IMF Smart Education

CYBERSECURITY MASTER'S DEGREE
ONLINE

Building and Breaking an Active Directory Environment

Master Thesis developed by: Pablo Gutiérrez Bueno
Master Thesis tutor: Raimundo Alcázar Quesada

– Barcelona –

19.04.2025

Appreciation / acknowledgements

I would like to begin by expressing my deepest gratitude to my family and friends, whose support and understanding have been essential throughout this intense year. Balancing a full-time job alongside this Master's program has demanded significant time, focus, and sacrifice. Without their encouragement and patience, this journey would not have been possible.

I am also grateful to the professors from my Telecommunications Engineering degree, whose teaching provided me with a solid technical foundation. The knowledge and mindset developed during those academic years have been instrumental in successfully tackling the challenges of this Master's program and are clearly reflected in the outcome of this thesis.

Lastly, I want to thank the entire Master's program team for the structure of the curriculum, the quality of the content, and their responsiveness throughout the course. I would like to extend a special thank you to Raimundo, my thesis advisor, whose guidance has been invaluable. Despite limited interaction with other instructors, Raimundo consistently provided clear, practical advice and timely support—especially in the moments when I felt stuck.

This project is as much a personal milestone as it is a professional one, and it is thanks to all of them that I have reached this point.

Abstract

Note: The superscript numbers (e.g., [1], [2]) indicate the first appearance of technical terms. All referenced terms are defined in the Glossary.

This Master's Thesis presents a practical security assessment of a corporate environment built upon ***Active Directory*** *(AD)* [1], the cornerstone technology for identity and access management in enterprise networks.

The goal of this project was to simulate a realistic internal network by designing and deploying a fully functional AD infrastructure, intentionally incorporating common misconfigurations and weaknesses observed in real-world environments. The assessment followed a manual, attacker-oriented methodology consisting of:

- Environment deployment and vulnerability injection.
- Enumeration and reconnaissance of network resources.
- Exploitation of attack paths such as ***Kerberoasting*** [2], ***AS-REP Roasting*** [3] and ***DCSync*** [4].
- Post-exploitation techniques and domain compromise.

By avoiding automated frameworks and focusing on step-by-step analysis, the thesis offers a clear view of how internal attackers navigate *Active Directory* ecosystems—from identifying weak entry points to achieving full domain control. All phases were carried out in a controlled laboratory and documented in detail.

This work highlights the critical importance of periodic auditing, strong access control policies, and awareness of internal threat vectors. It provides actionable insights and mitigation strategies, contributing to a deeper understanding of cybersecurity in enterprise environments.

Beyond its academic purpose, this project also serves as a practical foundation for internal red teaming and cybersecurity training initiatives within enterprise settings. Its methodology and findings can be easily adapted by security teams to simulate adversarial behaviour, assess organizational resilience, and design targeted mitigation strategies aligned with industry best practices.

Keywords

- *Active Directory*
- Pentesting
- Internal Threat Simulation
- Privilege Escalation
- *Kerberoasting*
- *AS-REP Roasting*
- *DCSync* Attack
- Password Reuse
- Security Misconfigurations
- Post-Exploitation

Motivation

The decision to focus this thesis on *Active Directory* security stems from both personal interest and hands-on experience. I am currently preparing for the eJPTv2 (Junior Penetration Tester) certification, with the goal of pursuing the OSCP (Offensive Security Certified Professional) — a benchmark qualification and aspiration for many aspiring penetration testers. Both certifications emphasize hands-on techniques in realistic environments, with a strong focus on *Active Directory* as a recurring challenge. In fact, most of the machines I successfully exploited involved AD infrastructures, which provided me with solid experience in identifying common misconfigurations and weaknesses.

This exposure inspired me to go one step further and build a complete *Active Directory* infrastructure from scratch. The intention was not only to understand its components from a defensive perspective but to explore how these systems can be compromised in realistic conditions. In cybersecurity, mastering the offensive side is essential for building robust defence mechanisms—hence the idea that *the best defence is a good attack*.

One of the alternative paths I initially considered for this thesis was bug bounty hunting, which focuses on identifying web application vulnerabilities. However, given the stage of my professional career, I believe it is more valuable to develop a solid understanding of *Active Directory*. AD integrates various layers of corporate infrastructure—authentication protocols, user and group management, network services, access control mechanisms, and *privilege escalation* [5] paths. These are essential components in any mid-to-large scale organization, making AD security a foundational topic for any cybersecurity professional.

Furthermore, I believe that AD security is one of the most critical areas within enterprise environments, alongside:

- Cloud Security
- Endpoint Detection and Response (EDR)
- Identity and Access Management (IAM)
- Security Operations Centre (SOC) Monitoring
- Network Segmentation and Zero Trust Architectures

This project also represents my first fully developed internal cybersecurity lab. It has allowed me to gain a comprehensive, real-world perspective on both offensive and defensive strategies—an invaluable experience that strengthens my foundation as an entry-level cybersecurity professional.

Index

List of figures and tables

Introduction

In modern corporate environments, centralized identity and access management is not just a convenience—it's a cornerstone of operational security. At the heart of this structure lies *Active Directory* (AD), a technology developed by Microsoft and adopted by thousands of organizations worldwide to control access, enforce policies, and manage authentication across internal networks.

Its widespread use, however, makes it a prime target for attackers. Misconfigurations, legacy protocols, excessive privileges, or weak password practices are common pitfalls that adversaries can exploit to gain footholds, escalate privileges, and compromise entire domains. In many cases, a single exposed service or a poorly configured user account is enough to trigger a full-scale breach.

This Master's Thesis seeks to bridge the gap between theoretical knowledge and real-world application by constructing a realistic *Active Directory* lab from scratch and subjecting it to a structured security assessment. The environment was intentionally misconfigured to mirror frequent enterprise-level mistakes, allowing the simulation of internal threat scenarios such as lateral movement [6], credential dumping [7], and domain escalation [8].

Rather than relying solely on automated tools, the audit follows a manual, hands-on methodology, reflecting how real attackers think and operate in compromised environments. Every step—from enumeration and exploitation to post-exploitation—is carefully documented to expose hidden risks and demonstrate how overlooked weaknesses can escalate to total domain compromise.

Objectives

The purpose of this project is to analyse the security posture of an *Active Directory*-based infrastructure in a controlled environment, simulating the internal attack surface of a real company. The specific objectives are:

1. **Design and deployment of a realistic** *Active Directory* environment that includes workstations, service accounts, users, and misconfigurations reflective of production networks.
2. **Identification and exploitation of vulnerabilities**, including password reuse, insecure privileges, and attack paths such as *AS-REP Roasting*, *Kerberoasting*, and *DCSync*.
3. **Documentation and proposal of mitigation strategies**, offering technical recommendations to reduce attack surfaces and improve resilience in enterprise environments.

Main body

Tools and Environment

The first phase of this project focused on building a realistic and isolated lab environment to simulate a mid-sized enterprise network based on *Active Directory*. The following virtual machines and technologies were selected:

VirtualBox

VirtualBox served as the primary virtualization platform to deploy and manage all lab machines. It enabled full control over network configurations, snapshots, cloning, and environment isolation. Leveraging internal networking and snapshots, the lab could be restored to a clean state at any time, making it ideal for iterative testing and experimentation.

Windows Server 2022

This system served as the core of the domain infrastructure. It hosts the ***Domain Controller*** (DC) [9] and is responsible for managing *Active Directory* Domain Services (AD DS) [10], ***DNS*** [11], DHCP [12], and Group Policy [13].

Windows Server 2022 was selected due to its broad adoption in enterprise environments and its compatibility with modern AD security features. Despite the availability of a beta version of Windows Server 2025 at the start of the project, it lacked sufficient documentation and community support. Most resources for AD lab environments still reference Windows Server 2016 or 2019. For this reason, the project adopted Windows Server 2022 as the most recent stable release consistent with current enterprise practices.

Windows 10 Enterprise

To reflect a realistic environment, it was necessary to join a workstation to the *Active Directory* domain. Only the Enterprise edition of Windows provides full integration capabilities with domain services.

Kali Linux

Kali Linux was configured as the attacker's machine. It includes a wide range of pre-installed penetration testing tools such as ***BloodHound*** [14], ***Kerbrute*** [15], ***Impacket*** [16], ***CrackMapExec*** [17], and ***Hashcat*** [18].

While alternative distributions such as Parrot OS exist, Kali was chosen for its widespread use and active community support, establishing it as the de facto standard for offensive security assessments.

pfSense

A ***pfSense*** virtual firewall was implemented to simulate a segmented network and control communication between virtual machines. In addition to firewall functionalities, it provides both DHCP and *DNS* services, helping to reproduce common constraints encountered in real-world enterprise environments.

This additional network layer significantly improves simulation fidelity and highlights the relevance of proper network segmentation in corporate infrastructures.

After deploying all components, the next step was to configure *pfSense*. This component plays a critical role in managing internal communications and simulating realistic network constraints within the lab.

Network Configuration with *pfSense*

To replicate realistic network segmentation and traffic control, the open-source firewall *pfSense* 2.7.2 was deployed following the methodology from Ben Heater's *VirtualBox* Lab Guide.

Acting as both gateway and firewall, *pfSense* isolated traffic between virtual machines and established a controlled environment for simulating both network-based attacks and defensive responses.

The configuration, executed within *VirtualBox*, utilized internal adapters to simulate corporate LAN [19] and DMZ [20] structures. Four distinct interfaces were defined to segment machines logically and enable scenarios such as lateral movement, service isolation, and internal privilege escalation.

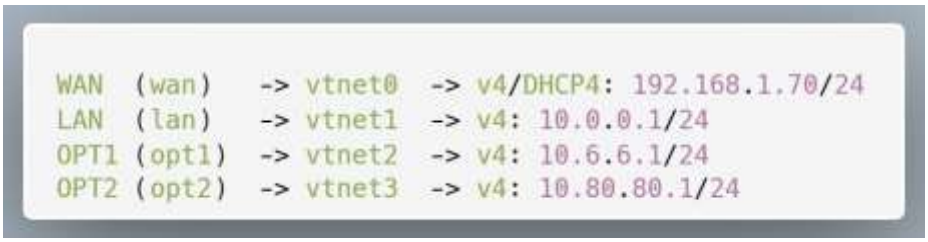


Figure 1 Final pfSense network configuration as seen from the console

PfSense's WebConfigurator was also enabled to simplify firewall and routing administration. Despite being based on external documentation, the setup proved robust and reproducible, forming the backbone of the lab network architecture.

Kali Linux Network Configuration

After finalizing the *pfSense* setup, the **Kali Linux** virtual machine was integrated into the *Cyber-Range-LAN* internal network managed by *pfSense*.

This allowed Kali to operate as an internal adversary, mimicking real-world internal attack scenarios.

Once connected, Kali received a dynamic IP [21] address via *pfSense*'s DHCP service. Proper integration was confirmed by verifying the assigned address *10.0.0.2/24* using the **ip** command, as shown in Figure 2.

With this configuration, the attacker machine gained unrestricted access to simulate enumeration, exploitation, and post-exploitation phases within the lab.

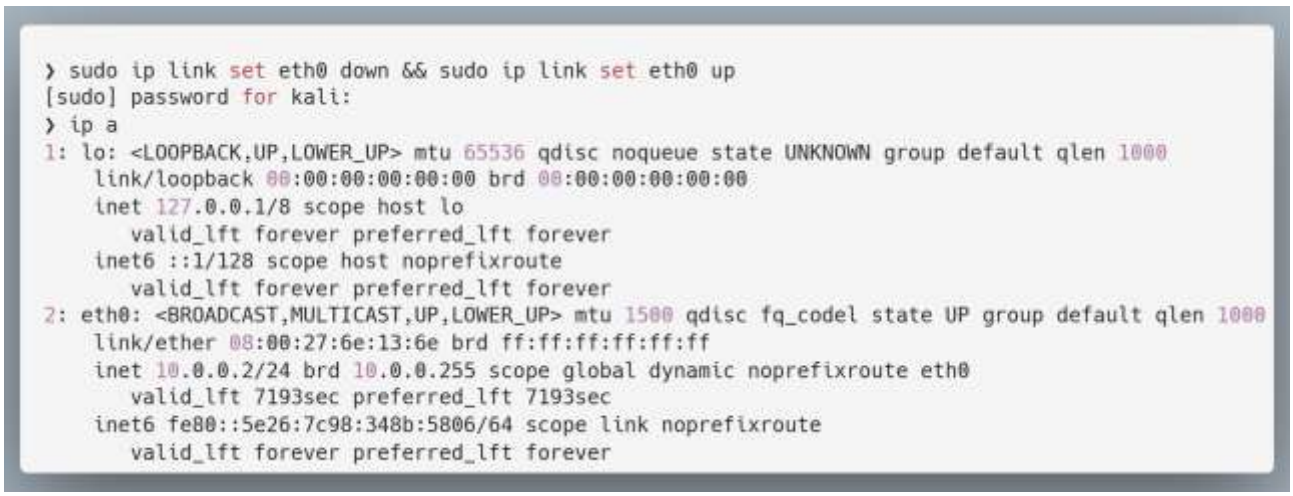


Figure 2 Kali successfully connected to the pfSense LAN with IP address 10.0.0.2

Active Directory Configuration

With the network infrastructure fully operational, the next step involved installing and configuring *Active Directory* Domain Services (AD DS) on the Windows Server machine. This process promotes the server to a *Domain Controller* (DC), which is responsible for managing identities, authentication, and access control across the environment.

Static IP Assignment

Before proceeding with the role installation, a static IP address (*10.80.80.2*) was manually assigned to the server, along with a default gateway and *DNS* pointing to *10.80.80.1* (*pfSense*).

This configuration ensures consistent network identification and stability of the *Domain Controller* during operations, preventing DHCP-induced disruptions.

Figure 3 Static IP configuration of the Domain Controller

Renaming the Server

The server was renamed to DC1 in line with enterprise naming conventions, which typically reflect the role or function of each system.

Meaningful hostnames facilitate administration, monitoring, and incident response, particularly during domain enumeration or forensic analysis.

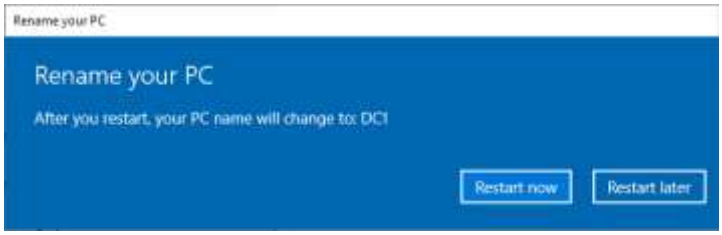


Figure 4 Renaming the Domain Controller to DC1

Installing AD DS and DNS Server Roles

Through the Server Manager interface, the following roles were installed:

- ***Active Directory* Domain Services (AD DS**): Provides centralized identity management and authentication.
- ***DNS* Server:** Ensures domain name resolution and integration with AD services.

Installing both roles simultaneously guarantees proper coordination between *DNS* and AD from the outset.

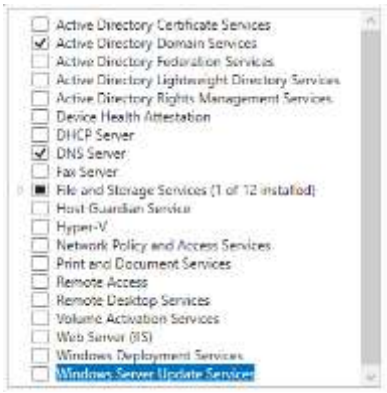


Figure 5 Installing AD DS and DNS Server roles via Server Manager

Promoting to Domain Controller

After installing the required features, the server was promoted to a *Domain Controller* (DC) by creating a new forest with the root domain name *tfm.lab*.

