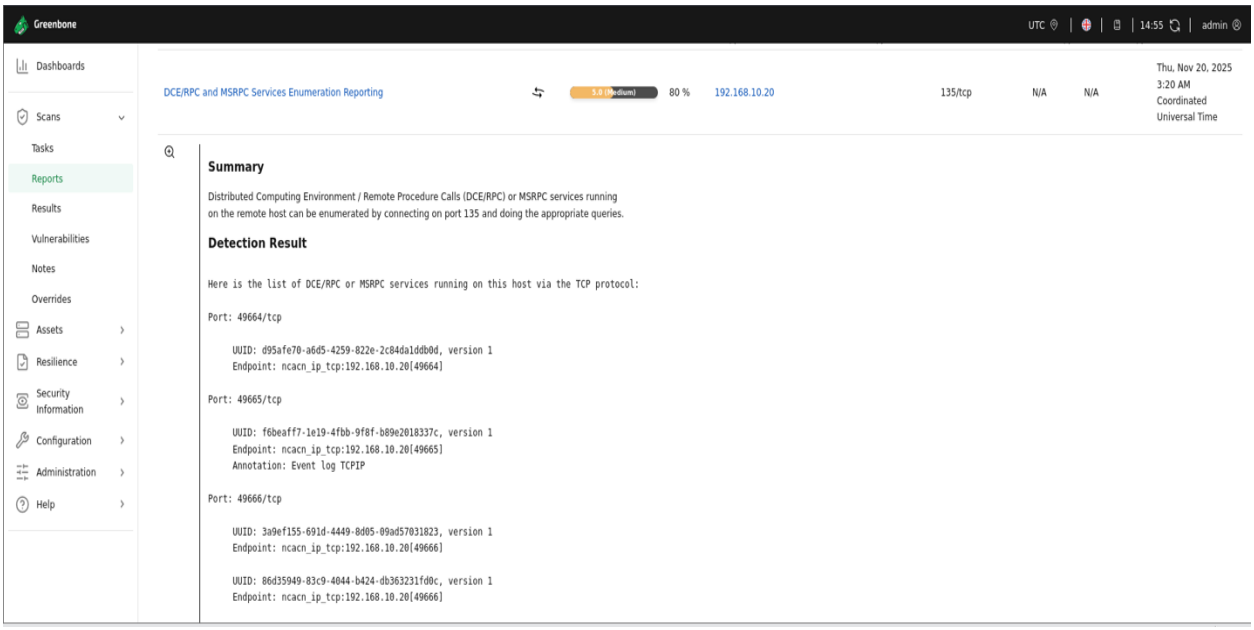
This is the proof of successful completion of OpenVAS scanning the vulnerabilities.



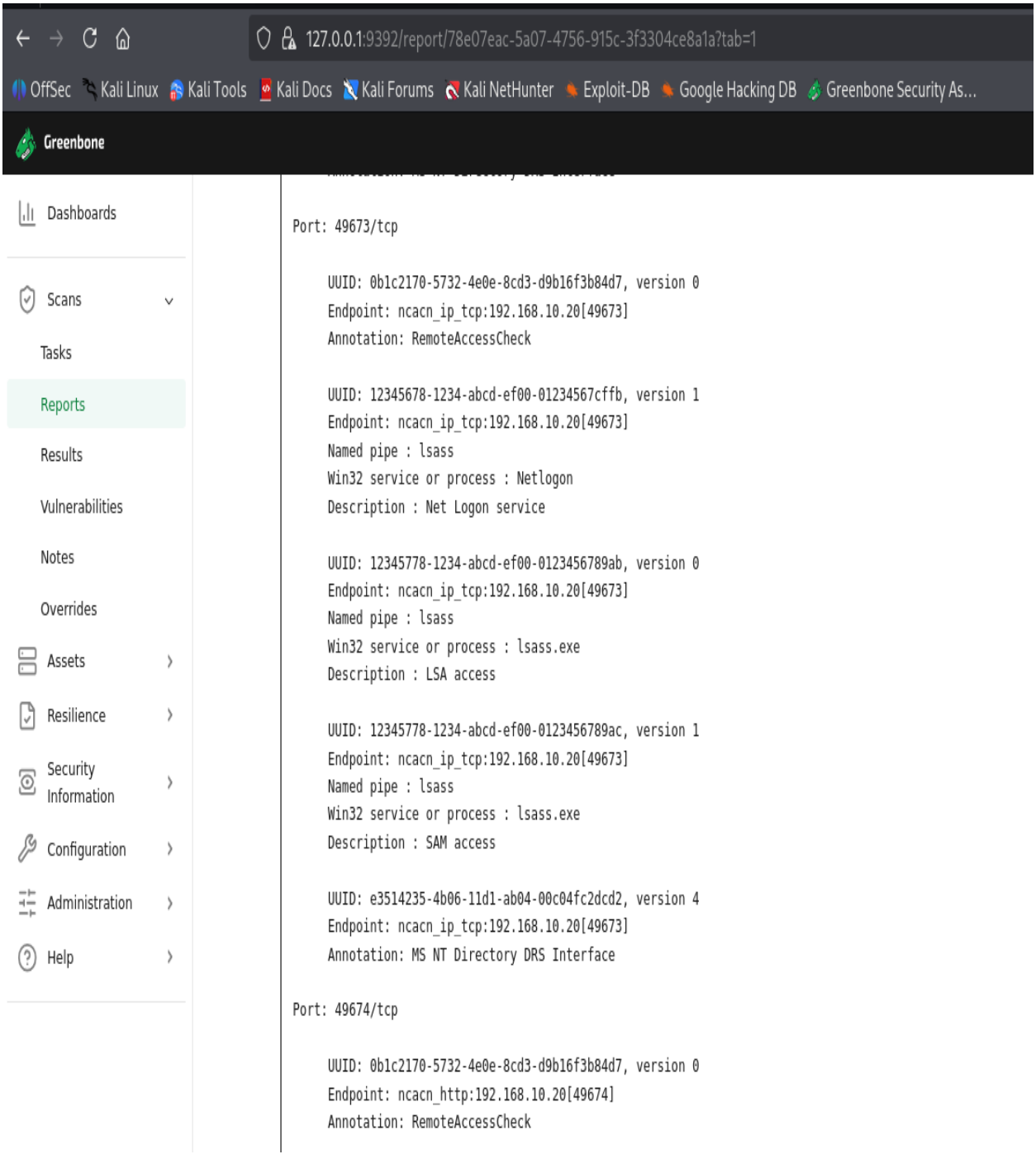
These results indicate that the scanner successfully enumerated DCE/RPC/MSRPC services on those two hosts, which is usually an informational-to-medium issue highlighting that these services are exposed and discoverable over TCP port 135.



