

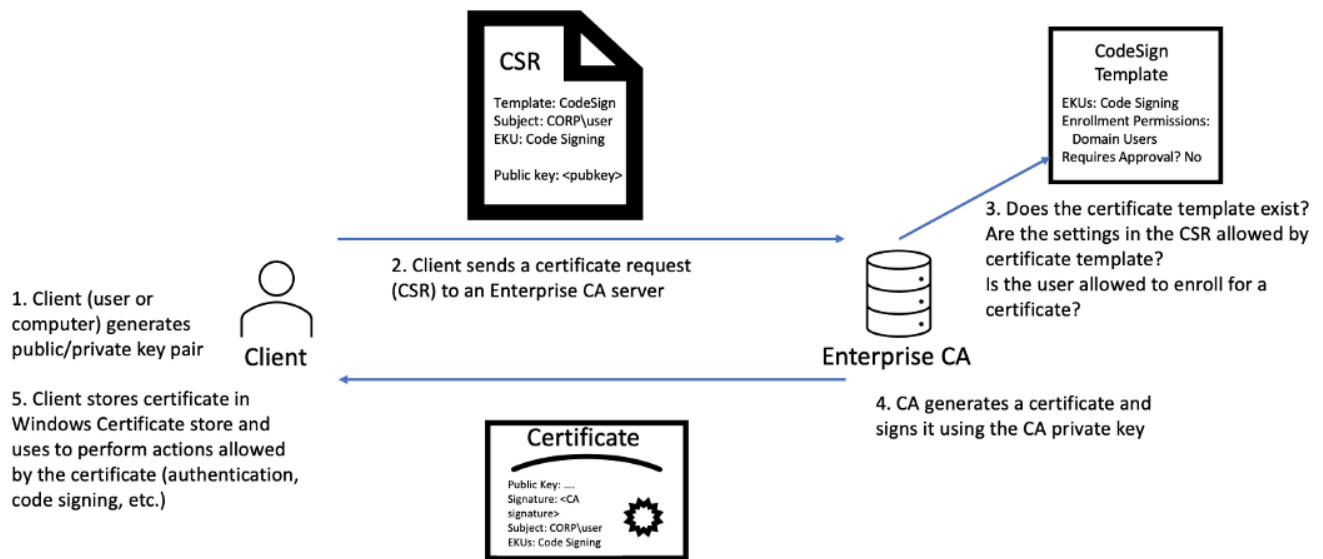
ADCS

Research done and released as a whitepaper by SpecterOps showed that it was possible to exploit misconfigured certificate templates for privilege escalation and lateral movement. Based on the severity of the misconfiguration, it could possibly allow any low-privileged user on the AD domain to escalate their privilege to that of an Enterprise Domain Admin with just a few clicks! [WhitePaper](#)

Windows Active Directory (AD) is not just for identity and access management but provides a significant amount of services to help you run and manage your organisation. A lot of these services are less commonly known or used, meaning they are often overlooked when security hardening is performed. One of these services is the Active Directory Certificate Services (AD CS).

When talking about certificates, we usually only think about the most common ones, such as those used to upgrade website traffic to HTTPS. But these are usually only used for applications that the organisation exposes to the internet. What about all those applications running on the internal network? Do we now have to give them internet access to allow them to request a certificate from a trusted Certificate Authority (CA)? Well, not really. Cue AD CS.

AD CS is Microsoft's Public Key Infrastructure (PKI) implementation. Since AD provides a level of trust in an organisation, it can be used as a CA to prove and delegate trust. AD CS is used for several things such as encrypting file systems, creating and verifying digital signatures, and even user authentication, which makes it a promising avenue for attackers. What makes it an even more dangerous attack vector, is that certificates can survive credential rotation, meaning even if a compromised account's password is reset, that would do nothing to invalidate the maliciously generated certificate, providing persistent credential theft for up to 10 years! The diagram below shows what the flow for certificate requests and generation looks like (taken from SpecterOps whitepaper):



Since AD CS is such a privileged function, it normally runs on selected domain controllers. Meaning normal users can't really interact with the service directly. On the other side of the coin, organisations tend to be too large to have an administrator create and distribute each certificate manually. This is where certificate templates come in. Administrators of AD CS can create several templates that can allow any user with the relevant permissions to request a certificate themselves. These templates have parameters that say which user can request the certificate and what is required. What SpecterOps has found, was that specific combinations of these parameters can be incredibly toxic and be abused for privilege escalation and persistent access!