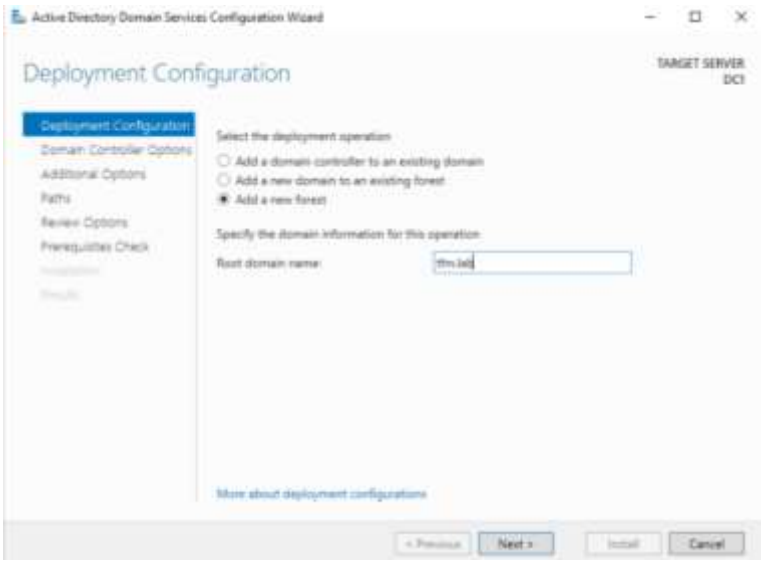


Figure 6 Promote the Domain Controller and change root domain to tfm.lab

A Directory Services Restore Mode (DSRM) [22] password was configured, and default installation values were retained to reflect common enterprise practices – despite their inherent risks.

The system was rebooted after promotion to complete its configuration as the authoritative *Domain Controller* for the lab.

Installing Active Directory Certificate Services

The *Active Directory* Certificate Services (AD CS) [23] role was added to issue digital certificates for services, users, and devices across the domain.

Only the Certification Authority component was installed to simulate common enterprise deployments. This enables scenarios involving smart card authentications and secure encrypted communications (e.g., ***LDAPS*** [24], HTTPS [25]).

Default options were used to mimic misconfigured environments typically observed in real-world audits.

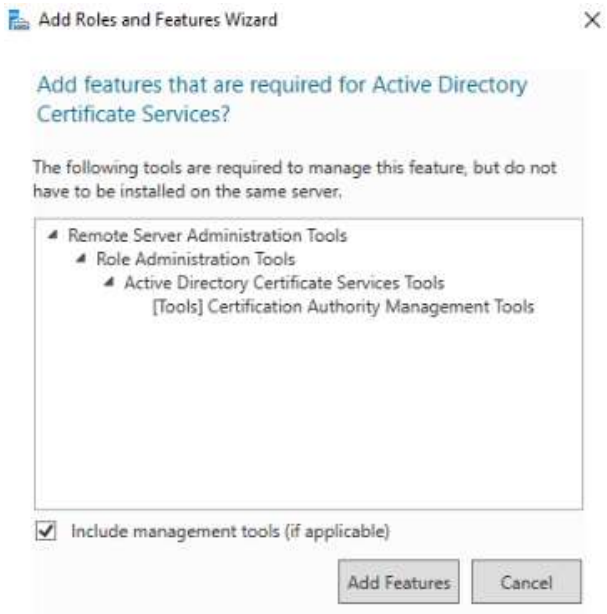


Figure 7 Adding Certificate Service feature to the Domain Controller

Figure 8 Select Certification Authority Role

DNS Forwarder Setup

To enable name resolution for external domains beyond the internal *tfm.lab,* a *DNS* forwarder was configured.

The *Domain Controller* was set to forward unresolved queries to the *pfSense* instance (10.80.80.1), which relays them to external resolvers.

This setup reflects standard enterprise practices, allowing outbound communication for updates, threat simulations (e.g., C2 callbacks), and malware delivery in red team exercises.

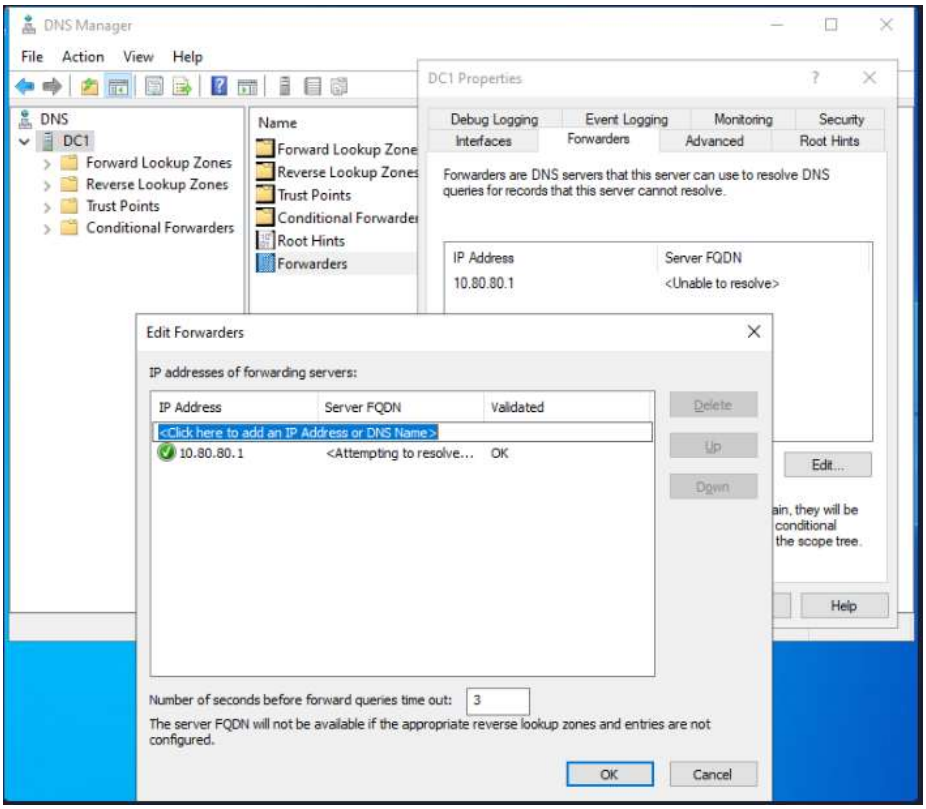


Figure 9 Configuring pfSense as the DNS forwarder in the DNS Manager.

DHCP Server Role Installation

Although enterprise-grade networks often delegate DHCP duties to routers or appliances, in this lab environment the DHCP Server role was installed directly on the *Domain Controller*.

This choice simplifies IP management and enables tighter integration with *DNS* and domain policies.

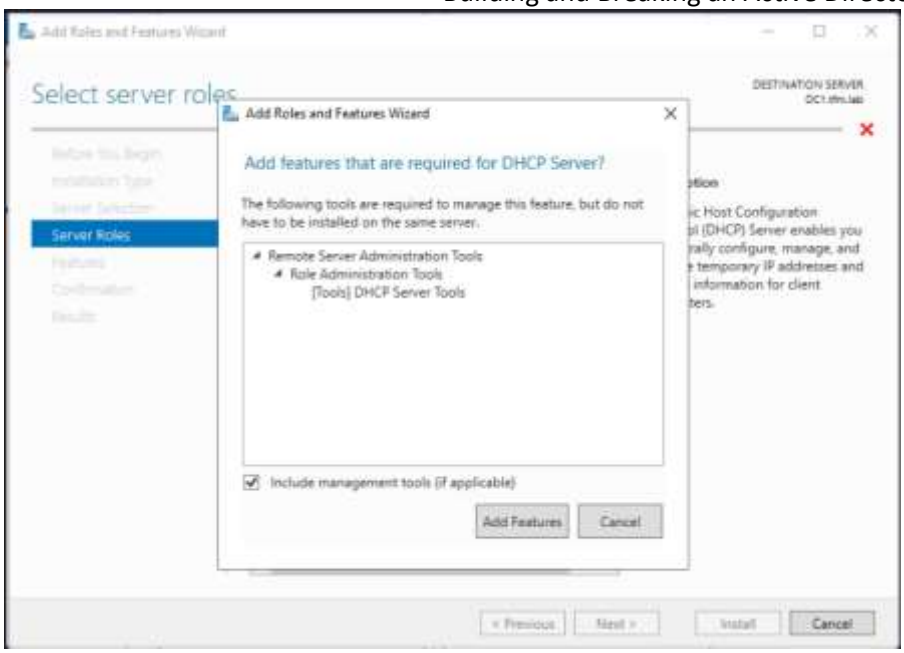


Figure 10 Adding DHCP Features

Following installation, the Complete DHCP Configuration Wizard was launched to finalize the role configuration and apply the required settings.

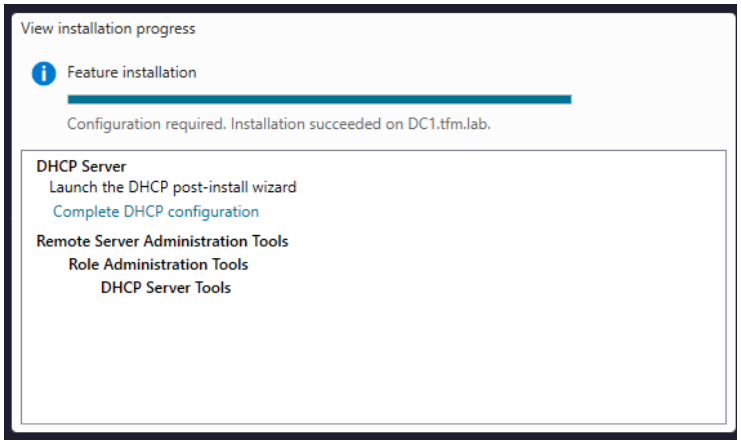


Figure 11 Completing DHCP configuration

DHCP Scope Creation

A new IPv4 DHCP scope named ad_tfm was configured with the following parameters:

- IP Range: 10.80.80.3 – 10.80.80.254 (Excluding the DC and *pfSense*)
- Subnet Mask [26]: 255.255.255.0 (/24)
- Gateway [27]: 10.80.80.1 (*pfSense*)
- *DNS* Server: 10.80.80.2 (*Domain Controller*)
- Domain Name: *tfm.lab*

The lease duration was extended to 365 days to simulate environments where static-like behaviour is preferred or poorly managed.

Such misconfigurations reduce IP turnover, aiding attacker persistence and asset tracking.

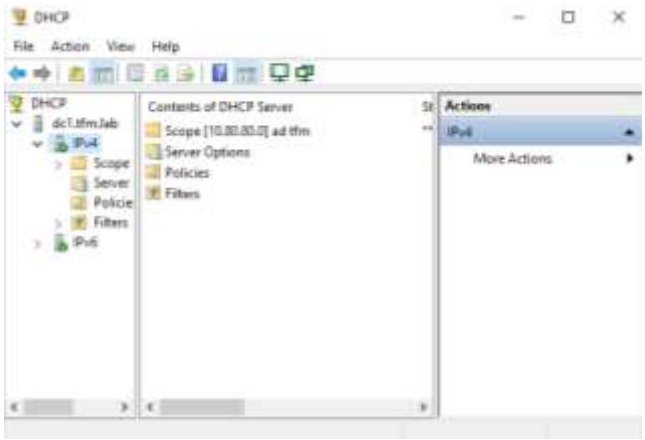


Figure 12 DHCP Scope Configuration for internal domain

Active Directory User and Group Management

Following the configuration of the network infrastructure and core AD services, the next step was to define the identity structure of the domain. This involved creating users, groups, and Organizational Units (OUs) to simulate a realistic enterprise environment with meaningful identity relationships.

Organizational Unit (OU) Creation

Using the *Active* Directory Users and Computers (AD UC) 28 console, a new Organizational Unit named Groups was created. Default security groups were moved from the generic Users container into this OU to promote logical separation between user and group objects — a best practice in enterprise environments.

Maintaining a clear and logically structured OU hierarchy simplifies administration, enhances Group Policy targeting, and is crucial for domain analysis using tools like *BloodHound*.

Creating a Domain Administrator Account

A new privileged user account named ***domain.admin*** was created in the *tfm.lab/Users* container with the following properties:

- **Username**: *domain.admin*
- **Password**: Password1 (for demonstration purposes)

This account was manually added to the Domain Admins group through the Groups OU. This setup emulates common post-deployment practices in which privileged accounts are manually created and added to administrative groups — a behaviour frequently exploited during privilege escalation attacks.

The configuration was verified by logging out and signing in again using the new domain account (*domain.admin@tfm.lab*), confirming successful authentication of domain-admin credentials.

Creating non-privileged Domain Users

Two additional user accounts were created to simulate standard domain users:

- **John Parker** (jparker@*tfm.lab*)
- **Emily Sparrow** (esparrow@*tfm.lab*)

Both users were created within the Users container and configured with standard permissions and default settings. Their passwords were set to never expire, a frequent misconfiguration in real-world enterprise environments where convenience often takes precedence over security, especially in lab scenarios.

Cloning and preparing the Windows 10 Enterprise Template

To streamline the deployment of multiple domain-joined workstation, a Windows 10 Enterprise virtual machines was generalized using the **Sysprep (System Preparation) Tool**. This ensures that each cloned instance received a unique ***Security Identifier (SID)*** [29], preventing duplication issues during domain integration.

The following command was executed via ***PowerShell*** [30]:

```
cd \Windows\System32\
.\Sysprep\sysprep.exe
```

Figure 13 Executing Sysprep to prepare Windows 10 template

The system was configured to enter **OUT-OF-Box Experience (OOBE)** [31] mode with the *Generalize* checkbox enabled and Shutdown selected, stripping hardware-specific information before cloning.

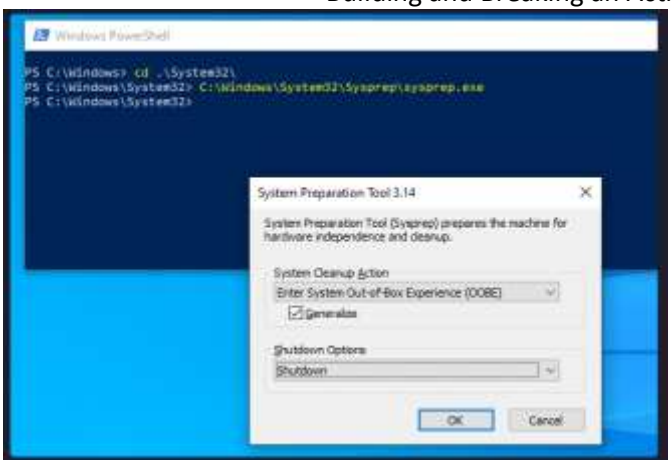


Figure 14 Sysprep generalization process for domain integration

During the cloning process in *VirtualBox*, the option *"Generate new MAC addresses for all network adapters"* was selected to avoid network conflicts.

The resulting image served as the base template for all subsequent workstation deployments.

Figure 15 VirtualBox cloning options in Spanish

Joining the Domain from a Client Machine

The client was manually joined to the domain by modifying system settings under **System Properties → Computer** Name, specifying the domain *tfm.lab* and hostname *Win10Ent1.*

Figure 16 Domain joining process – changing client hostname

Under More**…**, set the primary *DNS* suffix to *tfm.lab*

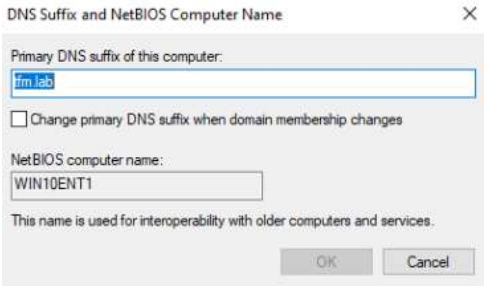


Figure 17 Set Primary DNS suffix

When prompted, provide credentials for a domain account with sufficient privileges. The previously created account *domain.admin@tfm.lab* was used for this purpose.