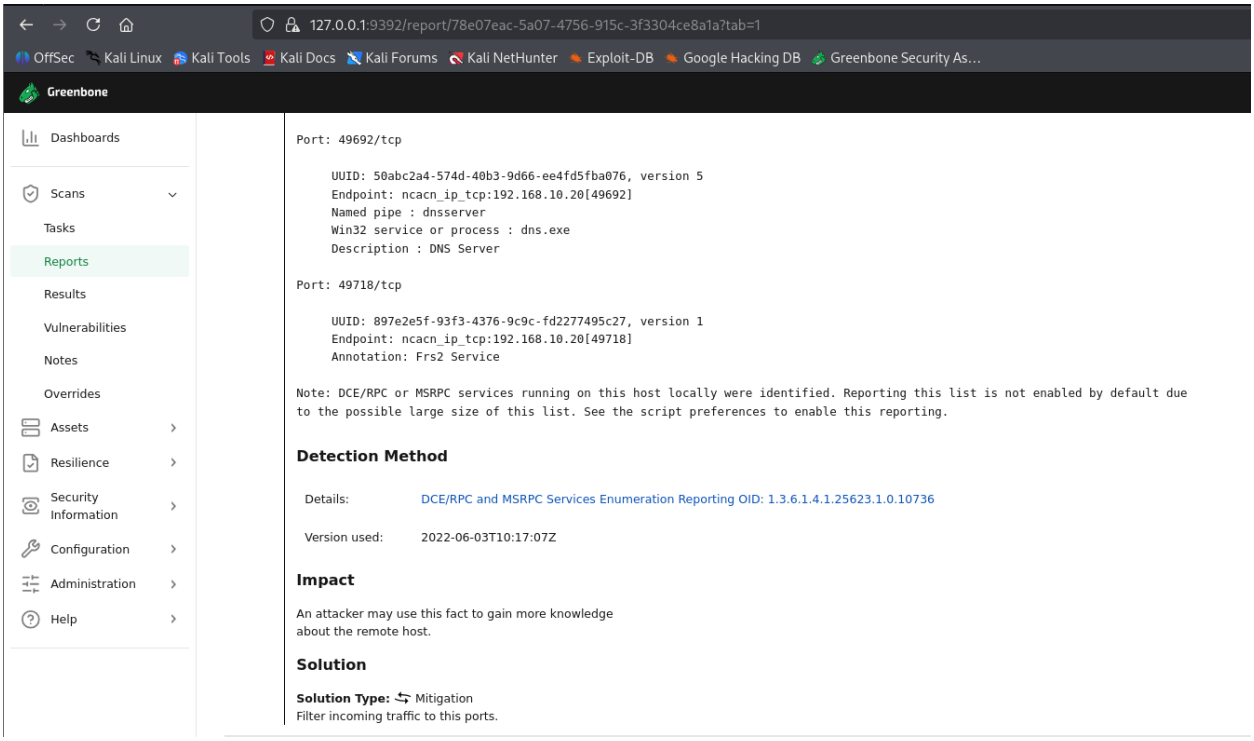
It is the detailed result page for that “DCE/RPC and MSRPC Services Enumeration Reporting” finding on host 192.168.10.20.



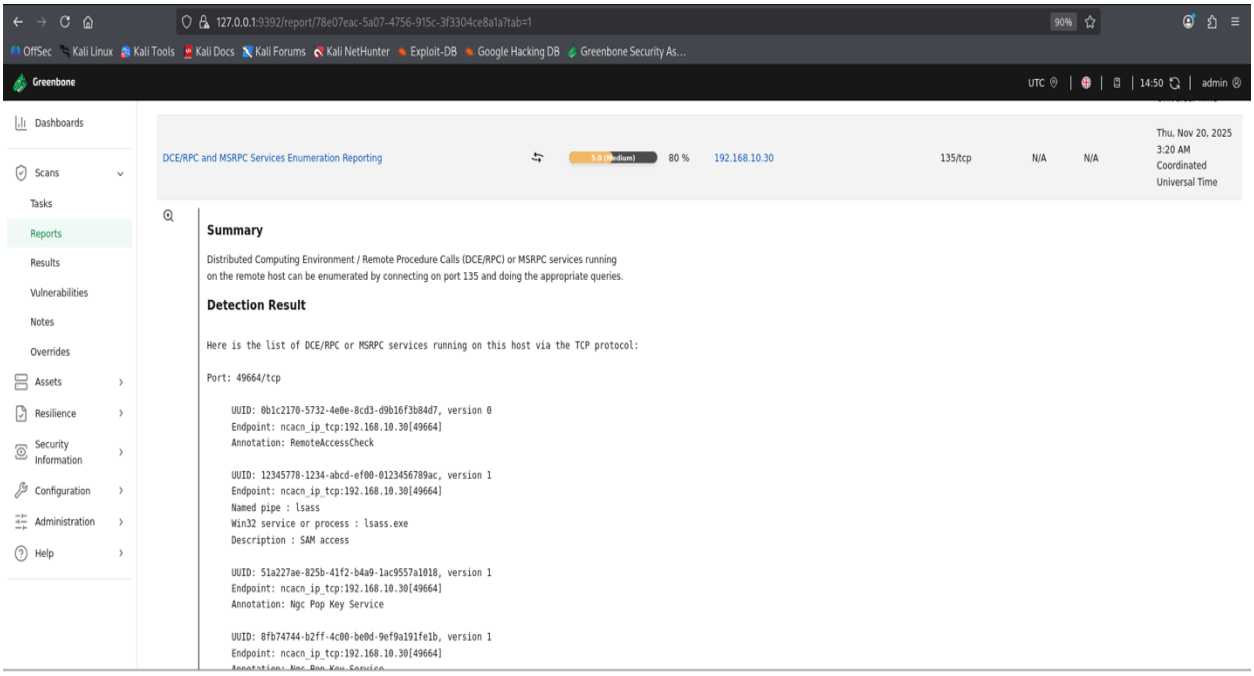
It shows that Greenbone detected several RPC endpoints over TCP ports 49673 and 49674 on host 192.168.10.20, exposing Netlogon and LSA/SAM services via the lsass named pipe but reporting them only as informational, not active vulnerabilities.



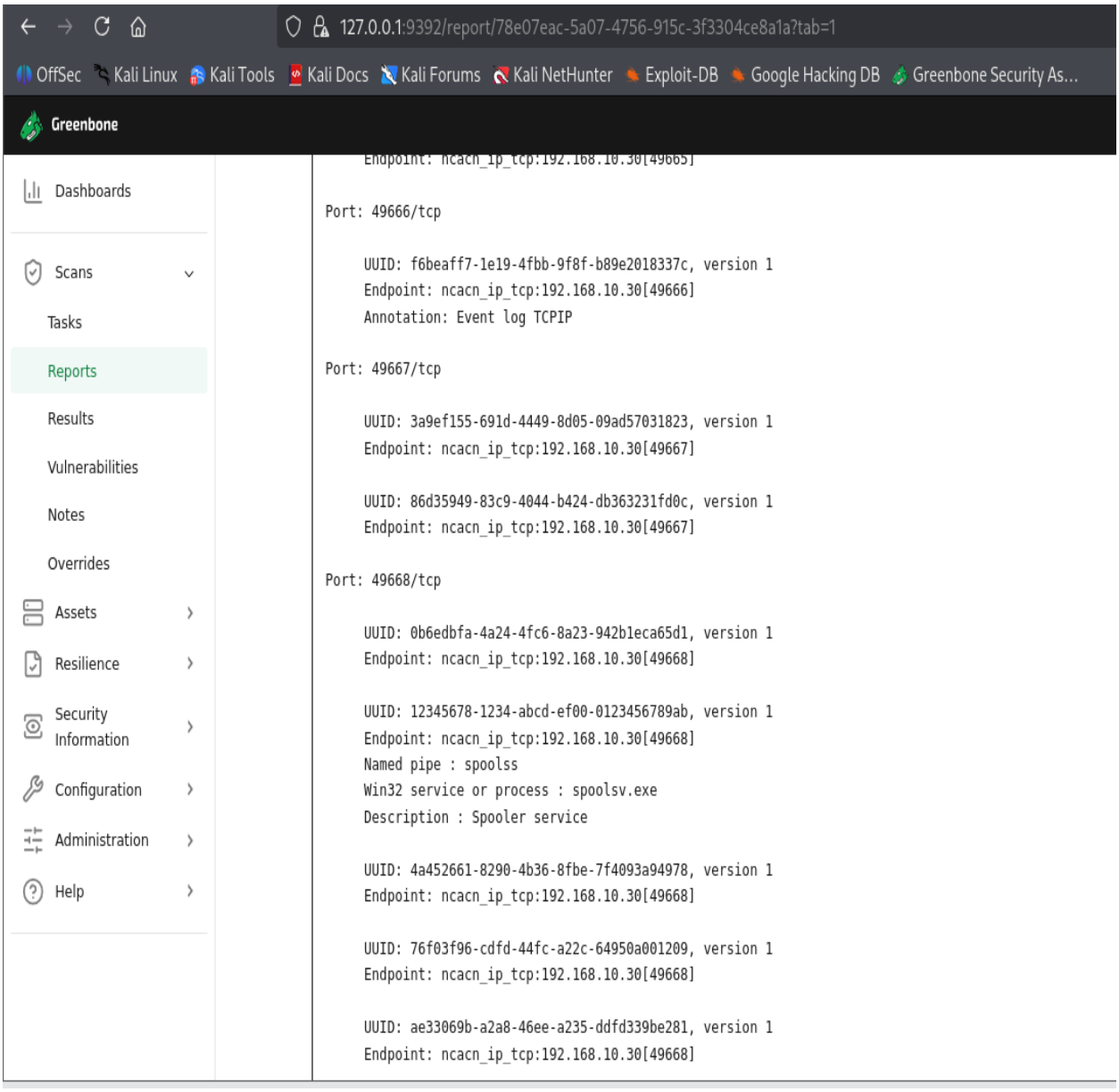
It lists DCE/RPC and MSRPC services found on the host (DNS and FRS on high TCP ports) and explains that this enumeration could give an attacker information about the system, with the suggested mitigation to filter incoming traffic to these ports.