Figure 18 Login with domain.admin credentials

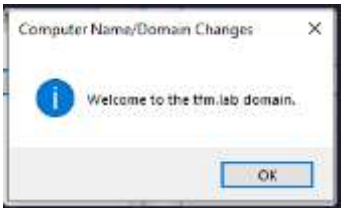


Figure 19 User added to domain

Once the domain join was completed, the machine was rebooted, and successful domain join was confirmed by logging in as user *jparker@tfm.lab* and validating credentials against the Domain.

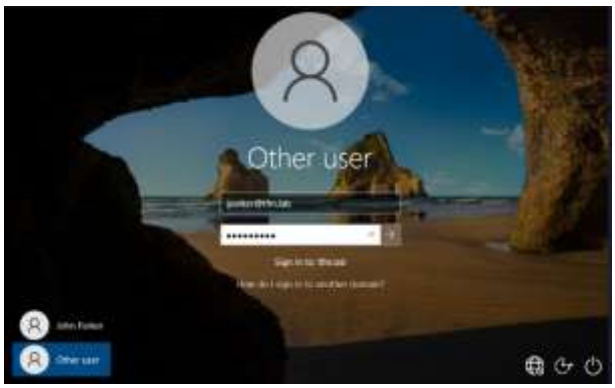


Figure 20 Login with John Parker credentials

Domain Membership Verification

To verify the successful domain integration, diagnostic commands were executed from the command line while logged in as **jparker**:

Figure 21 Domain verification using command-line tools

These commands validated:

- *DNS* resolution and connectivity to *Domain Controller* DC1.
- Membership of the client machine within the *tfm.lab* domain.
- Operational network connectivity confirmed via ICMP response.

These results confirmed full integration into the domain infrastructure, enabling realistic endpoint simulations in subsequent phases.

Figure 22 Confirming domain user context with whoami

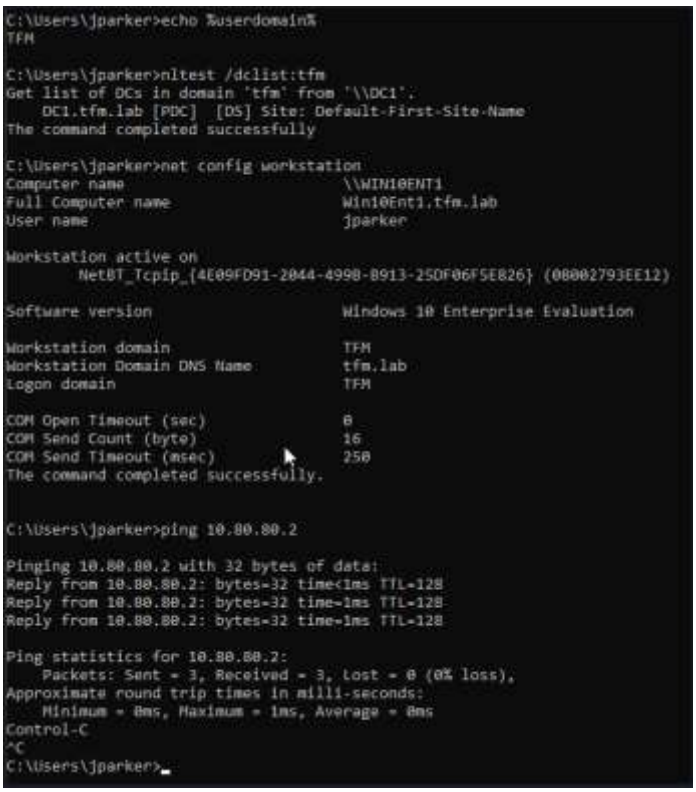


Figure 23 Domain verification commands and outputs

Network Design

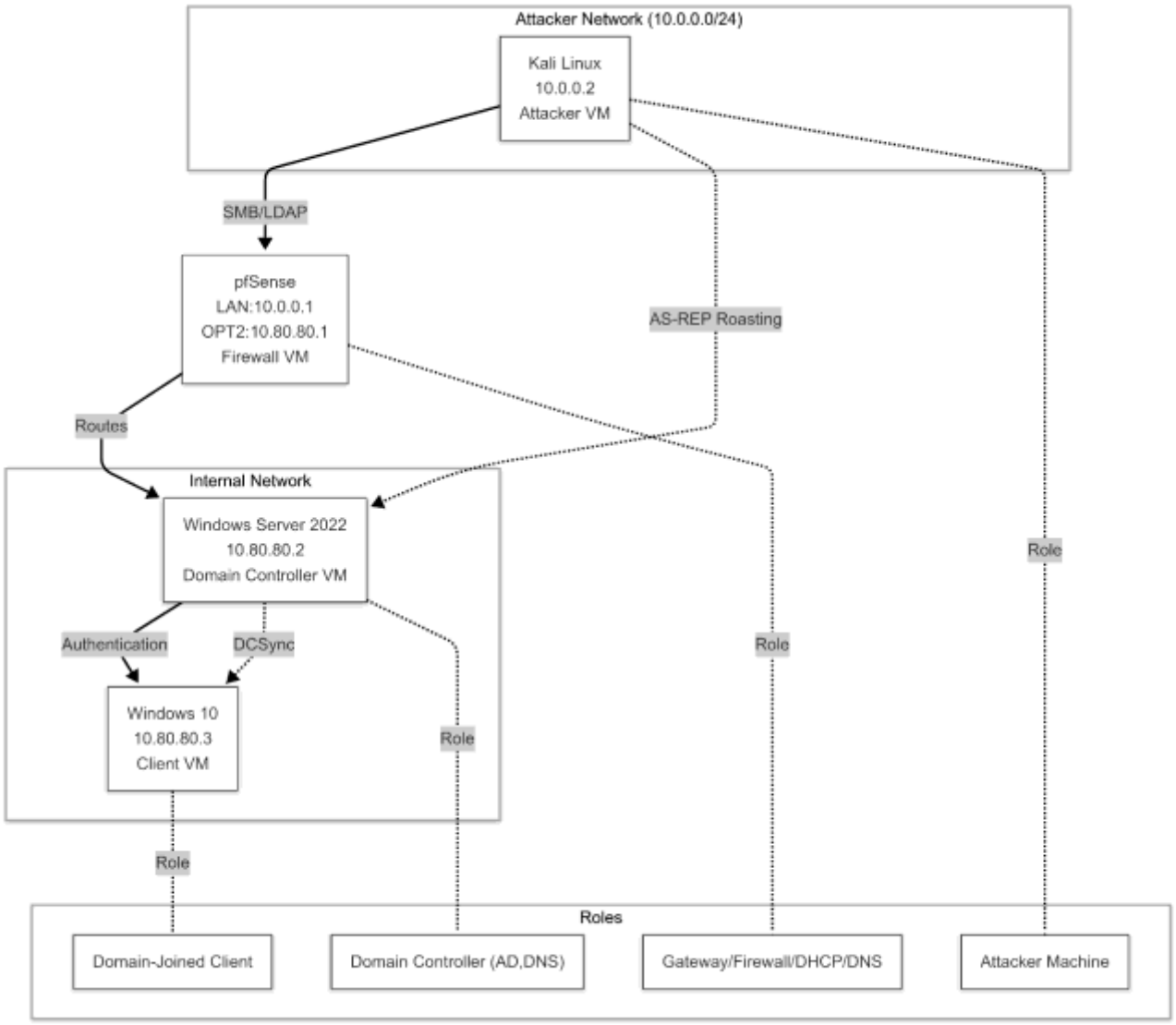


Figure 24 Network design

Vulnerabilities configuration

In order to support the subsequent security analysis and privilege escalation simulations, it is essential that the *Active Directory* environment resembles a realistic enterprise structure. This begins by populating the domain with non-privileged users who simulate a regular employee.

Script-Based User Generation

To simulate a realistic corporate environment, a *PowerShell* script was used to automatically generate non-privileged user accounts. While this setup does not represent a direct vulnerability, it enriches the attack surface and supports realistic simulation of enterprise environments.

The script dynamically creates users with realistic names by selecting randomized combinations from a predefined list. Each user receives a randomly generated, secure password, and is automatically enabled within the domain.

- Configuration and implementation

  The following *PowerShell* script was executed from a domain-joined administrative session:

```powershell
# --- Initial Configuration ---
Import-Module ActiveDirectory
$Global:Domain = "lab.local" # <-- Change this if your domain is different
$Global:HumansNames = @("Alice", "Bob", "Charlie", "Diana", "Eva", "Frank", "George", "Hannah")

function VulnAD-GetRandom {
    Param([array]$InputList)
    return $InputList | Get-Random
}

function VulnAD-AddADUser {
    Param([int]$limit = 5)
    Add-Type -AssemblyName System.Web

    for ($i = 1; $i -le $limit; $i++) {
        $firstname = VulnAD-GetRandom -InputList $Global:HumansNames
        $lastname = VulnAD-GetRandom -InputList $Global:HumansNames
        $SamAccountName = ('{0}.{1}' -f $firstname, $lastname).ToLower()
        $principalname = "$SamAccountName@$($Global:Domain)"
        $generated_password =
            ([System.Web.Security.Membership]::GeneratePassword(8,2))

        Write-Host "Creating user: $SamAccountName with password: $generated_password"

        Try {
            New-ADUser -Name "$firstname $lastname" -GivenName $firstname `
            -Surname $lastname -SamAccountName $SamAccountName -UserPrincipalName `
            $principalname -AccountPassword (ConvertTo-SecureString `
            $generated_password -AsPlainText -Force) -PassThru | Enable-ADAccount
        } Catch {
            Write-Host ■ Failed to create $SamAccountName
        }
    }
}

# --- Run the function to create users ---
VulnAD-AddADUser -limit 5
```

Figure 25 Script-Based User Generation

Each execution creates a predefined number of users (limit), generating names, login credentials and passwords automatically. Sample outputs show successful creation of accounts such as:

```
PS C:\Users\domain.admin\Desktop> dir


    Directory: C:\Users\domain.admin\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/22/2025  11:05 AM           1214 users.ps1


PS C:\Users\domain.admin\Desktop> Set-ExecutionPolicy RemoteSigned -Force
PS C:\Users\domain.admin\Desktop> .\users.ps1
Creating eva.steve User with password [y#N{y($
Creating eva.frank User with password RL#M{3En
Creating frank.george User with password hH{0?Yu6
Creating steve.steve User with password !3Bb19)5
Creating charlie.george User with password mH.[mz-H
PS C:\Users\domain.admin\Desktop>
```

Figure 26 Execute User Generation Script

Once created the accounts were visible through both the *Active Directory* Users and Computers (AD UC) console and *PowerShell* queries such as:

```
PS C:\Users\domain.admin\Desktop> Get-ADUser -Filter * | Select-Object SamAccountName, Name

SamAccountName Name
-------------- ----
Administrator  Administrator
Guest          Guest
krbtgt         krbtgt
domain.admin   Domain Admin
jparker        John Parker
esparrow       Emily Sparrow
eva.steve      Eva Steve
eva.frank      Eva Frank
frank.george   Frank George
steve.steve    Steve Steve
charlie.george Charlie George
```

Figure 27 List of AD users

- <u>Risk and exploitation</u>
  - Risk level: None
  - Justification: No vulnerability is introduced by this setup. However, the presence of multiple standard users is a prerequisite for simulating real attack techniques such as:
    - Enumeration [32]
    - *Password Spraying* [33]
    - *Kerberoasting*
    - Pass-The-Ticket [34] attacks
    - Relationship mapping with tools like *BloodHound*
  - Risk level: None

This setup reflects enterprise-scale environments where hundreds or thousands of identities must be managed.

- Mitigation and recommendations

  Although this setup is intentional in the lab, in real environments the following best practices should be observed:

  - **Log and monitor** any bulk user creation activities
  - Enforce **password complexity policies** beyond 8 characters
  - Assign users to the **least privilege groups** only (avoid unnecessary permissions)
  - Apply automated lifecycle management for account expiration and deprovisioning.

AS-REP Roasting Vulnerability

AS-REP Roasting is a known Kerberos [35]-based attack that targets accounts not requiring preauthentication. When this setting is disabled, an attacker can request a Kerberos Authentication Service Response (AS-REP) for that user without needing valid credentials. The resulting response, which contains an encrypted ***Ticket Granting Ticket (TGT)*** [36], can be **brute-forced offline** to recover the user's plaintext password.

This misconfiguration is frequently observed in environments with legacy systems or limited awareness of Kerberos security features.

- Configuration and implementation

  To intentionally expose the vulnerability, the following configurations were applied to the ***charlie.george*** user account:

```
$User = "charlie.george"
$Password = "Password123!"

# Set a weak password
Set-ADAccountPassword -Identity $User -Reset -NewPassword (ConvertTo-
SecureString $Password -AsPlainText -Force)

# Disable preauthentication for Kerberos
Set-ADAccountControl -Identity $User -DoesNotRequirePreAuth 1

Write-Host "✅ $User is vulnerable for AS-REP Roasting with password $Password"
```

Figure 28 Disabling pre-authentication for Charlie George (AS-REP Roasting)

To ensure the account was active and usable from external tools (e.g., Kali Linux), the user was explicitly enabled:

```
> Enable-ADAccount -Identity charlie.george
> Get-ADUser charlie.george -Properties Enabled | Select SamAccountName, Enabled
```

Figure 29 PowerShell command to enable the User

```
PS C:\Users\domain.admin\Desktop> dir


    Directory: C:\Users\domain.admin\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/22/2025  11:22 AM            319 asrep_charlie.ps1
-a----         3/22/2025  11:08 AM           1214 users.ps1


PS C:\Users\domain.admin\Desktop> Set-ExecutionPolicy RemoteSigned -Force
PS C:\Users\domain.admin\Desktop> .\asrep_charlie.ps1
The user is vulnerable to AS-REP Roasting with the password Password123!
PS C:\Users\domain.admin\Desktop>
```

Figure 30 PowerShell script enabling vulnerable user account

- Risk and exploitation
  - Severity: High
  - Attack Vector: Remote (no credential required)
  - Affected Component: Kerberos preauthentication
  - CVE [37] Reference: [CVE-2014-6324] (general Kerberos abuse)

  An attacker connected to the internal network can leverage tools such as:
  - *Impacket´s* ***GetNPUsers.py*** [38]
  - ***Rubeus*** [39]

  to extract and crack the *TGT* offline. If successful, the attacker obtains the cleartext password and can impersonate the user across the domain.
- Mitigation and recommendations
  - **Audit** all users for the *DoesNotRequirePreAuth* flag using *PowerShell* or *BloodHound*
  - **Enforce preauthentication** via *Group Policy* ("*Domain Account: Require Kerberos preauthentication*")
  - **Use complex, rotating passwords** for service accounts

- Monitor unusual **AS-REQ** [40] **traffic** from the same source, which could indicate brute-force attempts.

- Realism and Justification
  - Realism: High. This is still observed in many networks due to:
    - Compatibility with legacy systems
    - Manual provisioning errors
    - Lack of awareness about Kerberos risks

Kerberoasting Vulnerability

Kerberoasting is a post-authentication attack technique targeting service accounts with exposed ***Service Principal Names (SPNs)*** [41]. Authenticated attackers can extract **Ticket Granting Service (*TGS*)** [42] ticket for the *SPN*, extract the encrypted service ticket, and attempt to crack it offline. If the password is weak, this may lead to full account compromise. The attack combines credential abuse with excessive privilege assignments, posing a high lateral movement risk.