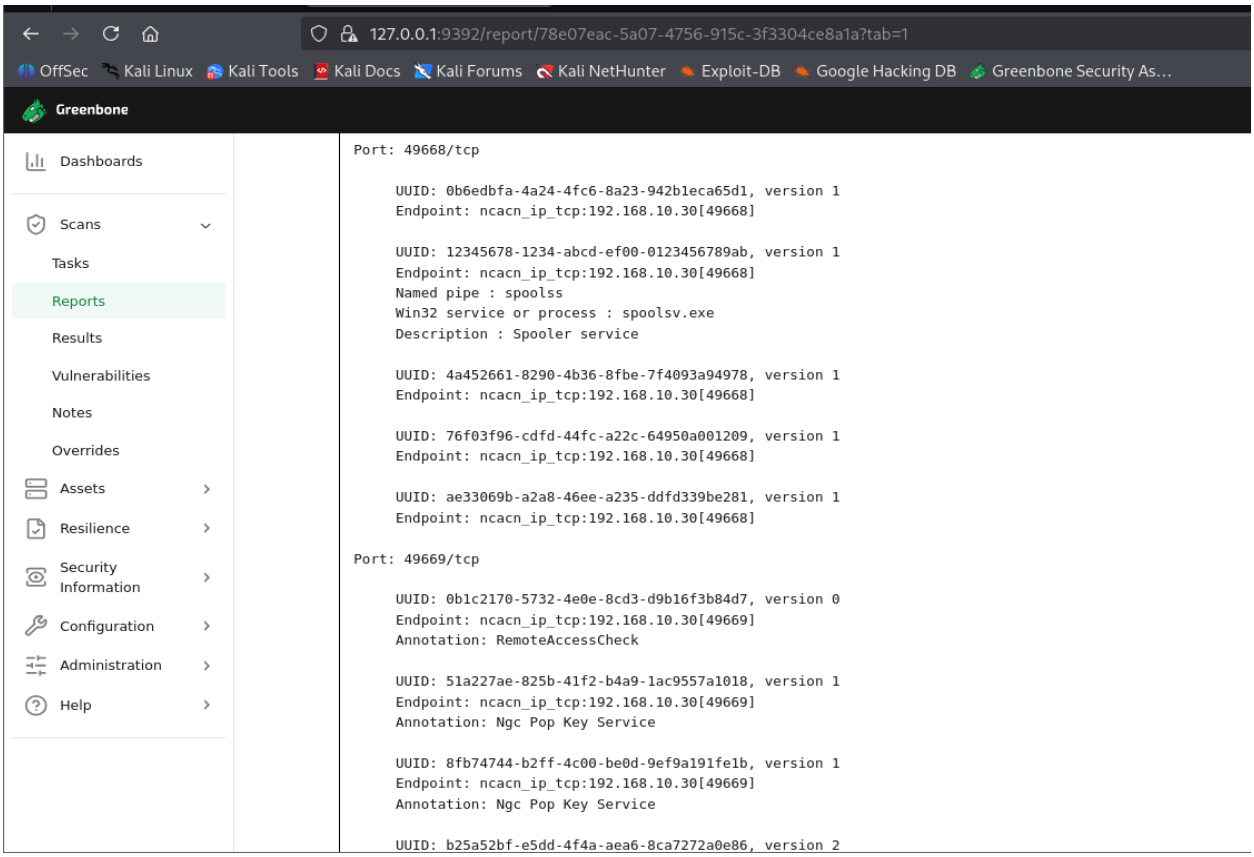
DCE/RPC and MSRPC services on 192.168.10.30 can be enumerated over TCP (port 135 and high ports), potentially giving attackers detailed information about available remote services.



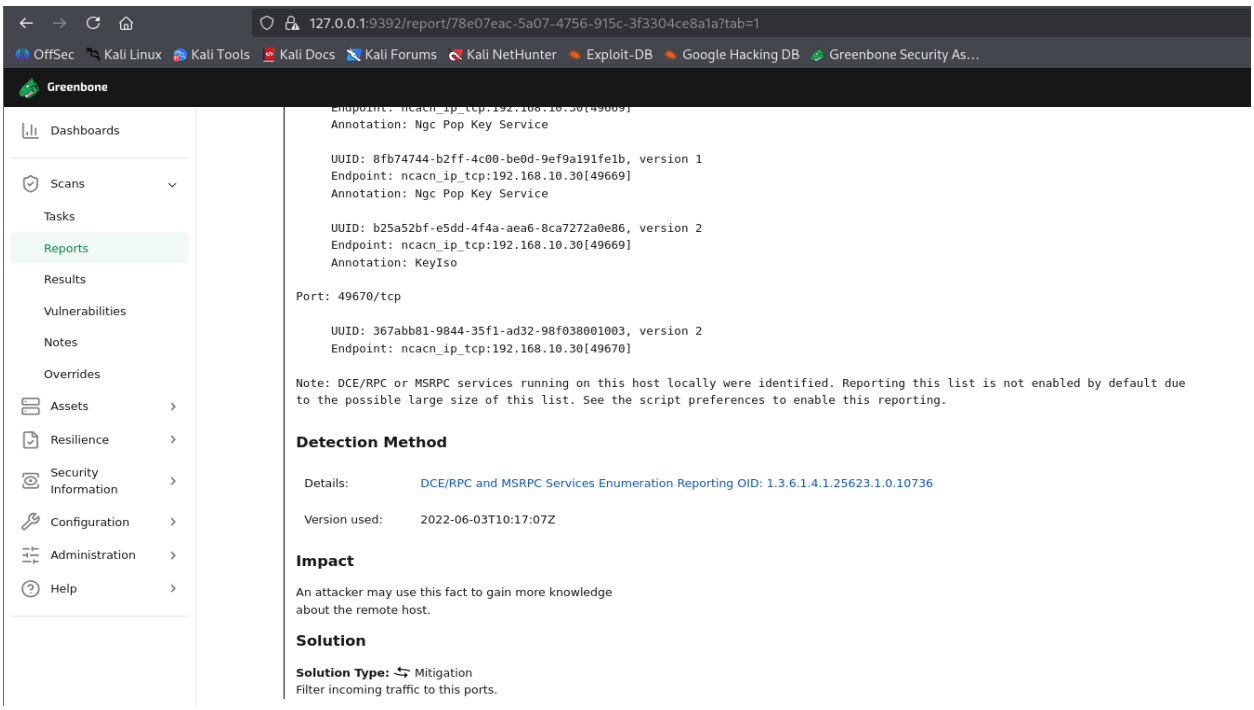
Greenbone lists additional DCE/RPC services exposed on high TCP ports (including Event Log over TCP/IP and the Print Spooler service), showing more enumerated services that could reveal host details to an attacker.