This scenario replicates a common misconfiguration in real-world environments, where service accounts use static or weak credentials, often combined with excessive privileges, making them valuable targets for lateral movement and privilege escalation.

- Configuration and implementation

To simulate the vulnerability, a service account named **svc-backup** was created with the following insecure properties:

```
$Username = "svc-backup"
$Password = "Password123!"
$Domain = "tfm.lab"
$SPN = "HTTP/backup.$Domain"
$Group = "Backup Operators"

# Create the service account with SPN
New-ADUser -Name $Username `
    -SamAccountName $Username `
    -ServicePrincipalNames $SPN `
    -UserPrincipalName "$Username@$Domain" `
    -AccountPassword (ConvertTo-SecureString $Password -AsPlainText -Force) `
    -Enabled $true `
    -PassThru | Enable-ADAccount

# Optional: Also vulnerable to AS-REP Roasting (adds another vector)
Set-ADAccountControl -Identity $Username -DoesNotRequirePreAuth $true

# Assign to privileged group (adds lateral movement opportunity)
Add-ADGroupMember -Identity $Group -Members $Username
```

Figure 31 Kerberoasting Vulnerability Script

This setup introduces multiple attack vectors simultaneously:

- *SPN* exposure for *Kerberoasting*
- Preauthentication disabled for *AS-REP Roasting*
- Membership in a privileged group (*Backup Operators*)

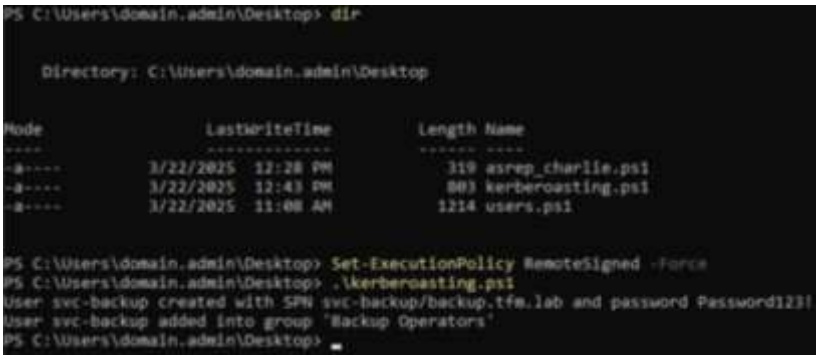


Figure 32 Execution of Kerberoasting misconfiguration script

- Risk and exploitation
  - Severity: Critical
  - Attack Vector: Internal (authenticated user required)
  - Affected Component: Kerberos Ticket Granting Service (*TGS*)
  - CVE References: [CVE-2020-17049] (related to *SPN* abuse)

  Once authenticated in the domain, an attacker can use tools such as:
  - *Impacket*'s *GetUserSPNs.py*
  - *Rubeus*

  to request and export service tickets for *SPN*s like *HTTP/backup.tfm.lab.* These tickets can then be **brute-forced offline** to recover the cleartext password of the service account.

  If the account holds privileged group membership, this can lead of **escalation of privileges** or even **domain compromise**.
- Mitigation and recommendations
  - Use **long and complex passwords** for all service accounts
  - Where possible, replace user-based *SPN*s with ***Group Managed Service Accounts (gMSA)*** [43]
  - Apply **fine-grained password policies** to service accounts
  - **Monitor *TGS* ticket requests**, especially those associated with high-privilege *SPN*s.
  - Audit *SPN* assignments and group memberships regularly

- Realism and Justification
  - Realism: Very high

    This vulnerability is frequently observed in enterprise environments, especially where:
    - Service accounts are not managed under strict policies
    - *SPN*s are misconfigured or overused
    - Legacy applications require static credentials

  This setup combines several insecure practices to emulate a layered attack surface commonly found in large, mismanaged AD environments.

DNSAdmins Privilege Abuse

The ***DNSAdmin*s** group in *Active Directory* is often overlooked, yet it holds powerful privileges. Members of this group can configure the *DNS* Server, including loading **arbitrary DLLs** [44] via the registry. Since the *DNS* service typically runs as NT AUTHORITY\SYSTEM, this can lead to privilege escalation.

This vulnerability does not rely on exploiting a flaw, but rather on abusing legitimate administrative capabilities. It is a clear example of privilege escalation via misconfigured delegation and poor group hygiene.

- Configuration and implementation

  To simulate this risk, a new user account named *Dnsadmin* was created with the following properties:

```
$User = "dnsadmin"
$Password = "Password123!"
$Domain = "tfm.lab"

# Create user
New-ADUser -Name $User `
    -SamAccountName $User `
    -UserPrincipalName "$User@$Domain" `
    -AccountPassword (ConvertTo-SecureString $Password -AsPlainText -Force) `
    -Enabled $true |
Enable-ADAccount

# Add user to DnsAdmins group
Add-ADGroupMember -Identity "DnsAdmins" -Members $User
```

Figure 33 Script for Dnsadmin

```
PS C:\Users\domain.admin\Desktop> dir


    Directory: C:\Users\domain.admin\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/22/2025  12:28 PM            319 asrep_charlie.ps1
-a----         3/22/2025   1:11 PM            671 dnsadmins.ps1
-a----         3/22/2025  12:43 PM            803 kerberoasting.ps1
-a----         3/22/2025  11:08 AM           1214 users.ps1


PS C:\Users\domain.admin\Desktop> Set-ExecutionPolicy RemoteSigned -Force
PS C:\Users\domain.admin\Desktop> .\dnsadmins.ps1
User dnsadmin created with password Password123!
User dnsadmin added into 'DnsAdmins'
PS C:\Users\domain.admin\Desktop>
```

Figure 34 PowerShell Script Execution

After executing the corresponding *PowerShell* script (*Dnsadmin*s.ps1), the console confirmed:

- Account creation
- Addition to the *DNSAdmin*s group

This permission allows the user to configure critical *DNS* server parameters, including modify the ServerLevelPluginDll registry key, which can be used to load malicious DLLs under SYSTEM context.

- Risk and exploitation
  - Severity: Critical
  - Attack Vector: Internal (group membership required)
  - Affected Component: Windows *DNS* Server
  - Techniques Involved: Registry tampering, DLL injection, privilege escalation

An attacker who compromises this account can:

1. Write a malicious DLL.
2. Modify the registry key: *HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters\ServerLevelPluginDll.*
3. Restart the *DNS* service.
4. Achieve SYSTEM-level code execution.

If the *DNS* Server runs on a *Domain Controller*, this may escalate to full domain compromise.

- Mitigation and recommendations
    - Restrict *DNSAdmin*s membership to trusted, audited personnel only.
    - Separate *DNS* services from *Domain Controller*s where possible.
    - Monitor registry keys and group membership changes (e.g., with Sysmon [45] + SIEM [46]).
    - Implement LAPS [47], tiered admin models, and Just Enough Administration (JEA) [48].
- Realism and Justification
    - Realism: High

      This vulnerability reflects real scenarios where:
    - *DNS* is installed on DCs for convenience.
    - *DNS* administration is delegated carelessly.
    - Group memberships are not monitored or audited.

  Reasons this misconfiguration exists:
    - Underestimation of *DNSAdmin*s power.
    - Lack of proper privilege separation.
    - Operational shortcuts in hybrid or legacy infrastructures.

DCSync Privilege Abuse

DCSync attacks simulate *Domain Controller* behaviour by leveraging replication privileges to exfiltrate credential data stealthily. Any account with specific replication rights can impersonate a *Domain Controller* and request sensitive data, such as *NTLM* password hashes, including those from ***KRBTGT*** [49], Domain Admins, and other privileged users.

This simulation demonstrates a misconfigured access control scenario, in which a regular user account is granted *DCSync* permissions through extended rights in the domain's *ACL* [50].

- Configuration and implementation

  To replicate this misconfiguration:
    1. A new user named **svc-sync** was created.
    2. The user was added to a group called **SyncOperators**.
    3. The group was granted ExtendedRights via three critical GUIDs:
        - DS-Replication-Get-Changes
        - DS-Replication-Get-Changes-All
        - DS-Replication-Get-Changes-In-Filtered-Set

```powershell
$sidString = (Get-ADUser svc-sync).SID.Value
$identity = New-Object System.Security.Principal.SecurityIdentifier($sidString)

# Assign DCSync rights
foreach ($guid in @(
    [Guid]"1131f6aa-9c07-11d1-f79f-00c04fc2dcd2",
    [Guid]"1131f6ad-9c07-11d1-f79f-00c04fc2dcd2",
    [Guid]"1131f6ae-9c07-11d1-f79f-00c04fc2dcd2")) {

    $ace = New-Object System.DirectoryServices.ActiveDirectoryAccessRule (
        $identity, "ExtendedRight", "Allow", $guid, "All"
    )
    $acl.AddAccessRule($ace)
}

# Apply modified ACL
$domainObject.psbase.ObjectSecurity = $acl
$domainObject.psbase.CommitChanges()
```

Figure 35 Granting DCSync permissions via PowerShell script

```
PS C:\Users\domain.admin\Desktop> dir


    Directory: C:\Users\domain.admin\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         3/22/2025  12:28 PM            319 asrep_charlie.ps1
-a----         3/22/2025   5:51 AM           2088 dcsync.ps1
-a----         3/22/2025   1:11 PM            671 dnsadmins.ps1
-a----         3/22/2025  12:43 PM            803 kerberoasting.ps1
-a----         3/22/2025  11:08 AM           1214 users.ps1


PS C:\Users\domain.admin\Desktop> Set-ExecutionPolicy RemoteSigned -Force
PS C:\Users\domain.admin\Desktop> .\dcsync.ps1
[X] The group SyncOperators already exists or there was an error
The specified group already exists
User svc-sync has been created  with password Password123!
Added svc-sync into  group SyncOperators
DCSync permissions applied to group SyncOperators
PS C:\Users\domain.admin\Desktop>
```

Figure 36 Successful execution of DCSync configuration script

The output confirmed that:

- The user svc-sync was created with password *Password123!*
- Added to SyncOperators
- *DCSync* rights were applied successfully

- <u>Risk and exploitation</u>
  - Severity: Critical
  - Attack Vector: Internal (requires directory write access or ACL misconfiguration)
  - Technique: Credential Dumping via Replication
  - Tools: *Mimikatz* [51], *secretsdump* [52], SharpHound [53]

If exploited, this grants:

- Stealthy, persistent access to domain credentials.
- Ability to exfiltrate hashes of Domain Admins.
- *Golden Ticket* [54] generation for long-term persistence.

- Mitigation and recommendations

  To prevent or detect this scenario:

  - Never assign replication rights to non-DC accounts.
  - Regularly audit ACLs using tools like *BloodHound*, PingCastle [55], or PowerView [56].
  - Use tiered administration models and segment roles and privileges.
  - Monitor for *DCSync* behaviour patterns in event logs (Event ID 4662).

- Realism and Justification

  - Realism: Very High

    DCSync is one of the most abused attack techniques in red teaming and Advanced Persistent Threat (APT) [57] scenarios due to its stealth and power.

  Reasons this misconfiguration appears in real environments:

  - Delegation of replication rights to service accounts.
  - Lack of knowledge about ACL inheritance and GUIDs [58].
  - No periodic audits of extended rights on domain objects.

- Consequence if Exploited

  - Immediate Domain Admin compromise.
  - Extraction of *KRBTGT* hash → Golden Ticket attacks.
  - Long-term persistence and undetectable lateral movement.

  This vulnerability was intentionally configured to demonstrate how dangerous ACL misconfigurations can be, especially when overlooked during delegation of replication tasks.

Remote Code Execution via Evil-WinRM

Evil-WinRM [59] (Evil Windows Remote Management) is a powerful post-exploitation tool used to gain remote interactive shell access on Windows machines via the *WinRM* (Windows Remote Management) protocol.

While *WinRM* is designed for legitimate remote administration, enabling it without proper restrictions can allow attackers with valid credentials to execute arbitrary code remotely with administrative privileges.

- Configuration and implementation

  To simulate this vulnerability, the following configuration was applied:

```
# Enable WinRM to allow remote connections
Write-Host "✅ Enabling WinRM..."

# Enable the WinRM service
Enable-PSRemoting -Force

# Ensure the service starts automatically
Set-Service WinRM -StartupType Automatic

# Add a firewall rule if it's active
Try {
    netsh advfirewall firewall add rule name="WinRM" dir=in action=allow protocol=TCP localport=5985
    Write-Host "✅ Firewall rule for WinRM added"
} Catch {
    Write-Host "⚠️ Error adding the firewall rule or it already exists"
}

# Confirm that the service is running
$status = Get-Service WinRM | Select-Object -ExpandProperty Status
Write-Host "📊 WinRM service status: $status"
```

Figure 37 WinRM Script

1. *WinRM* was deliberately enabled on the *Domain Controller* and a firewall rule was added to allow inbound traffic on TCP port 5985 (HTTP).
2. Service status was verified
3. The *Dnsadmin* domain user was added to the local ***Administrator*** group on the *Domain Controller*

  This misconfiguration enables the *Dnsadmin* user, added to the local *Administrator*s group, to establish remote shell sessions with full SYSTEM privileges.

- Risk and exploitation
  - Severity: Critical
  - Attack Vector: Internal (requires valid credentials)
  - Technique: Remote Code Execution (RCE) [60] over *WinRM*
  - Tools: *Evil-WinRM*, *PowerShell*, *CrackMapExec*

  If exploited, an attacker can:
  - Gain full shell access to the *Domain Controller*.
  - Execute arbitrary code as SYSTEM.
  - Dump credentials, disable defences, or establish persistence mechanisms.
- Mitigation and recommendations

  To prevent or limit the impact of this vulnerability:
  - Disable *WinRM* if not explicitly required.
  - Use Just Enough Administration (JEA) and Role-Based Access Control (RBAC) [61].

- Monitor changes to the local *Administrator*s group and *WinRM* session activity.
- Restrict *WinRM* through Group Policies and firewall rules.

- Realism and justification
  - Realism: High

    This scenario is common in real-world environments where:
  - Domain users are temporarily granted local admin rights and never removed.
  - *WinRM* is enabled by default but lacks proper access control.
  - There is little to no visibility on remote *PowerShell* activity or privilege assignment.

  Why it exists:
  - Operational convenience in IT teams.
  - Misconfigured Group Policies.
  - Inadequate privilege delegation practices.
- Consequence if Exploited
  - Complete remote control over the *Domain Controller*.
  - Execution of any script or binary as *Administrator*.
  - Credential harvesting, disabling of defences, and lateral movement.
  - Can serve as an initial pivot for broader domain compromise.

  This configuration was deliberately introduced to highlight how a standard domain user — if granted inappropriate privileges and exposed services — can escalate to full system control. It demonstrates the real danger of privilege abuse + exposed services in enterprise environments.

Default Passwords in Account Descriptions

To emulate poor operational practices found in real-world enterprise environments, several domain users were created using weak or common passwords that were explicitly written into the *Active Directory* description attribute.

This presents a severe security risk, as most AD enumeration tools retrieve description fields by default and are readable by any authenticated domain user, making it trivial for an attacker to discover credentials without requiring elevated permissions.

- Configuration and implementation

  The following *PowerShell* script was used to create three user accounts with predictable, insecure passwords. These passwords were also written directly in the Description field:

```powershell
# Load the Active Directory module
Import-Module ActiveDirectory

# Users to be created
$users = @("Daniel","Steve","Larry")
$domain = "tfm.lab"

# Weak or commonly used passwords
$passwords = @(
    "Winter2024!",
    "Welcome123!",
    "SuperM4n"
)

# Create each user with password listed in description field
for ($i = 0; $i -lt $users.Count; $i++) {
    $user = $users[$i]
    $pwd = $passwords[$i]

    Try {
        New-ADUser -Name $user `
            -SamAccountName $user `
            -UserPrincipalName "$user@$domain" `
            -AccountPassword (ConvertTo-SecureString $pwd -AsPlainText -Force) `
            -Enabled $true `
            -Description "Default password: $pwd" `
            -PassThru |
        Enable-ADAccount

        Write-Host "✅ $user created with password '$pwd' written in the description"
    } Catch {
        Write-Host "⚠️ Failed to create $user: $($_.Exception.Message)"
    }
}
```

Figure 38 Creating insecure AD user descriptions via PowerShell

This setup allows any authenticated user to query AD and extract passwords from account descriptions using tools such as:

  - PowerView: Get-DomainUser -Properties description
  - AD Explorer
  - Native *PowerShell*: Get-ADUser -Filter * -Properties Description

- Risk and exploitation
  - Severity: High
  - Attack Vector: Internal (requires basic authenticated user)
  - Technique: Credential Exposure via *LDAP* Enumeration
  - Tools: PowerView, *BloodHound*, ADExplorer

Once an attacker gains access to any domain account, they can enumerate user objects and read their descriptions, allowing them to:

- Instantly retrieve plaintext passwords.
- Attempt lateral movement to more privileged accounts.

- Mitigation and recommendations
  - Audit descriptions and other user attributes for sensitive keywords (e.g., “password”, “admin”, etc.).
  - Never store credentials in AD attributes (including info, comment, or description).
  - Enforce strong password policies and automatic expiration via *GPOs* [62].
  - Apply access control lists (ACLs) to limit who can read user properties in AD.
- Realism and justification
  - Realism: Very High

    This misconfiguration is commonly found in environments where:
  - Environments are quickly promoted from staging to production without cleanup.
  - *Administrator*s store onboarding or temporary credentials in AD fields for convenience.
  - There is low security awareness regarding *LDAP* visibility.

  Why it exists:
  - Lack of proper operational standards.
  - No audit procedures for attribute misuse.
  - Misuse of AD as a credential store.
- Consequence if Exploited
  - Instant compromise of accounts through no-effort enumeration.
  - Privilege escalation if affected users belong to high-privilege groups.
  - Can serve as the initial foothold for further post-exploitation activities.

  This scenario demonstrates how minor misconfigurations in AD hygiene can lead to severe consequences, especially when combined with lateral movement techniques or automation tools.

Anonymous LDAP Queries

To simulate weak configurations frequently observed in legacy or poorly maintained environments, anonymous LDAP access was explicitly enabled on the Domain Controller by modifying the system registry.

This change allows unauthenticated users to query the directory using tools like *LDAPsearch* [63]*, nmap* [64]*,* or *AD Explorer* — without valid domain credentials. This exposes the domain to unauthenticated reconnaissance, enabling attackers to retrieve structural data such as OUs, users, and groups.

- Configuration and implementation

  The following *PowerShell* script was executed to enable anonymous *LDAP* access by modifying the NTDS [65] service registry key:

```powershell
# Enable anonymous LDAP queries on a Domain Controller

$regPath = "HKLM:\SYSTEM\CurrentControlSet\Services\NTDS\Parameters"

Try {
    Set-ItemProperty -Path $regPath -Name "Allow Anonymous Access" -Value 1 -Type DWord
    Write-Host "✅ Anonymous LDAP access enabled (requires NTDS service restart)"
} Catch {
    Write-Host "❌ Error modifying registry"
    Write-Host $_.Exception.Message
}
```

Figure 39 PowerShell script to enable anonymous LDAP access via registry modification

After this modification, a restart of the NTDS service or reboot of the *Domain Controller* is required for the change to take effect. Once applied, LDAP queries from unauthenticated systems are permitted, exposing directory metadata to potential attackers.

- Risk and exploitation
  - Severity: High
  - Attack Vector: External (Unauthenticated network access)
  - Technique: Directory Enumeration via *LDAP*
  - Affected Component: NTDS Service on *Domain Controller*

  This misconfiguration permits unauthenticated enumeration of:
  - Domain users and groups
  - Organizational Units (OUs)
  - Group memberships
  - Service accounts

- Tools commonly used:
  - *LDAP*search
  - AD Explorer
  - Custom *PowerShell* scripts
  - nmap --script *LDAP**
- Mitigation and recommendations
  - Disable Anonymous Bindings:

    Set Allow Anonymous Access back to 0 and restart the NTDS service.
  - Enforce *LDAP* Signing:

    Use Group Policy to reject unsigned *LDAP* requests by setting

  - Audit and Monitor:

    Regularly scan registry keys for insecure configurations using *PowerShell* baselines or centralized configuration management tools.
  - Tools for detection:
    - PingCastle
    - AD ACL Scanner
    - Custom scripts with Get-ItemProperty [66]
- Realism and justification
  - Realism: Moderate to High

    While less common in modern environments, this configuration is still found in:
    - Upgraded legacy domains (e.g., Windows Server 2003 → 2016+)
    - Hybrid environments where old applications still depend on anonymous *LDAP*
    - Misconfigured security baselines during rushed deployments

  Why it exists:
  - Backward compatibility requirements
  - Registry misconfiguration by untrained *Administrator*s
  - Lack of GPO enforcement
- Consequences if exploited
  - Unauthorized users can gain read-only insight into the domain's structure.
  - Valid usernames and groups can be harvested, aiding:
    - Password spraying attacks
    - Targeted phishing or social engineering

- Credential brute-force and lateral movement

This attack path provides a stealthy, low-noise vector that avoids authentication logs, making it highly desirable for threat actors in early attack stages.

Public SMB Share with Full Access

To simulate risky configurations commonly observed in smaller or poorly secured environments. An unauthenticated ***SMB*** share was configured with *Everyone: Full Control*, replicating a frequent misconfiguration observed in small and medium-sized business (*SMB*s).

The share \\DC1***PublicShare*** was configured with:

- Full **NTFS** [67] permissions for the Everyone group
- Unrestricted *SMB* [68] access, meaning any user (authenticated or not) can read, write, or modify its contents.

- Configuration and implementation

The following *PowerShell* script was used to configure the insecure share:

```
# Create a shared folder accessible by all users (including unauthenticated ones)

$shareName = "PublicShare"
$path = "C:\PublicShare"

# Create folder if it does not exist
if (-not (Test-Path $path)) {
    New-Item -Path $path -ItemType Directory -Force
    Write-Host "📁 Folder created: $path"
}

# Grant NTFS permissions to Everyone
icacls $path /grant "Everyone:(OI)(CI)F" /T
Write-Host "✅ NTFS permissions granted to Everyone"

# Create the network share with full access
New-SmbShare -Name $shareName -Path $path -FullAccess Everyone
Write-Host "✅ Shared folder as '\\$env:COMPUTERNAME\$shareName' with full access"
```

Figure 40 Creating insecure public SMB share configuration

Once executed, the system exposes C:*PublicShare* as a writable network share accessible without restrictions, representing a high-risk configuration.

- Risk and exploitation
  - Severity: High
  - Attack Vector: Internal and potentially External (if guest access is enabled)
  - Technique: File planting, script manipulation, malware staging
  - Affected Component: *SMB* File Sharing Service

  Possible attack vectors:
  - Dropping and executing malicious scripts or DLLs
  - Overwriting legitimate files for privilege escalation
  - Credential theft via login scripts or hijacked config files
  - Data exfiltration through a public export path
- Mitigation and recommendations
  - Audit *SMB* Shares

    Use tools like PowerView, SharpShares [69], or ShareEnum [70] to identify misconfigured shares.
  - Apply Least Privilege Access

    Remove Everyone and replace with tightly scoped groups (e.g., Authenticated Users - Read Only).
  - Disable Guest/Anonymous Access

    Via Group Policy:

  - Enable Logging and Monitoring

    Use advanced audit policies to track file access, modification, and unexpected connections to shared folders.
- Realism and justification
  - Realism: Very High

    This misconfiguration is frequently observed in:
    - *SMB*s with limited IT staff or policy enforcement
    - Environments with legacy scripts or printer driver sharing
    - Ad-hoc developer shares that are never properly decommissioned

Why it exists:

- Convenience during development or troubleshooting
- Misunderstanding of access scope (confusing "Everyone" with "Domain Users")
- Lack of security validation on new shares

- Consequence if exploited
  - Remote Code Execution via dropped malicious executables or scripts
  - Privilege escalation through hijacked login/startup scripts
  - Worm or ransomware propagation
  - Exfiltration of sensitive data or tampering with internal documentation

This setup simulates how a simple share misconfiguration can evolve into a powerful entry point for attackers, often without triggering alerts or being noticed by defenders.

SMB Signing Disabled

To weaken the domain security posture and simulate common enterprise misconfigurations, *SMB signing* was explicitly disabled on both the client and server sides. *SMB* (Server Message Block) is a core protocol used for file sharing, remote management, and *Active Directory* replication.

SMB signing ensures the integrity and authenticity of *SMB* traffic. Disabling it allows attackers to intercept and tamper with communications — enabling techniques such as *NTLM* [71] relay and **man-in-the-middle (MITM)** [71] attacks.

- Configuration and implementation

The following *PowerShell* script was used to disable *SMB* signing:

```
# Disable SMB signing on both the client and server

# SMB Client (workstation)
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters" `
    -Name "EnableSecuritySignature" `
    -Value 0 `
    -Type DWord `
    -Force

# SMB Server (lanmanserver)
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" `
    -Name "EnableSecuritySignature" `
    -Value 0 `
    -Type DWord `
    -Force

Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters" `
    -Name "RequireSecuritySignature" `
    -Value 0 `
    -Type DWord `
    -Force

# Optional: Enable anonymous LDAP access (for wider attack surface)
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\NTDS\Parameters" `
    -Name "Allow Anonymous Access" `
    -Value 1

Write-Host "`n✅ SMB signing disabled (client and server)"
Write-Host "🔁 Restart the system to apply changes"
```

Figure 41 Disable SMB Signing through PowerShell

Disabling *SMB* signing removes integrity verification, allowing adversaries to intercept and manipulate *SMB* traffic.

- Risk and exploitation
  - Severity: Critical
  - Attack Vector: Internal (no special privileges required)
  - Technique: *NTLM* Relay / MITM over *SMB*
  - Affected Component: *SMB* Protocol (LanmanWorkstation[73] & LanmanServer[74] services)

  Exploitation Scenarios:
  - Capture and relay *NTLM*v2 [75] hashes with tools like *NTLM*relayx.py, Responder[76], or Inveigh [77]
  - Trick privileged users (e.g., Domain Admins) into initiating *SMB* connections and gain elevated access
  - Tamper with *SMB* traffic to inject malware or redirect communications
- Mitigation and recommendations
  - Enforce *SMB* Signing via Group Policy:
    - Client Policy:

    - Server Policy:

  - Audit *SMB* Traffic:
    - Monitor for unsigned packets using Wireshark, Event Logs, or SIEMs
    - Look for patterns of *NTLM* relaying or spoofed connections
  - Disable *NTLM* Authentication:
    - In modern environments, enforce Kerberos-only authentication wherever possible
  - Use Defender for Identity / MITM Detection Tools:
    - Detect relay and downgrade attacks across your AD infrastructure
- Realism and justification
  - Realism: Extremely High
    - This misconfiguration is frequently encountered due to:

- Legacy compatibility with older Windows versions or network appliances
- Admins disabling it for performance gains or troubleshooting
- Test or dev environments promoted to production without security review
- Lack of awareness about the security importance of *SMB* signing

Even Microsoft has highlighted this issue in official security baselines and CISA advisories.

- Consequence if exploited
  - Full Domain Compromise: Relay attacks [78] can impersonate any privileged user
  - Credential Theft: Capture of *NTLM*v2 credentials for offline cracking or replay
  - Lateral Movement: Attackers can pivot within the network, executing commands, dropping payloads, and exfiltrating data

  This setup was intentionally deployed to illustrate how a single registry misconfiguration can open the door to devastating attacks, making it a top priority for hardening in production domains.

Table Resume of vulnerabilities

#	Vulnerability	Technique/Attack	Criticality	Notes / Realism
1	*AS-REP Roasting*	*GetNPUsers.py*, *Rubeus*	High	Exploitable without credentials; common in legacy systems.
2	*Kerberoasting*	*SPN* abuse, GetUser*SPN*s.py	Critical	Needs auth; common in weak service accounts.
3	*Dnsadmin*s Privilege Abuse	DLL injection via reg key	Critical	Full SYSTEM access; misused permissions.
4	*DCSync* Rights Assigned to User	***secretsdump.py***, Mimikatz	Critical	Grants domain hashes; common in misconfigured ACLs.
5	*Evil-WinRM* Access with Privileged User	*WinRM* + local admin	Critical	Enables full RCE as SYSTEM.
6	Default Passwords in Descriptions	PowerView enumeration	High	Easy to spot, serious if reused passwords.
7	Anonymous *LDAP* Queries Enabled	*LDAP*search, nmap	High	Leaks directory info; uncommon but dangerous.
8	Public *SMB* Share with Full Access	*SMB* share exploitation	High	Data exfiltration, RCE possible via drop/exec.
9	*SMB* Signing Disabled	*NTLM* relay, Responder	Critical	Enables MITM and full relay attacks.

Enumeration

The enumeration phase focuses on collecting as much information as possible from the target system while avoiding detection. This includes identifying exposed services, open ports, operating system details, authentication mechanisms, and potential usernames within the domain.

- Network and Service Discovery

  Initial connectivity was validated using ICMP echo requests to ensure that the attacking machine (kali) could reach the *Domain Controller* (DC) at 10.80.80.2:

Figure 42 Connectivity test using ping

Once connectivity was confirmed, a port scan was conducted using nmap to identify open TCP ports and associated services:

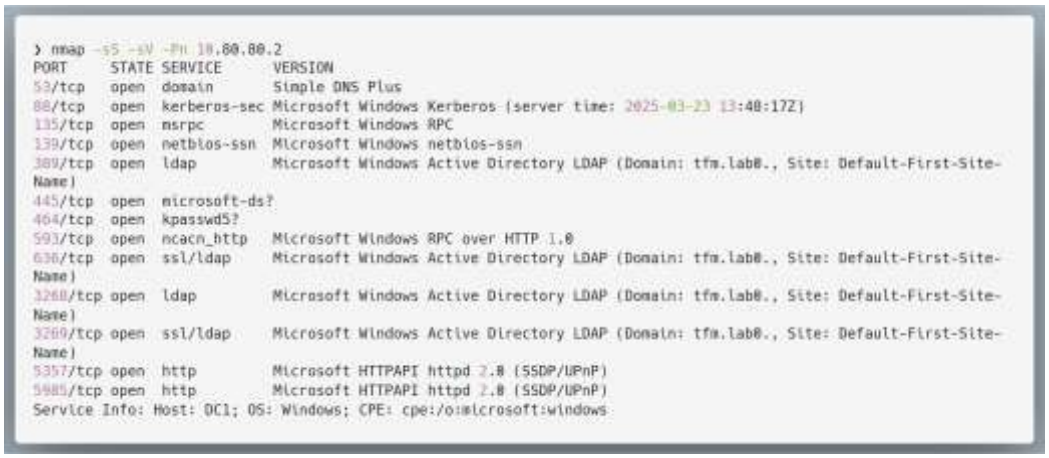


Figure 43 Network scan of DC using nmap

The results revealed several services critical for *Active Directory* environments:

- *DNS* (53/tcp)
- Kerberos (88/tcp)
- MSRPC [79] (135/tcp)
- *SMB* (139/tcp, 445/tcp)
- *LDAP* / *LDAP*S (389/tcp, 636/tcp)
- Global Catalog [80] (3268/tcp, 3269/tcp)

OS fingerprinting results, from RPC [81] and HTTP headers, confirmed that the target host was running Windows Server 2022, validating its role as the primary *Domain Controller*.