The 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.



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.



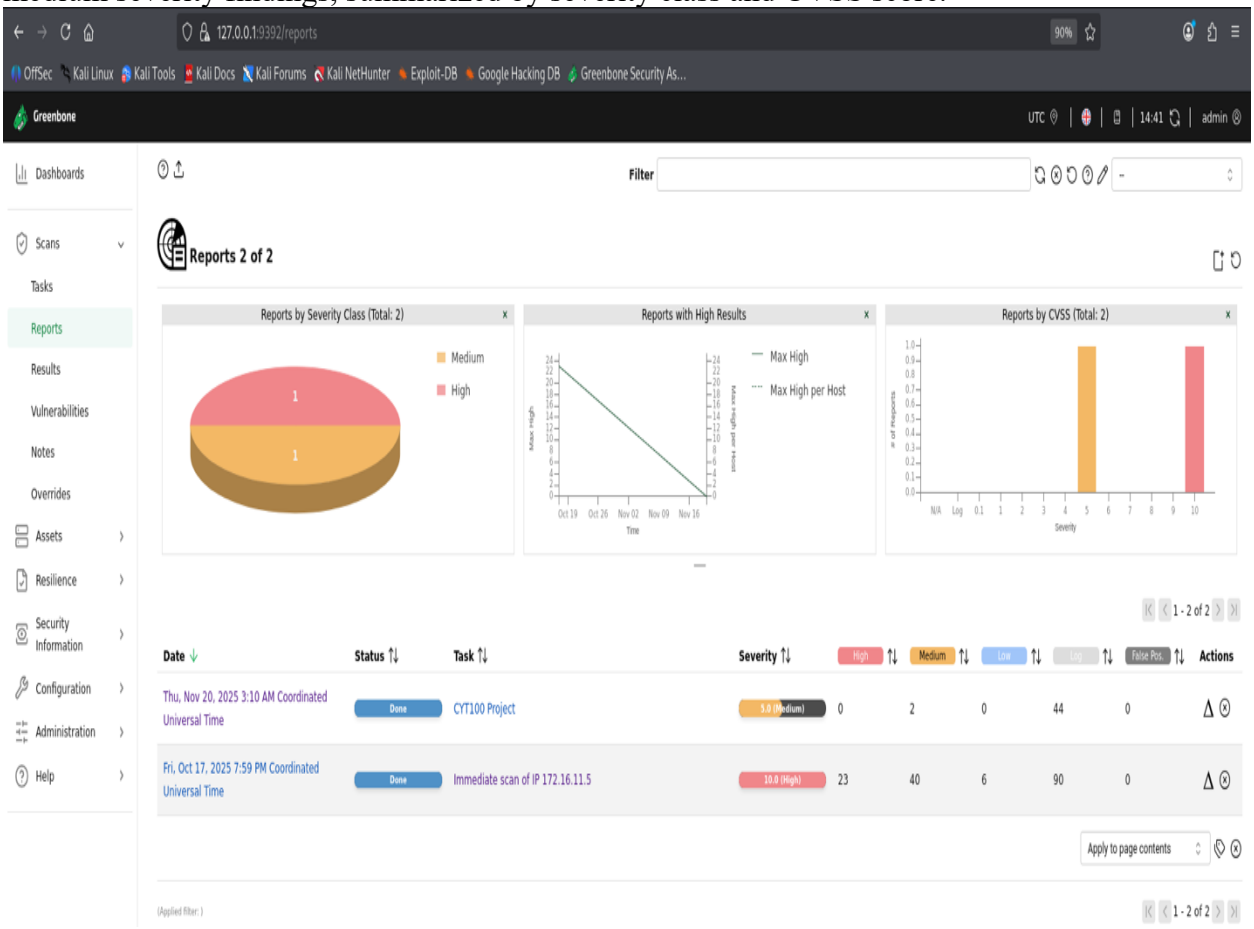
8.2 Scan Results

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

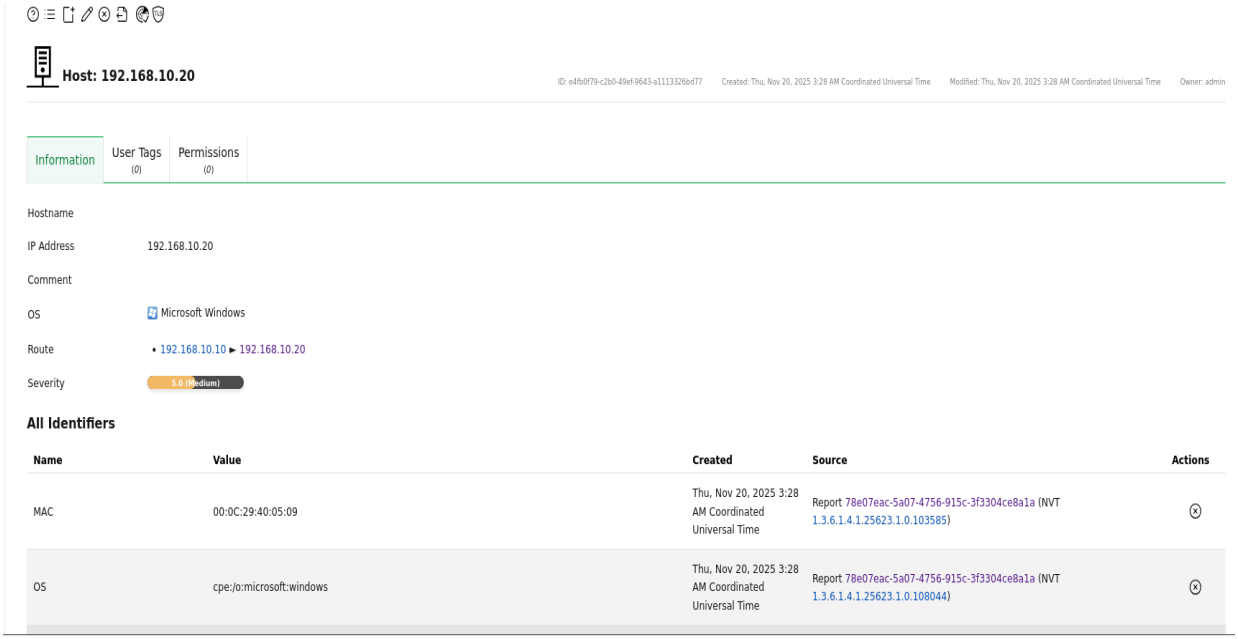
- Overall **scan summary**, including number of high, medium, and low severity findings.
- Example vulnerabilities, such as:
    - Missing security patches or outdated software versions.
    - Insecure services or protocols (e.g., SMBv1 being enabled).
    - Weak or default configuration on services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

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

Information | User Tags (0) | Permissions (0)

Hostname

IP Address 192.168.10.20

Comment

OS Microsoft Windows

Route • 192.168.10.10 ► 192.168.10.20

Severity 5.0 (Medium)

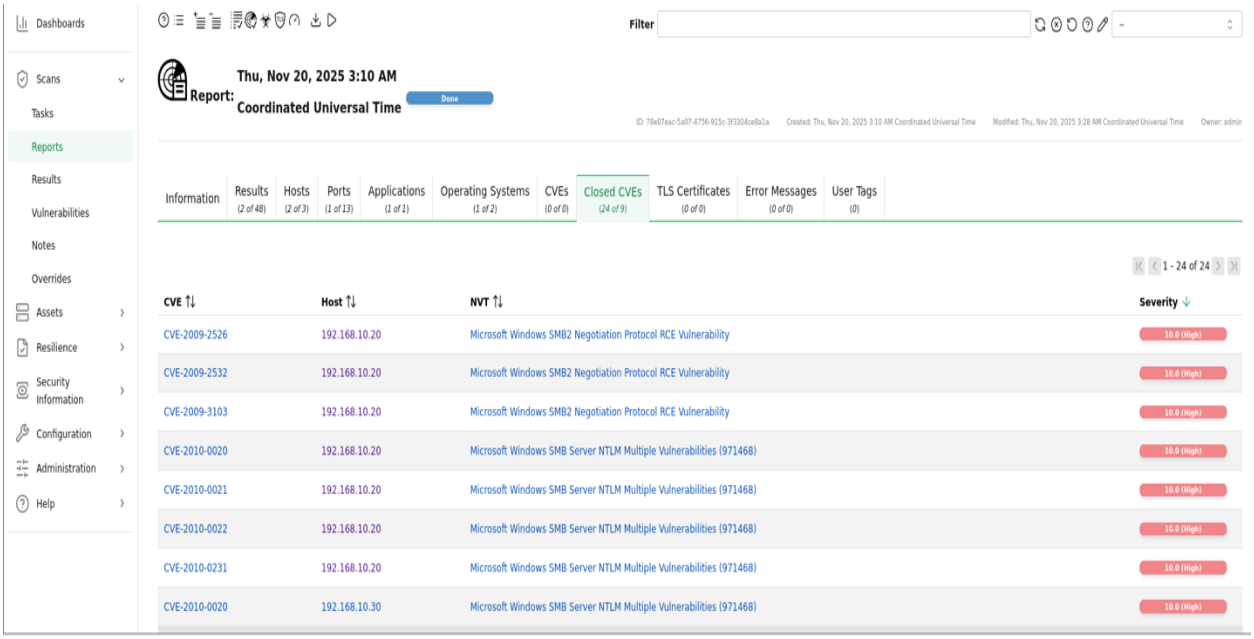
All Identifiers

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)	
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)	

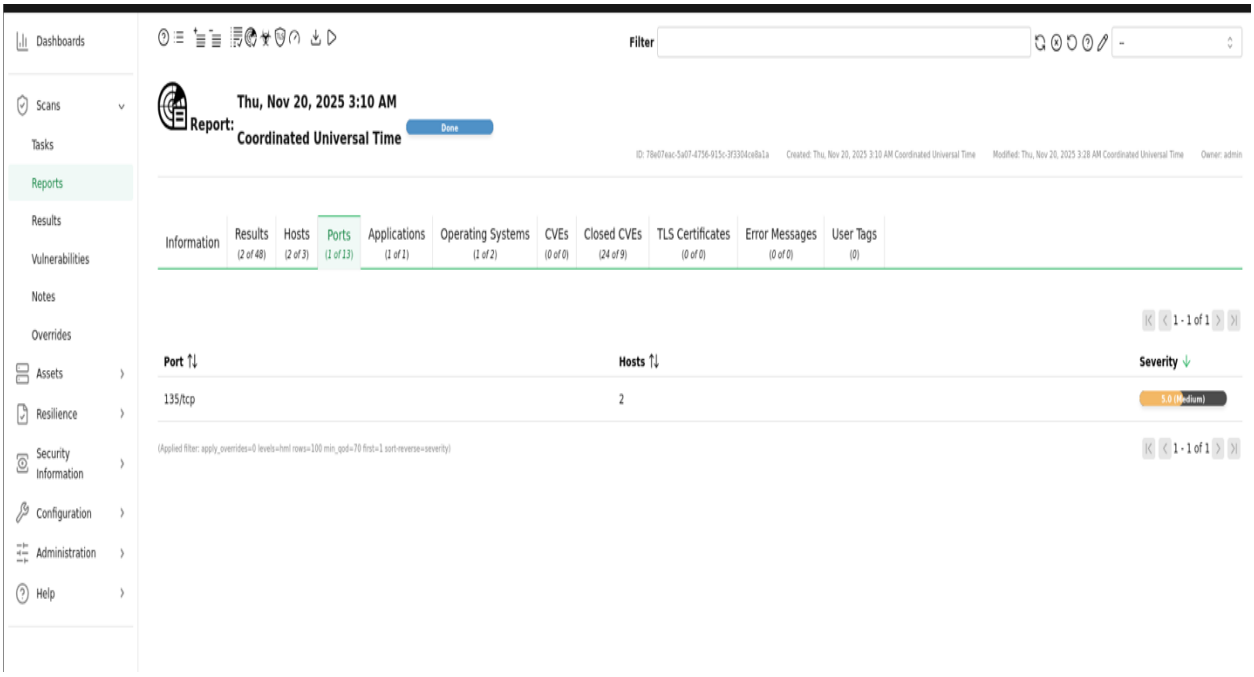
Greenbone identifies host 192.168.10.20 as a Microsoft Windows system cpe:/o:microsoft:windows:10.0.2.5413) with MAC 00:0C:29:40:05:09.

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)	
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)	
OS	cpe:/o:microsoft:windows:10.0.25413	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)	
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)	
ip	192.168.10.20	Thu, Nov 20, 2025 3:28 AM Coordinated Universal Time	Report 78e07eac-5a07-4756-915c-3f3304ce8a1a (Target Host)	

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.



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.