- *SMB* Enumeration

  Next, the tool *CrackMapExec* was used to extract additional metadata from the *SMB* service:

Figure 44 SMB metadata enumeration using CrackMapExec

This revealed:

- Hostname: DC1
- Domain: *tfm.lab*
- *SMB* Signing: Enabled
- *SMB*v1: Disabled

While *SMB* Signing was reported as active, further testing would be necessary to determine whether it is strictly enforced. The absence of *SMB*v1 is a positive baseline configuration against known exploits like EternalBlue [82].

To test for accessible shares via anonymous login, the following command was executed:

Figure 45 SMB Enumeration

Although the login succeeded anonymously, no shared folders were listed. This suggests that either:

- No *SMB* shares are accessible anonymously, or
- *SMB* version mismatch prevented enumeration.

- User Enumeration via Kerberos

  The next objective was to identify valid domain users. For this, the tool *Kerbrute* was used in combination with a small custom wordlist derived from SecLists [83] xato-net-10-million-usernames.txt:

Figure 46 Kerbrute

Out of 9 usernames tested, 5 were confirmed to be valid. This technique exploits the discrepancy in Kerberos ***KDC*** **(Key Distribution Centre)** [84] response between valid and invalid usernames, allowing enumeration without triggering authentication logs. The effectiveness of this technique is rated **Low to Medium** in terms of criticality, as it does not grant access, but leaks valuable domain user information.

Such enumeration is common in real-world environments unless mitigated through:

- Pre-authentication requirements for all users
- Account lockout policies
- Log monitoring (e.g., Kerberos Event ID 4771)

Exploitation

Once valid domain users have been discovered through enumeration techniques, the next step is to explore potential paths for gaining unauthorized access. This phase includes leveraging exposed credentials, misconfigurations, and weak access controls to obtain a foothold inside the *Active Directory* environment.

- *AS-REP Roasting* Candidate Discovery

  After discovering *charlie.george* as an *AS-REP Roasting* candidate, a hash was extracted using *GetNPUsers.py* and subjected to offline cracking with *Hashcat* (mode 18200) and leveraging the well-known **rockyou.txt** wordlist.

Figure 47 GetNPUsers.py

- Offline Hash Extraction

  The retrieved hash was then subjected to a brute-force attack using *Hashcat* with the rockyou.txt wordlist:

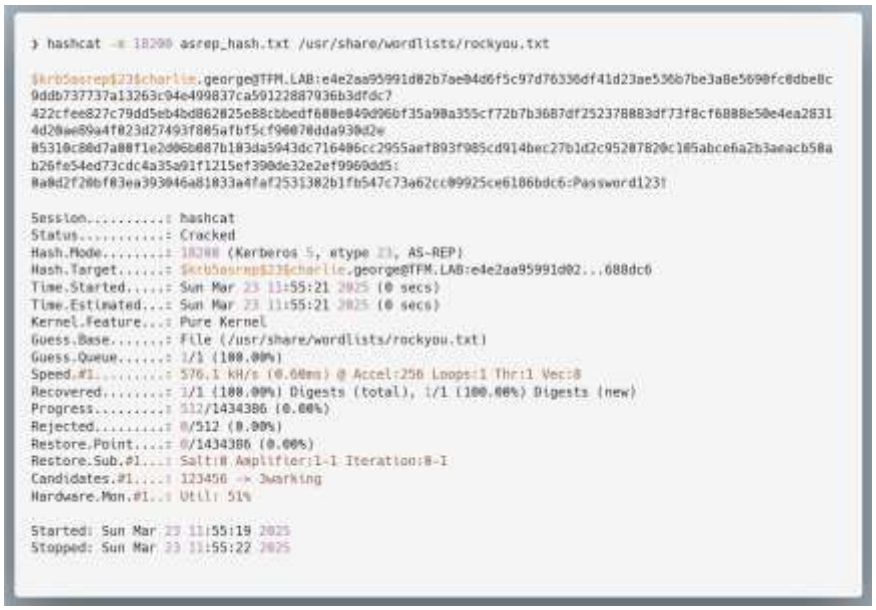


Figure 48 Hashcat

The password was successfully cracked, revealing **Password123!** as the password for the user *charlie.george*. This confirms that the account is both vulnerable and actively exploitable.

- Accessing *SMB* Resources with Recovered Credentials

  After retrieving the password for *charlie.george* was obtained, authentication against the *SMB* service of the *Domain Controller* (10.80.80.2) was tested using ***smbclient***:

Figure 49 Login SMB with charlie.george credentials

The result revealed access to several standard and custom shares, including:

- ADMIN$ – Remote administration share.
- C$ – Default administrative share.
- IPC$ – Inter-process communication.
- ***NETLOGON*** – Logon scripts.
- **SYSVOL** [24] – Group policy data.
- *PublicShare* – Custom public folder exposed during lab setup.

This validation confirmed access to several standard shares such as *NETLOGON*, SYSVOL, and a custom share named *PublicShare*. The presence of these resources is common in *Active Directory* environments and often reveals sensitive configurations, scripts, or backup files.

On the other hand, as expected, the connection attempt using *SMB*1 for workgroup enumeration failed. This is a normal behaviour, as modern environments typically use *SMB*2 or *SMB*3, and *SMB*1 is deprecated.

- Discovery of Password Reuse in Public Share

  Leveraging the same set of credentials, the *PublicShare* directory was accessed interactively:

Figure 50 PublicShare directory

The file contained a note explicitly stating: "*I use the same password as* svc-sync", revealing critical password reuse.

This revealed a case of **password reuse**, a common security weakness in enterprise environments where service or privileged accounts share credentials with standard users. This disclosure pointed directly to the svc-sync account, previously enumerated in the domain.

- Reuse of Credentials for Privileged Service Account

  An authentication attempt was made against the *Domain Controller* using the svc-sync username and the previously discovered password:

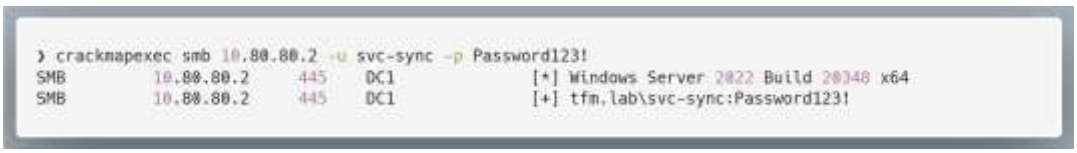
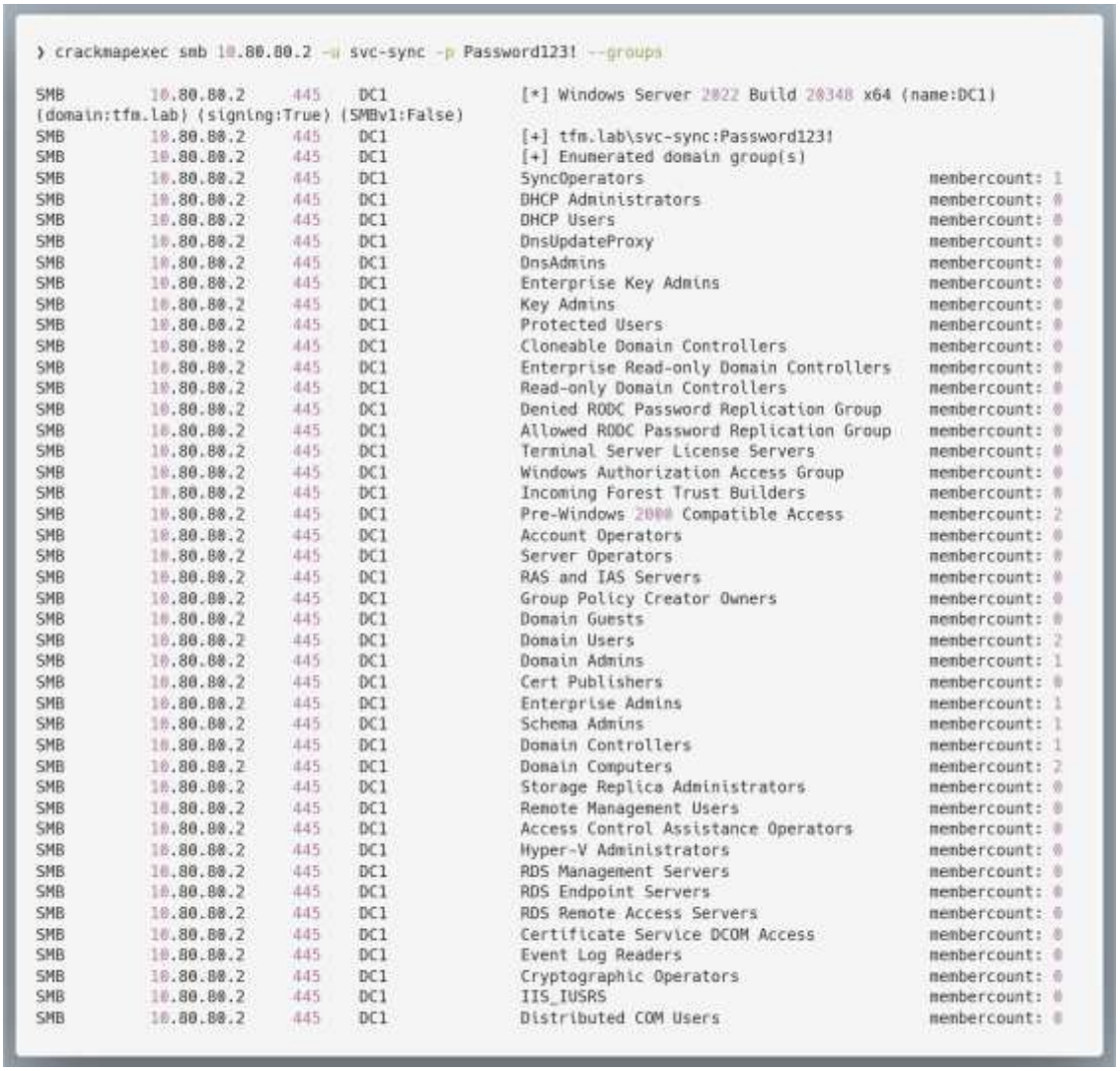


Figure 51 Authenticating with CrackMapExec

Authentication was completed successfully, confirming that the service account svc-sync shared the same credentials as *charlie.george*. Although no elevated privileges were yet confirmed, this established a second authenticated entry point into the domain.

- Group Membership and Privilege Discovery (svc-sync)

  Once the svc-sync credentials were confirmed as valid, the next logical step was to identify the privileges assigned to this account. Using *CrackMapExec* with the --groups flag, we enumerated all domain groups and confirmed that svc-sync is a member of the SyncOperators group — a custom security group configured earlier in the environment.



```
❯ crackmapexec smb 10.80.80.2 -u svc-sync -p Password123! --groups

SMB         10.80.80.2     445    DC1              [*] Windows Server 2022 Build 20348 x64 (name:DC1)
(domain:tfm.lab) (signing:True) (SMBv1:False)
SMB         10.80.80.2     445    DC1              [+] tfm.lab\svc-sync:Password123!
SMB         10.80.80.2     445    DC1              [+] Enumerated domain group(s)
SMB         10.80.80.2     445    DC1              SyncOperators                                membercount: 1
SMB         10.80.80.2     445    DC1              DHCP Administrators                          membercount: 0
SMB         10.80.80.2     445    DC1              DHCP Users                                   membercount: 0
SMB         10.80.80.2     445    DC1              DnsUpdateProxy                               membercount: 0
SMB         10.80.80.2     445    DC1              DnsAdmins                                    membercount: 0
SMB         10.80.80.2     445    DC1              Enterprise Key Admins                        membercount: 0
SMB         10.80.80.2     445    DC1              Key Admins                                   membercount: 0
SMB         10.80.80.2     445    DC1              Protected Users                              membercount: 0
SMB         10.80.80.2     445    DC1              Cloneable Domain Controllers                 membercount: 0
SMB         10.80.80.2     445    DC1              Enterprise Read-only Domain Controllers      membercount: 0
SMB         10.80.80.2     445    DC1              Read-only Domain Controllers                 membercount: 0
SMB         10.80.80.2     445    DC1              Denied RODC Password Replication Group       membercount: 0
SMB         10.80.80.2     445    DC1              Allowed RODC Password Replication Group      membercount: 0
SMB         10.80.80.2     445    DC1              Terminal Server License Servers              membercount: 0
SMB         10.80.80.2     445    DC1              Windows Authorization Access Group           membercount: 0
SMB         10.80.80.2     445    DC1              Incoming Forest Trust Builders               membercount: 0
SMB         10.80.80.2     445    DC1              Pre-Windows 2000 Compatible Access           membercount: 2
SMB         10.80.80.2     445    DC1              Account Operators                            membercount: 0
SMB         10.80.80.2     445    DC1              Server Operators                             membercount: 0
SMB         10.80.80.2     445    DC1              RAS and IAS Servers                          membercount: 0
SMB         10.80.80.2     445    DC1              Group Policy Creator Owners                  membercount: 0
SMB         10.80.80.2     445    DC1              Domain Guests                                membercount: 0
SMB         10.80.80.2     445    DC1              Domain Users                                 membercount: 2
SMB         10.80.80.2     445    DC1              Domain Admins                                membercount: 1
SMB         10.80.80.2     445    DC1              Cert Publishers                              membercount: 0
SMB         10.80.80.2     445    DC1              Enterprise Admins                            membercount: 1
SMB         10.80.80.2     445    DC1              Schema Admins                                membercount: 1
SMB         10.80.80.2     445    DC1              Domain Controllers                           membercount: 1
SMB         10.80.80.2     445    DC1              Domain Computers                             membercount: 2
SMB         10.80.80.2     445    DC1              Storage Replica Administrators               membercount: 0
SMB         10.80.80.2     445    DC1              Remote Management Users                      membercount: 0
SMB         10.80.80.2     445    DC1              Access Control Assistance Operators          membercount: 0
SMB         10.80.80.2     445    DC1              Hyper-V Administrators                       membercount: 0
SMB         10.80.80.2     445    DC1              RDS Management Servers                       membercount: 0
SMB         10.80.80.2     445    DC1              RDS Endpoint Servers                         membercount: 0
SMB         10.80.80.2     445    DC1              RDS Remote Access Servers                    membercount: 0
SMB         10.80.80.2     445    DC1              Certificate Service DCOM Access              membercount: 0
SMB         10.80.80.2     445    DC1              Event Log Readers                            membercount: 0
SMB         10.80.80.2     445    DC1              Cryptographic Operators                      membercount: 0
SMB         10.80.80.2     445    DC1              IIS_IUSRS                                    membercount: 0
SMB         10.80.80.2     445    DC1              Distributed COM Users                        membercount: 0
```

Figure 52 Group enumeration

The command confirmed that svc-sync was a member of a custom security group named SyncOperators. This group had previously been granted replication privileges over the domain using custom Access Control Entries (ACEs) [85], specifically:

- DS-Replication-Get-Changes
- DS-Replication-Get-Changes-All
- DS-Replication-Get-Changes-In-Filtered-Set

These permissions allow the user to replicate sensitive directory data, including password hashes from any domain account. This effectively enables the svc-sync account to perform a *DCSync* attack.

- *DCSync* Constraints and Interactive Access Limitations

  Despite possessing replication rights, the svc-sync account was not part of the *Administrator*s or Remote Management Users groups on the *Domain Controller*. Therefore, when attempting remote code execution using *Evil-WinRM*, the connection failed with a *WinRMAuthorizationError*.