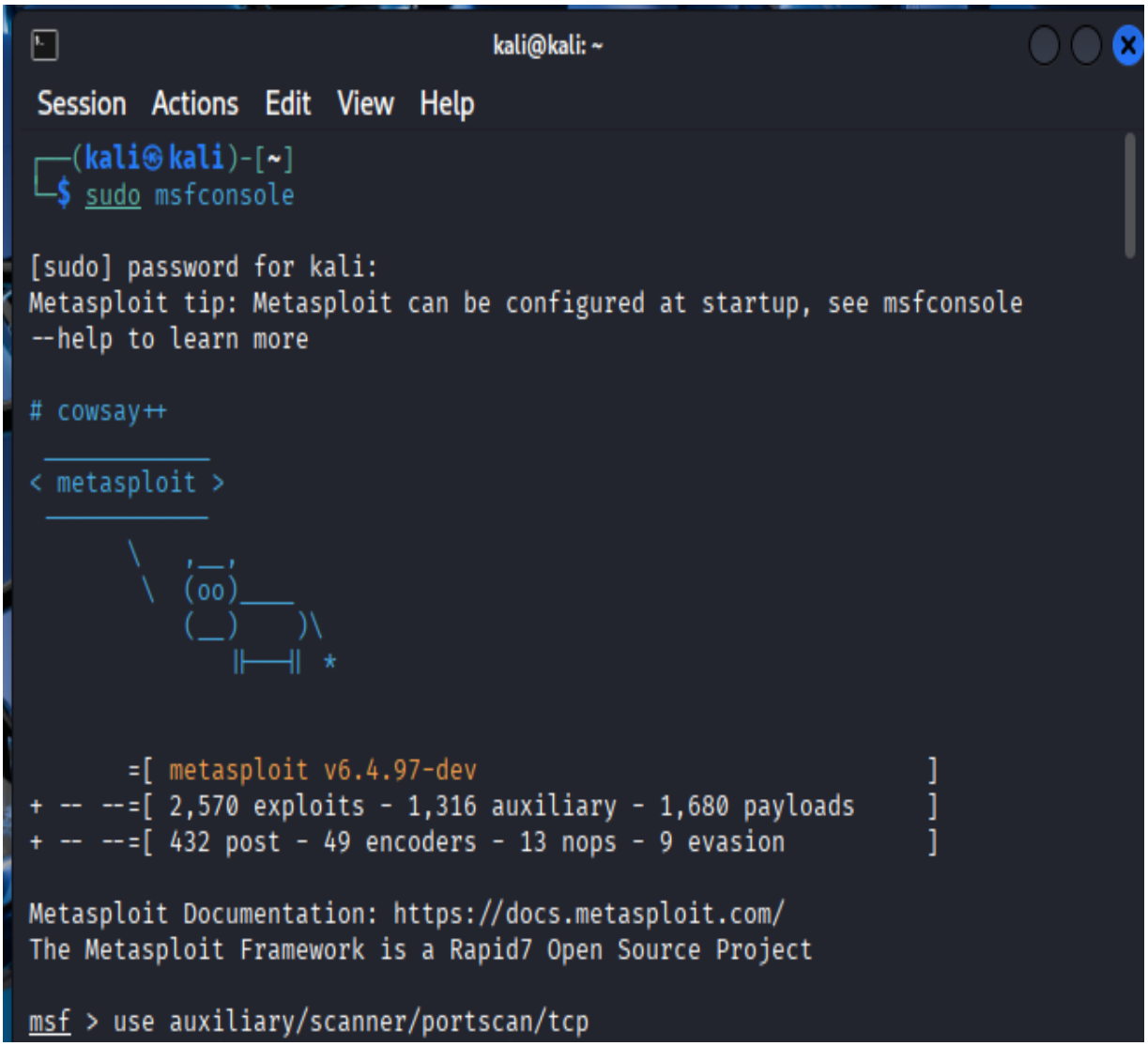


9. Exploitation using Metasploit

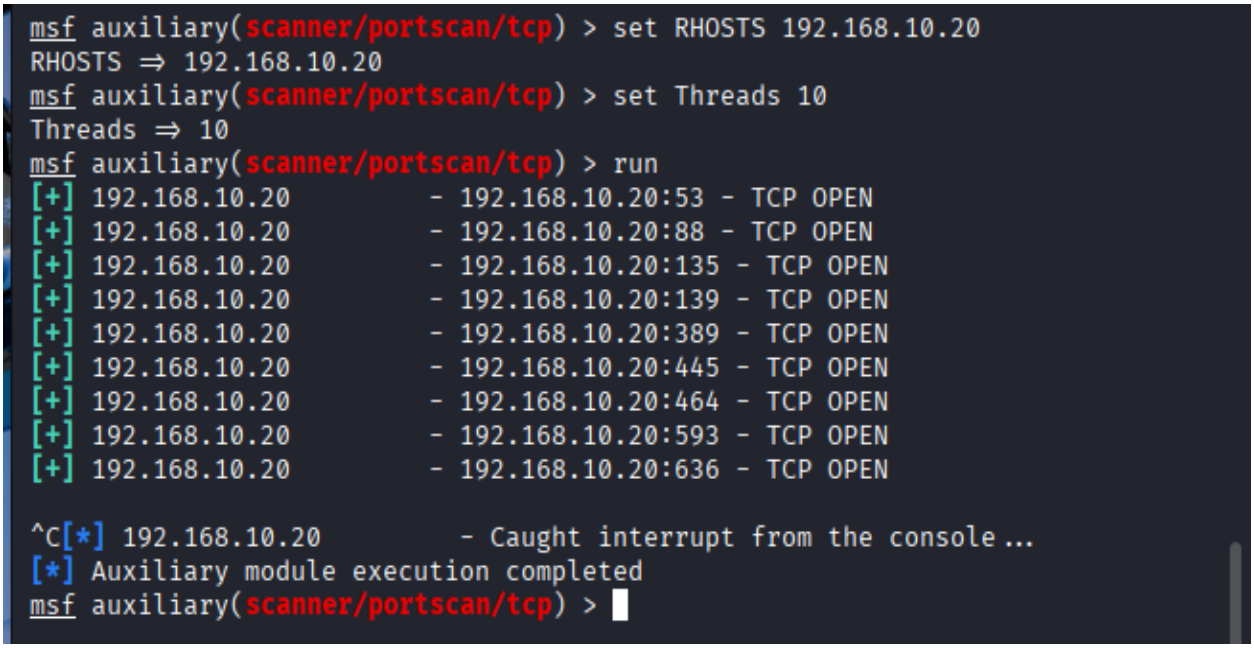
Using the high and medium-severity findings from OpenVAS and the port and service information gathered with Nmap and Enum4Linux, we selected a Windows vulnerability that allowed remote code execution on one of our lab systems. Our goal in this phase was to move from “theoretical vulnerability” to **actual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

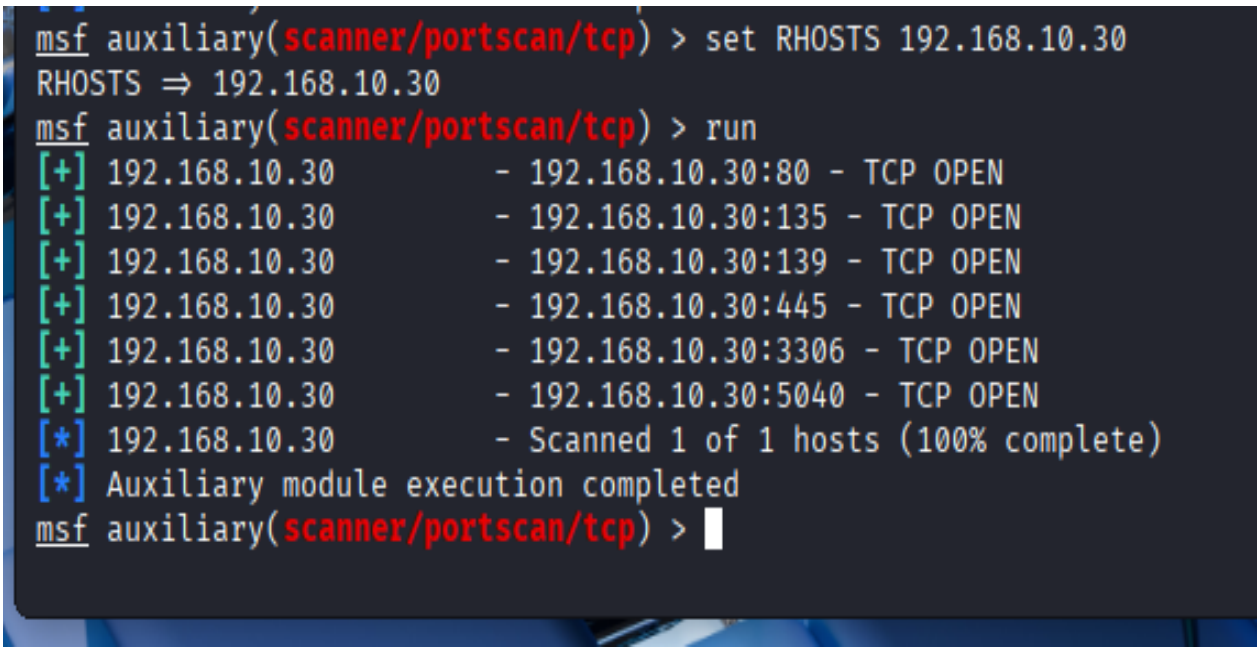
Kali terminal showing Metasploit Framework v6.4.97-dev starting via sudo msfconsole



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.