Figure 53 WinRM connection failed

This limitation illustrates a key concept in *Active Directory*: domain-level privileges do not automatically translate to local administrative access, and the ability to interact with systems remotely still depends on local group memberships.

This behaviour, although restrictive in one aspect, still leaves the domain at critical risk due to the replication capabilities granted via ACL misconfiguration.

- Identification of Additional *AS-REP Roasting* Candidates

  With continued access via the svc-sync account, the script *GetNPUsers.py* was executed again, this time revealing two accounts configured without Kerberos preauthentication:

  - *charlie.george*
  - svc-backup

Figure 54 AS-REP Roasting Candidates

Analysis results indicated that both accounts responded with encrypted AS-REP messages, confirming that the *UF_DONT_REQUIRE_PREAUTH* flag was enabled. These messages could be cracked offline to recover additional passwords without generating alerts.

Both accounts showed recent login timestamps and a UserAccountControl value of 0x400200, confirming that they were active and not disabled.

- Remote Enumeration via RPCClient

  To expand user and group enumeration beyond previous Kerberos-based techniques, the rpcclient tool was utilized with the credentials of the compromised user *charlie.george*. This tool interacts with the Remote Procedure Call (RPC) services over *SMB*, allowing deeper inspection of *Active Directory* data exposed by the DC.

Figure 55 RPCClient analysis

Once authenticated, several commands were executed to enumerate domain users and groups.

Users Enumeration

The enumdomusers command revealed a comprehensive list of users within the domain, including:

- Built-in accounts: *Administrator*, Guest, *KRBTGT*
- Service accounts: svc-backup, svc-sync, *Dnsadmin*
- Domain admins and regular users: *domain.admin*, jparker, esparrow, *charlie.george*, etc.

This technique exposes a broader view of the user landscape, including accounts not previously identified through *Kerbrute* or *AS-REP Roasting*, such as:

- eva.steve, eva.frank, steve.steve, carol, and others.

Groups Enumeration

The enumdomgroups command listed all domain groups, including high-privilege groups such as:

- Domain Admins (RID: 0x200)
- Enterprise Admins (RID: 0x207)
- Schema Admins, *DNS*UpdateProxy, and SyncOperators

Group Membership Queries

This revealed that:

- *Administrator* and *domain.admin* accounts are members of the Domain Admins group.
- The svc-sync user was confirmed as a member of the SyncOperators group.
- The svc-backup user appeared under *DNS*UpdateProxy, likely due to its role in managing *DNS*-related services.

This technique reinforces the privilege mapping of compromised accounts and helps identify potential lateral movement paths based on group memberships

- Share Enumeration and SYSVOL Inspection

  Using the credentials of the svc-sync user, share enumeration was performed to identify accessible resources on the *Domain Controller*.

```
❯ crackmapexec smb 10.80.80.2 -u svc-sync -p Password123! --shares

SMB         10.80.80.2     445    DC1              [*] Windows Server 2022 Build 20348 x64
(name:DC1) (domain:tfm.lab) (signing:True) (SMBv1:False)
SMB         10.80.80.2     445    DC1              [+] tfm.lab\svc-sync:Password123!
SMB         10.80.80.2     445    DC1              [+] Enumerated shares
SMB         10.80.80.2     445    DC1              Share           Permissions     Remark
SMB         10.80.80.2     445    DC1              ADMIN$          -               Remote Admin
SMB         10.80.80.2     445    DC1              C$              -               Default share
SMB         10.80.80.2     445    DC1              IPC$            READ            Remote IPC
SMB         10.80.80.2     445    DC1              NETLOGON        READ            Logon server share
SMB         10.80.80.2     445    DC1              PublicShare     READ,WRITE      -
SMB         10.80.80.2     445    DC1              SYSVOL          READ            Logon server share
```

Figure 56 Authenticated share enumeration with svc-sync

The following shares were discovered with varying access permissions:

- *NETLOGON* and SYSVOL: READ
- *PublicShare*: READ, WRITE
- ADMIN$, C$, IPC$: default administrative shares

A deeper inspection was performed against SYSVOL, which commonly stores login scripts and Group Policy files.

```
❯ smbclient //10.80.80.2/SYSVOL -U svc-sync
Password for [WORKGROUP\svc-sync]:
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Sun Mar 16 16:28:42 2025
  ..                                  D        0  Sun Mar 16 16:28:42 2025
  tfm.lab                             Dr       0  Sun Mar 16 16:28:42 2025

    12946687 blocks of size 4096. 10839634 blocks available

smb: \> cd tfm.lab\
smb: \tfm.lab\> ls
  .                                   D        0  Sun Mar 16 16:38:21 2025
  ..                                  D        0  Sun Mar 16 16:28:51 2025
  DfsrPrivate                         DHSr     0  Sun Mar 16 16:28:51 2025
  Policies                            D        0  Sun Mar 16 16:28:51 2025
  scripts                             D        0  Sun Mar 16 16:28:42 2025

    12946687 blocks of size 4096. 10839634 blocks available

smb: \tfm.lab\> cd Policies
smb: \tfm.lab\Policies\> ls
  .                                   D        0  Sun Mar 16 16:28:51 2025
  ..                                  D        0  Sun Mar 16 16:28:51 2025
  {31B2F348-016D-11D2-945F-00C04FB984F9}  D     0  Sun Mar 16 16:28:51 2025
  {6AC1786C-016F-11D2-945F-00C04FB984F9}  D     0  Sun Mar 16 16:28:51 2025

    12946687 blocks of size 4096. 10839634 blocks available
```

Figure 57 SYSVOL analysis

Within the share, the following paths were identified:

- *tfm.lab*\scripts: May be used for user logon tasks
- *tfm.lab*\Policies: Contains GPO templates, with GUID-named directories such as:
  - {31B2F340-016D-11D2-945F-00C04FB984F9}
  - {6AC1786C-016F-11D2-945F-00C04FB984F9}

These structures are standard in *Active Directory* environments, and further GPP password harvesting or GPO abuse may be possible depending on permissions.

- *Kerberoasting* Enumeration

  Continuing with the svc-sync credentials, a *Kerberoasting* enumeration was performed to identify service accounts with Service Principal Names (*SPN*s) that could be exploited.

Figure 58 SPNs exposed for svc-backup *account (Kerberoasting)*

This revealed the following *SPN*s associated with the svc-backup user:

- HTTP/backup.*tfm.lab*
- svc-backup/backup.*tfm.lab*

The presence of these *SPN*s indicated that the account was eligible for a *Kerberoasting* attack.

- *Kerberoasting* Hash Extraction

  A Ticket Granting Service (*TGS*) hash for the svc-backup account was requested using the -request flag:

Figure 59 TGS hash extraction using GetUserSPNs.py

This command successfully returned an encrypted *TGS* hash, which could be brute-forced offline to retrieve the plaintext password.

- Offline Hash Cracking

  The extracted *Kerberoasting* hash was cracked using *Hashcat* and the rockyou.txt wordlist with mode 13100 (*TGS*-REP):

Figure 60 Offline TGS hash cracking using Hashcat

The password Password123! was successfully recovered, confirming weak password reuse. This granted direct access to the svc-backup account, marking it as another point of compromise within the domain.

The next section explores how this account is leveraged for privilege escalation via *DCSync* attack.

- Share Enumeration with svc-backup

  Using the newly recovered credentials (svc-backup: Password123!), a share enumeration was conducted again with *CrackMapExec*:

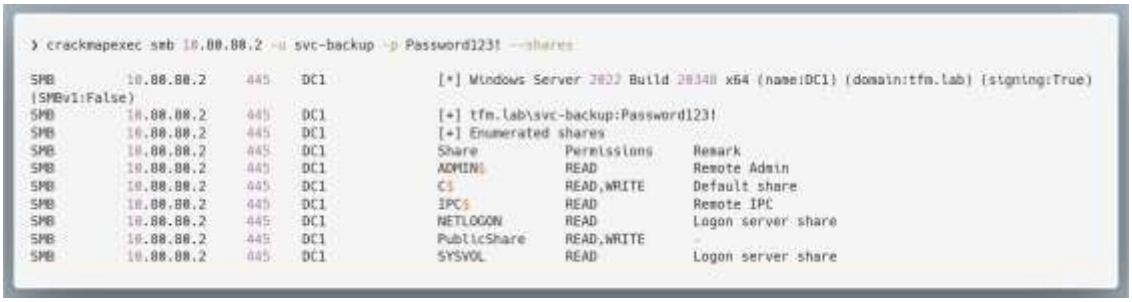


Figure 61 SMB share enumeration using svc-backup credentials

The following shares were accessible:

- ADMIN$: READ
- C$: READ, WRITE
- IPC$: READ
- *NETLOGON*: READ
- *PublicShare*: READ, WRITE
- SYSVOL: READ

The C$ share, a default administrative share, granted read and write permissions.

- NTDS Directory Access Attempt

  An attempt was made to access the \Windows\NTDS directory to locate the *ntds.dit* [86] file — a critical file that stores *Active Directory* data including password hashes.

Figure 62 NTDS directory access attempt from C$ share

This indicates that although the share was accessible, directory-level permissions restricted access to sensitive contents like ntds.dit. The privilege level of svc-backup was insufficient to continue with this method of credential extraction.

Post-Exploitation

- *DCSync* Hash Extraction

  Having identified earlier that the svc-sync account is a member of the SyncOperators group with replication privileges, an attempt was made to execute a *DCSync* attack using *secretsdump.py* from the *Impacket* toolkit:

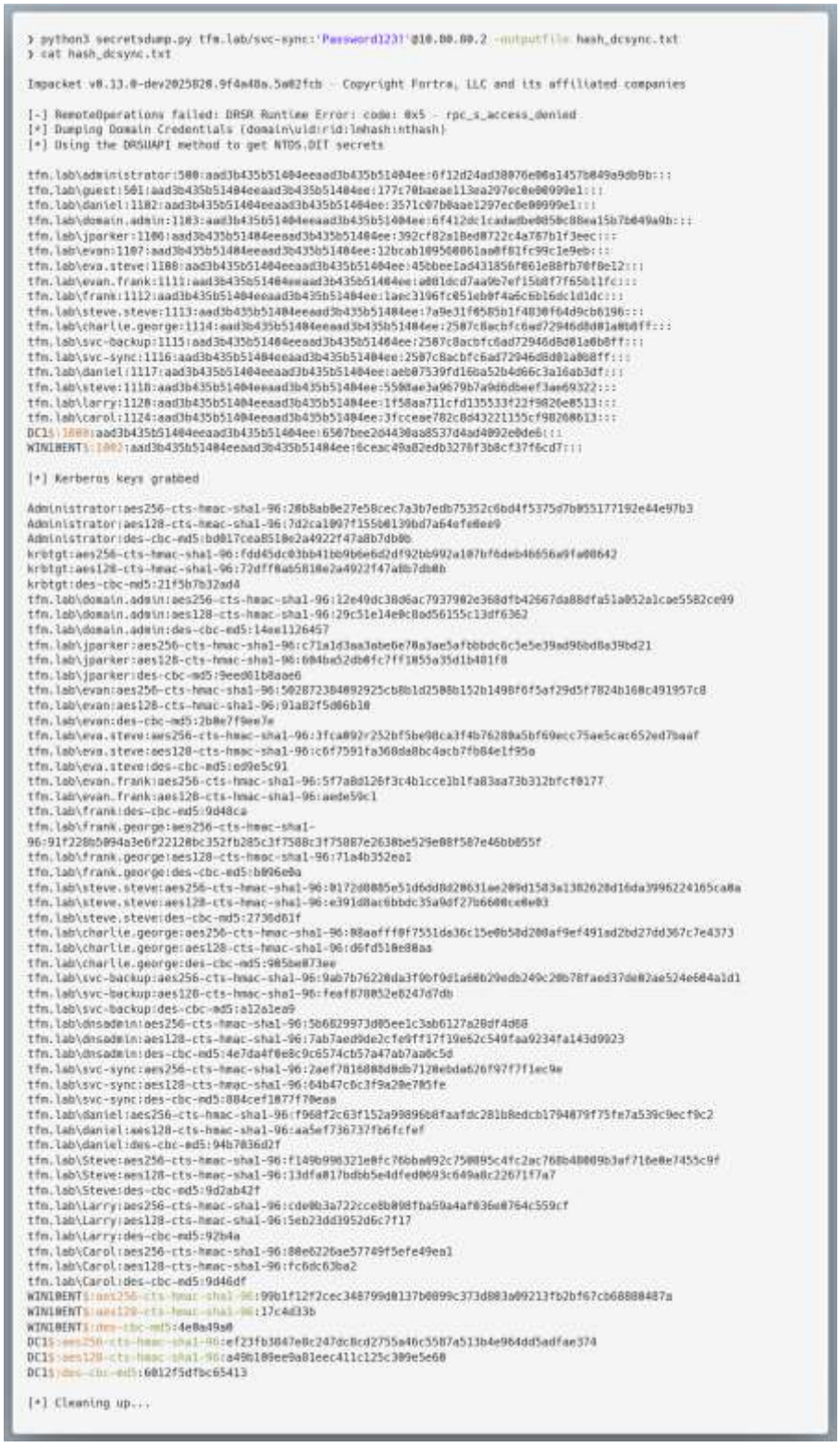


Figure 63 secretsdump.py output showing NTLM hashes

The operation completed successfully, and all *NTLM hashes* from the *Domain Controller* were dumped. This includes hashes for:

- Default accounts: *Administrator*, *KRBTGT*
- Domain users: *domain.admin*, jparker, *charlie.george*, etc.
- Service accounts: svc-sync, svc-backup, *Dnsadmin*

The hashes extracted are stored in the output file hash_*DCSync*.txt.

- Filtering Domain Admin Hash

  To isolate high-value targets, the hash for *domain.admin* was extracted using a simple grep command:

Figure 64 NTLM hash filtering for domain.admin

This revealed several encryption types for the user *domain.admin*, including:

- *NTLM* hash
- AES256-CTS-HMAC-SHA1-96
- AES128-CTS-HMAC-SHA1-96
- DES-CBC-MD5

The extracted *NTLM* hash was selected as the credential of choice for subsequent pass-the-hash authentication.

- Remote Code Execution as Domain Admin

  With the *NTLM* hash for *domain.admin* in hand, an *Evil-WinRM* session was initiated using *pass-the-hash* [87] technique:

Figure 65 Remote access as domain.admin using Evil-WinRM

Access was successfully granted to the *Domain Controller* with full privileges. Navigating to the Desktop directory confirmed access to sensitive post-exploitation scripts such as:

- *DCSync*.ps1
- domain.ps1
- asrep_charlie.ps1
- *SMB*_signature.ps1
- test.ps1

This final step confirms full domain compromise. The attacker now has unrestricted access and can perform any administrative action within the environment, including dumping additional credentials, creating new domain users, modifying group policies, and establishing persistence mechanisms.

Conclusions

The audit conducted throughout this laboratory environment demonstrates how misconfigurations, weak password policies, and poorly managed privileges can lead to complete domain compromise. From basic user enumeration and *SPN* exposure to offline hash cracking and abuse of *DCSync* replication rights, the assessment highlights a realistic attack path that adversaries could exploit in production environments.

Throughout this audit, we demonstrated how a relatively low-privileged user (*charlie.george*) could pivot through multiple misconfigurations—gaining access to public shares, discovering reused passwords, and eventually escalating to full administrative control via a *DCSync* attack and pass-the-hash technique against the *domain.admin* account.

These findings reaffirm the importance of hardening *Active Directory* environments and performing regular internal audits, even in seemingly isolated or test domains. Simple lapses such as password reuse or disabled pre-authentication can offer attackers a foothold that enables further lateral movement and *privilege escalation*.

Fulfilment of Objectives

The objectives outlined at the beginning of this project have been fully accomplished. A realistic *Active Directory* environment was successfully designed and deployed, deliberately populated with misconfigurations that reflect real-world enterprise mistakes. Vulnerabilities were not only identified but also exploited through a structured methodology, documenting the full path from initial access to domain compromise. Finally, the project provides a detailed set of recommendations to mitigate each weakness, fulfilling the educational and technical goals set for this thesis.

Additional Techniques Not Used

While the audit successfully achieved full domain compromise through manual and scripted enumeration, there are several additional techniques and tools that could further enrich the scope of this assessment:

- ***BloodHound* with SharpHound Collector**: *BloodHound* is a powerful graph-based tool for mapping relationships and trust paths within *Active Directory*. SharpHound can collect data on sessions, group memberships, ACLs, and more to identify hidden privilege escalation paths such as "kerberoastable" users with admin sessions, unconstrained delegation, or write access to sensitive objects.

- **LSASS Dumping**: If local admin privileges had been obtained earlier, tools such as Mimikatz could be used to dump LSASS [88] memory and extract cleartext credentials or additional *NTLM* hashes.
- **GPP Password Retrieval**: Group Policy Preferences stored in SYSVOL often contain XML [89] files with credentials encrypted using a known static key. Although no such files were found in this audit, they are still a common source of escalation in many domains.
- *NTLM* **Relay Attacks**: In environments with *SMB* Signing disabled, *NTLM* relay attacks can be used to impersonate users by relaying authentication requests to services like *LDAP*, *SMB*, or HTTP.
- **Token Impersonation and Service Abuse**: Leveraging tools like *Rubeus* or Rotten Potato [90], attackers can impersonate privileged tokens or abuse scheduled tasks and services for escalation.
- **Password Spraying**: This technique involves attempting a single password across multiple user accounts instead of brute-forcing one account. In this lab environment, it would have been particularly effective due to the reuse of the password Password123! across several users such as *charlie.george*, svc-sync, and svc-backup. This method is commonly used in real-world attacks to bypass account lockout policies and avoid detection, especially when user enumeration has already been successful.

Importance of Periodic Auditing

Security in enterprise environments is not a one-time effort but a continuous process. Each change, update, or onboarding of a new system introduces potential weaknesses that may not be immediately obvious.

For this reason, periodic internal audits are essential:

- After system updates or group policy changes
- Following onboarding of new third-party applications or service accounts
- Upon structural changes in teams or user access

Routine assessments allow organizations to detect drift from baseline configurations, identify exposure of critical services, and uncover hidden privilege relationships that evolve over time.

The Role of Cybersecurity in Modern Enterprises

Cybersecurity today is not just a support function; it is a strategic business enabler. Modern threats have evolved to target not just perimeter defences but the internal structure of organizations, making *Active Directory* a primary focus due to its central role in authentication and resource management.

Key priorities for enterprise cybersecurity teams should include:

- Principle of Least Privilege (PoLP)
- Segmentation of administrative roles (Tiered Access Models)
- Continuous logging and alerting
- Threat modelling and adversary simulation
- Employee training and awareness on phishing and credential hygiene

Failing to address these areas can lead to severe consequences including data breaches, ransomware propagation, or full domain takeovers as illustrated in this audit.

Ultimately, proactive defence, regular auditing, and strong access control are vital to reducing attack surfaces and ensuring resilience in enterprise environments.

Future Work

While this thesis successfully demonstrates full domain compromise through internal threat simulation, several opportunities remain to expand the depth and realism of such assessments.

Future work could explore hybrid environments by integrating Azure Active Directory and cloud identities to reflect modern enterprise infrastructure. Incorporating Endpoint Detection and Response (EDR) tools, such as Microsoft Defender for Endpoint or Velociraptor, would enable the evaluation of detection and response capabilities during live attacks.

Additionally, implementing real-time event monitoring using Sysmon with centralized log collection (e.g., ELK Stack or Splunk) could provide deeper visibility into attacker behavior and support correlation-based alerting.

Finally, automating portions of the attack chain using adversary emulation frameworks like CALDERA or Empire could simulate more advanced threat scenarios aligned with the MITRE ATT&CK framework, elevating the realism and challenge of internal security testing.

Bibliographic references

Information:

¿Qué es Active Directory? ¿Cómo funciona? | Quest. (n.d.). Quest Software. https://www.quest.com/mx-es/solutions/active-directory/what-is-active-directory.aspx

Microsoft. (n.d.). Active Directory Domain Services Overview. Microsoft Learn. https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview

MITRE. (n.d.). MITRE ATT&CK® Framework. https://attack.mitre.org/

SecureAuth. (n.d.). Impacket. GitHub. https://github.com/fortra/Impacket

Hack The Box. (n.d.). Cybersecurity Training Platform. https://www.hackthebox.com/

TryHackMe. (n.d.). Hands-on Cybersecurity Training. https://tryhackme.com/

xBEN. (2025, January 4). Building a PfSense VM for our VirtualBox Cyber range. 0xBEN. https://benheater.com/VirtualBox-lab-pfSense-firewall/#pfSense-ce-offline-installer

xBEN. (2024, September 17). Importing Kali using the official VirtualBox image. 0xBEN. https://benheater.com/VirtualBox-lab-import-kali-linux/

xBEN. (2025c, April 3). Configuring the PfSense firewall for our VirtualBox lab. 0xBEN. https://benheater.com/VirtualBox-lab-configure-pfSense/

xBEN. (2024a, September 17). Adding an Active Directory forest to our VirtualBox lab. 0xBEN. https://benheater.com/VirtualBox-lab-active-directory-lab/

xBEN. (2025c, January 27). Hack your VirtualBox AD Lab. 0xBEN. https://benheater.com/hack-your-VirtualBox-ad-lab/

CVE. (n.d.). Common Vulnerabilities and Exposures (CVE). CVE Program. https://www.cve.org/

Proofpoint. (n.d.). Active Directory. Threat Reference. https://www.proofpoint.com/es/threat-reference/active-directory

IMF Smart Education. (2025). Master's Degree in Cybersecurity. Internal academic training material (unpublished).

Mermaid. (n.d.). Mermaid Live Editor. https://mermaid.live/

Tools:

Carbon. (n.d.). Carbon. https://carbon.now.sh/

Grammarly: free AI writing assistance. (n.d.). https://www.grammarly.com/

OpenAI. (2024). ChatGPT (versión GPT-4) [Modelo de lenguaje grande]. https://chat.openai.com/

Anthropic. (n.d.). Claude [Large language model]. https://claude.ai/

Scribbr. (n.d.). Scribbr: Free citation generator. https://www.scribbr.com/

Glossary of Technical Terms

[1] ***Active Directory*** **(AD):** A Microsoft directory service used for managing users, computers, and other resources in a Windows domain network.

[2] ***Kerberoasting*****:** Post-authentication attack that retrieves service tickets to crack service account passwords offline.

[3] ***AS-REP Roasting*****:** Attack targeting accounts without Kerberos pre-authentication to obtain hashes for offline cracking.

[4] ***DCSync*****:** A technique used to simulate a *Domain Controller* and request credential replication.

[5] **Privilege Escalation:** A process of gaining higher-level permissions after initial access

[6] **Lateral Movement:** The technique used by attackers to move across system within a compromised network

[7] **Credential dumping:** Technique used to extract passwords, hashes, or other credentials from memory or system files.

[8] **Domain escalation:** Technique used to extract passwords, hashes, or other credentials from memory or system files.

[9] ***Domain Controller*** **(DC):** A server that manages security authentication requests in an *Active Directory* environment.

[10] ***Active Directory*** **Domain Services (AD DS):** Core component of *Active Directory* responsible for storing and managing domain information.

[11] ***DNS*****:** Service that translates domain names into IP addresses.

[12] **DHCP:** Protocol that automatically assigns IP addresses to devices on a network.

[13] **Group Policy:** A feature used to centrally manage settings and permissions for users and computers in a domain.

[14] ***BloodHound*****:** Tool that uses graph theory to reveal hidden privilege escalation paths in AD.

[15] ***Kerbrute*****:** A collection of Python tools and scripts for interacting with network protocols

[16] ***Impacket*****:** A collection of Python tools and scripts for interacting with network protocols

[17] ***CrackMapExec*****:** Swiss-army knife for pentesters, useful for assessing *SMB*, RDP, *WinRM*, and *LDAP*.

[18] ***Hashcat*:** Swiss-army knife for pentesters, useful for assessing *SMB*, RDP, *WinRM*, and *LDAP*.

[19] **LAN:** Local Area Network; a network that connects computers within a limited area such as an office.

[20] **DMZ:** Demilitarized Zone; a part of the network that adds a layer of security between internal and external networks.

[21] **IP:** Internet Protocol address that identifies a device on a network.

[22] **Directory Services Restore Mode (DSRM):** A special boot mode for repairing or restoring *Active Directory*.

[23] ***Active Directory* Certificate Services (AD CS):** Role that allows issuing and managing digital certificates in a domain.

[24] ***LDAP*:** Lightweight Directory Access Protocol, used to read and modify directory services

[25] **HTTPS:** Secure version of HTTP using encryption (TLS) for communication over the web.

[26] **Subnet Mask:** A number that divides IP addresses into network and host parts.

[27] **Default Gateway:** The IP address used to access other networks outside the local one.

[28] ***Active Directory* Users and Computers (AD UC):** Administrative tool for managing users, groups, and computers in AD.

[29] **Security Identifier (SID):** A unique ID used to identify user, group or computer accounts in Windows.

[30] ***PowerShell*:** A command-line shell and scripting language for task automation in Windows.

[31] **OOBE (Out-of-box Experience):** Initial setup process shown when Windows is launched for the first time.

[32] **Enumeration:** The process of gathering information about a system or network.

[33] **Password Spraying:** Technique where the same password is tried across many usernames to avoid lockout policies.

[34] **Pass-The-Ticket:** A technique where attackers use stolen Kerberos tickets to access network resources without needing user credentials**.**

[35] **Kerberos:** A network authentication protocol that uses tickets to allow secure communication between systems.

[36] **Ticket Granting Ticket (*TGT*):** A special ticket in Kerberos that allows users to request access to other services without re-entering credentials.

[37] **CVE (Common Vulnerabilities and Exposures):** A system that provides identifiers for publicly known cybersecurity vulnerabilities.

[38] ***GetNPUsers.py*****:** A script from the *Impacket* toolkit used to extract AS-REP hashes for offline cracking.

[39] ***Rubeus*****:** Post-exploitation tool used to interact with Kerberos in *Active Directory*

[40] **AS-REQ:** Authentication Service Request; the first step in the Kerberos authentication process.

[41] **Service Principal Name (*SPN*s):** A unique identifier for a service instance used in Kerberos authentication.

[42] **Ticket Granting Service (*TGS*):** Part of the Kerberos protocol that issues service tickets after verifying a *TGT*.

[43] **Group Managed Service Accounts (gMSA):** Service accounts managed automatically by AD, often used for applications and services.

[44] **DLLs:** Dynamic Link Libraries; files containing code that can be loaded and executed by programs.

[45] **Sysmon:** A Windows system monitoring tool that logs detailed information about system activity for security analysis.

[46] **SIEM (Security Information and Event Management):** software that collects and analyses log data for security monitoring.

[47] **LAPS:** Local *Administrator* Password Solution; a Microsoft tool that manages and rotates local admin passwords automatically.

[48] **Just Enough Administration (JEA):** A *PowerShell*-based security model that allows delegated administration with minimal privileges.

[49] ***KRBTGT*****:** The Kerberos service account in *Active Directory* responsible for encrypting and signing all *TGT*s.

[50] **Access Control List (ACL):** A list of permissions attached to an object defining who can access it and how.

[51] **Mimikatz:** A powerful tool for extracting credentials from memory and other Windows attacks

[52] **Secretsdump:** A script in *Impacket* used to extract credentials and hashes from remote Windows machines.

[53] **SharpHound:** Data collection component of *BloodHound* for Windows environments.

[54] **Golden Ticket:** A forged Kerberos ticket generated using the *KRBTGT* hash to impersonate any user

[55] **PingCastle:** A tool for assessing the security level of an *Active Directory* environment and detecting vulnerabilities.

[56] **PowerView:** A *PowerShell* tool used for enumerating *Active Directory* components and permissions.

[57] **Advanced Persistent Threat (APT):** A stealthy and prolonged cyberattack where attackers gain and maintain unauthorized access to a network.

[58] **GUID (Globally Unique Identifier):** A unique reference number used to identify objects in systems like *Active Directory*.

[59] ***Evil-WinRM*:** A post-exploitation tool for executing commands remotely over the Windows Remote Management protocol.

[60] **Remote Code Execution (RCE):** A vulnerability that allows attackers to execute arbitrary code on a remote system.

[61] **Role-Based Access Control (RBAC):** A system that restricts access based on a user's role within an organization.

[62] **GPO (Group Policy Object):** A set of rules that control the working environment of user accounts and computers.

[63] ***LDAP*search:** A command-line tool for querying *LDAP* directory servers.

[64] **nmap:** A network scanning tool used to discover hosts and services on a network**.**

[65] **NTDS:** *Active Directory* database service responsible for storing domain information.

[66] **Get-ItemProperty:** A *PowerShell* command used to read registry entries and system properties.

[67] **NTFS:** New Technology File System; the standard file system used by Windows operating systems.

[68] ***SMB* (Server Block Message):** a protocol used for sharing files, printers, and ports on a network.

[69] **SharpShares:** A tool for enumerating shared folders in *Active Directory* environments.

[70] **ShareEnum:** A graphical tool for viewing shared resources across a Windows network.

[71] **NTFS:** An attack where the attacker intercepts and possibly alters communication between two parties.

[72] **Man-In-The-Middle (MITM):** An attack where the attacker intercepts and possibly alters communication between two parties.

[73] **LanmanWorkstation:** A Windows service that allows the client to connect to remote *SMB* resources.

[74] **LanmanServer:** A Windows service that enables sharing files and printers over a network.

[75] ***NTLM*v2:** An updated version of the *NTLM* authentication protocol, providing stronger security.

[76] **Responder:** A tool used to poison name resolution requests on a local network, capturing credentials and hashes.

[77] **Inveigh:** A *PowerShell*-based tool for performing network poisoning and credential harvesting attacks like Responder.

[78] **Relay attack:** A type of MITM attack where intercepted authentication is forwarded to another service to gain access.

[79] **MSRPC (Microsoft's implementation of the Remote Procedure Call (RPC):** Protocol for network service interaction.

[80] **Global Catalog:** A distributed data repository used by *Active Directory* to quickly locate objects in the forest.

[81] **Remote Procedure Call (RPC):** A protocol that allows programs to execute code on a remote system.

[82] **EternalBlue:** A Windows *SMB* exploit used in major ransomware attacks like WannaCry.

[83] **SecLists:** A collection of multiple types of wordlists used during security assessments and penetration tests.

[84] **Key Distribution Centre (*KDC*):** Part of the Kerberos protocol that issues authentication and service tickets.

[85] **Access Control Entries (ACE):** Individual permission entries within an ACL that define access rights.

[86] **ntds.dit:** The main *Active Directory* database file containing all domain objects and user credentials.

[87] **Pass-The-Hack:** Technique that allows authentication using *NTLM* hashes without knowing the actual password.

[88] **LSASS:** Local Security Authority Subsystem Service; handles security policy and stores sensitive credentials in memory.

[89] **XML:** Extensible Markup Language used for structuring data in a readable and transferable format.

[90] **Rottan Potato:** A privilege escalation tool exploiting token impersonation and named pipe vulnerabilities in Windows.