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



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:Password!' 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:

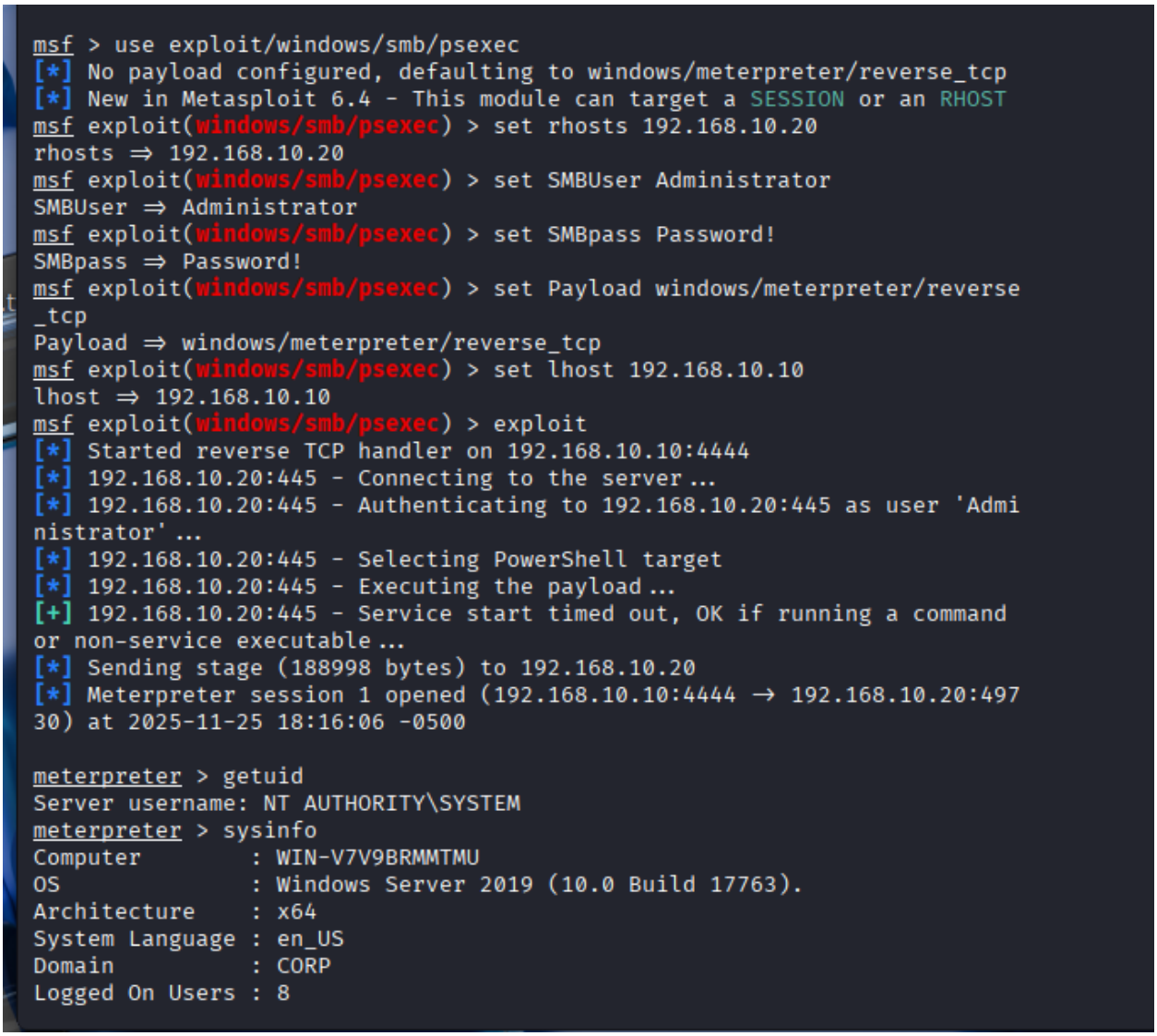
- A message indicating that the exploit was successful.
- A new session listed via the `sessions` command (for example, `session 1 opened`).

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.



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.

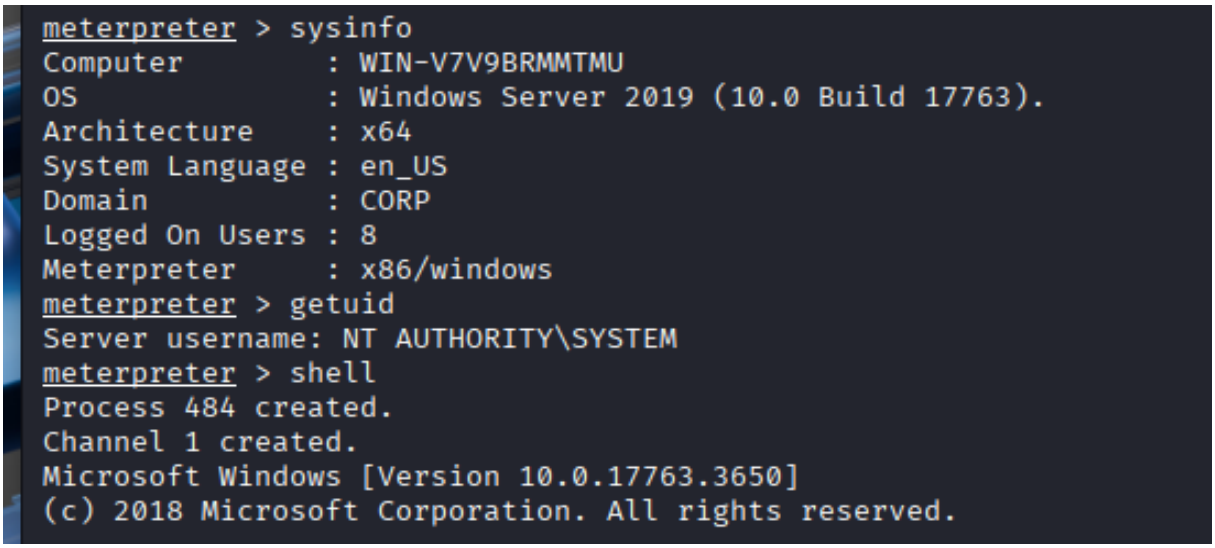
3. **Credential Access**

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

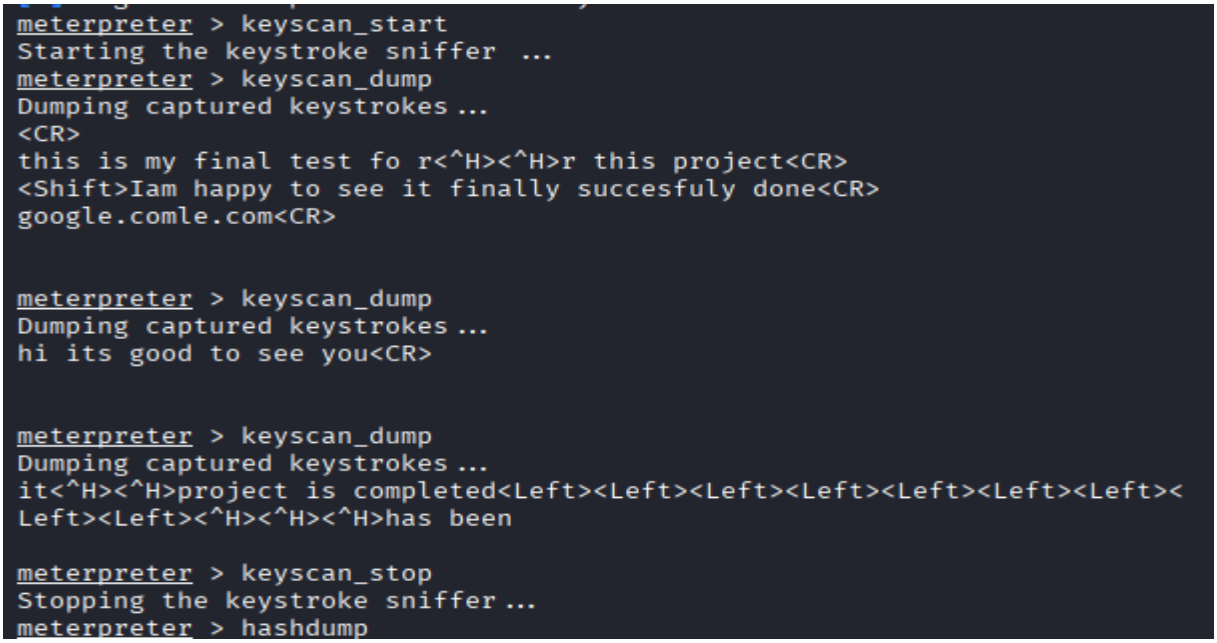
    This showed how a single compromised machine can become a **stepping stone**: hashes can often be cracked offline or reused elsewhere (for example, in pass-the-hash attacks) if password reuse is common. We did not attempt to crack any hashes beyond the scope of the project; our goal was simply to show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

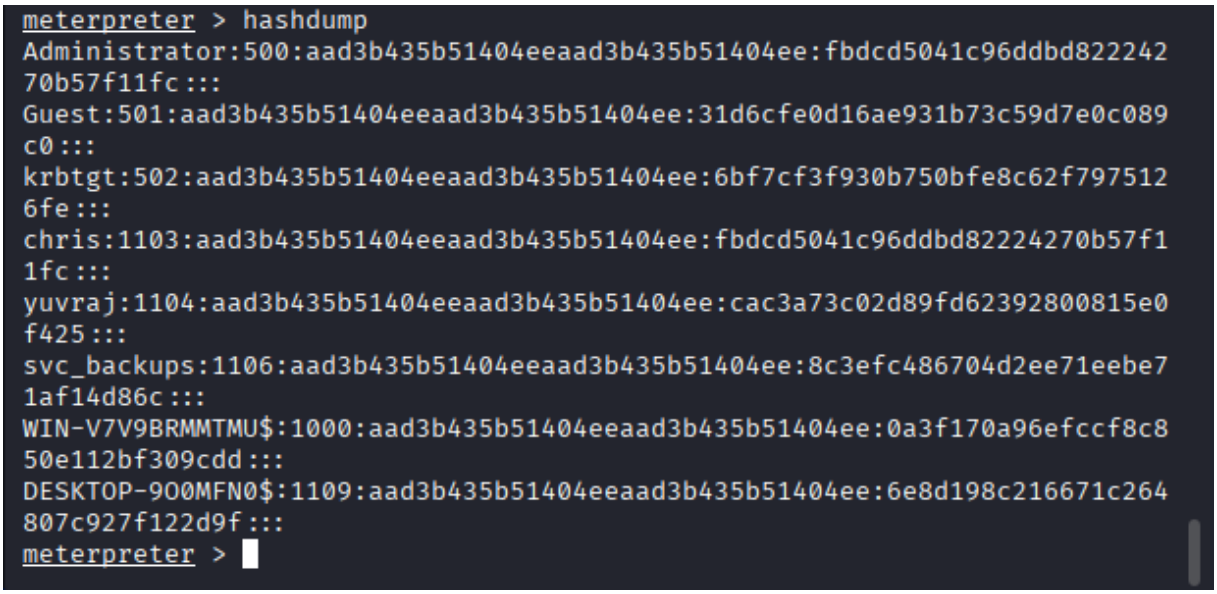
This is a proof of that we have successful completion of post-exploitation phase, and we have meterpreter's sysinfo and getuid commands confirm you are NT AUTHORITY\SYSTEM on a CORP domain Windows Server 2019, and you have spawned a standard Windows command shell from that SYSTEM-level Meterpreter session.



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'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.



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.

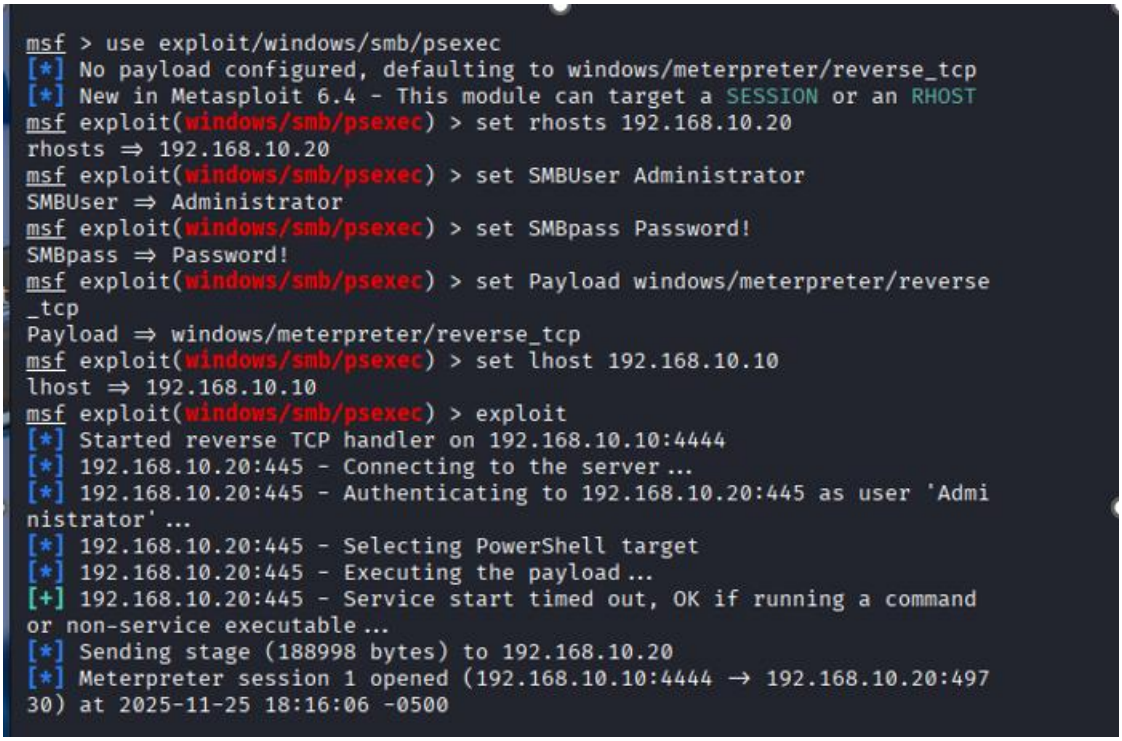
We explored two main approaches:

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.



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.