Before we dive deeper into certificate abuse, some terminology:

- PKI - Public Key Infrastructure is a system that manages certificates and public key encryption
- AD CS - Active Directory Certificate Services is Microsoft's PKI implementation which usually runs on domain controllers
- CA - Certificate Authority is a PKI that issues certificates
- Certificate Template - a collection of settings and policies that defines how and when a certificate may be issued by a CA
- CSR - Certificate Signing Request is a message sent to a CA to request a signed certificate
- EKU - Extended/Enhanced Key Usage are object identifiers that define how a generated certificate may be used

The first step in this path is to enumerate all certificate templates to identify vulnerable ones and understand what is required to exploit them.

Luckily, Windows has some awesome built-in tools that can be used to enumerate all certificate templates and their associated policies. The most common approach is to use certutil. If we have access to a domain-joined computer and are authenticated to the domain,

we can execute the following command in a cmd window to enumerate all templates and store them in a file:

```
certutil -v -template > cert_templates.txt
```

You should get output like this in the textfile:

```
Template[1]:
TemplatePropCommonName = ClientAuth
TemplatePropFriendlyName = Authenticated Session
TemplatePropEKUs =
1 ObjectIds:
  1.3.6.1.5.5.7.3.2 Client Authentication

TemplatePropCryptoProviders =
  0: Microsoft Enhanced Cryptographic Provider v1.0
  1: Microsoft Base Cryptographic Provider v1.0
  2: Microsoft Base DSS Cryptographic Provider

TemplatePropMajorRevision = 3
TemplatePropDescription = User
TemplatePropSchemaVersion = 1
TemplatePropMinorRevision = 1
TemplatePropRASignatureCount = 0
TemplatePropMinimumKeySize = 800 (2048)
TemplatePropOID =
  1.3.6.1.4.1.311.21.8.13251815.15344444.12602244.3735211.11040971.202.1.4
```

The command will provide a bunch of output that may be difficult to read, but we are looking for some key indicators that will tell us that one of the templates is vulnerable. In the output, each template is denoted by `Template[X]` where X is the number. This can be used if you want to split the output from a single template.

The specific toxic parameter set that we are looking for is one that has the following:

- A template where we have the relevant permissions to request the certificate or where we have an account with those permissions
- A template that allows client authentication, meaning we can use it for Kerberos authentication
- A template that allows us to alter the subject alternative name (SAN)

Parameter 1: Relevant Permissions

We need to have the permissions to generate a certificate request in order for this exploit to work. We are essentially looking for a template where our user has either the **Allow Enroll** or **Allow Full Control** permission. You will probably never find a certificate where you have the *Allow Full Control* permission. However, if you do, congratulations! You can misconfigure the template yourself to make it vulnerable! But for now, let's focus on *Allow Enroll*.

It is not as simple as just grepping through the output for the keywords **Allow Enroll** and your AD account, since certificate template permissions are in most cases assigned to AD groups, not directly to AD users. So you will have to grep for all **Allow Enroll** keywords and

review the output to see if any of the returned groups match groups that your user belongs to. If you need to find your own groups, you can use this command:

```
net user <username> /domain
```

There are two groups that will be fairly common for certificates:

- Domain Users - This means in most cases that any authenticated users can request the certificate
- Domain Computers - This means that the machine account of a domain-joined host can request the certificate. If we have admin rights over any machine, we can request the certificate on behalf of the machine account

However, it is usually wise to review all certificate permissions, as it might point you in the direction of an account that will be in your reach to compromise.

Parameter 2: Client Authentication

Once we've shortened the list to certificate templates that we are allowed to request, the next step is to ensure that the certificate has the **Client Authentication** EKU. This EKU means that the certificate can be used for Kerberos authentication. There are other ways to exploit certificates, but for this room, this EKU will be the primary focus.

Therefore, for now, we are only interested in certificates that allow Client Authentication, meaning the certificate will be granted, given that we are the authenticated user on the machine requesting the certificate.

To find these, we need to review the EKU properties of the template and ensure that the words **Client Authentication** is provided. Other templates that do not match this, for now, we can discard.

Parameter 3: Client Specifies SAN

Last but definitely not least, we need to verify that the template allows us, the certificate client, to specify the Subject Alternative Name (SAN). The SAN is usually something like the URL of the website that we are looking to encrypt. For example: tryhackme.com. However, if we have the ability to control the SAN, we can leverage the certificate to actually generate a kerberos ticket for any AD account of our choosing!

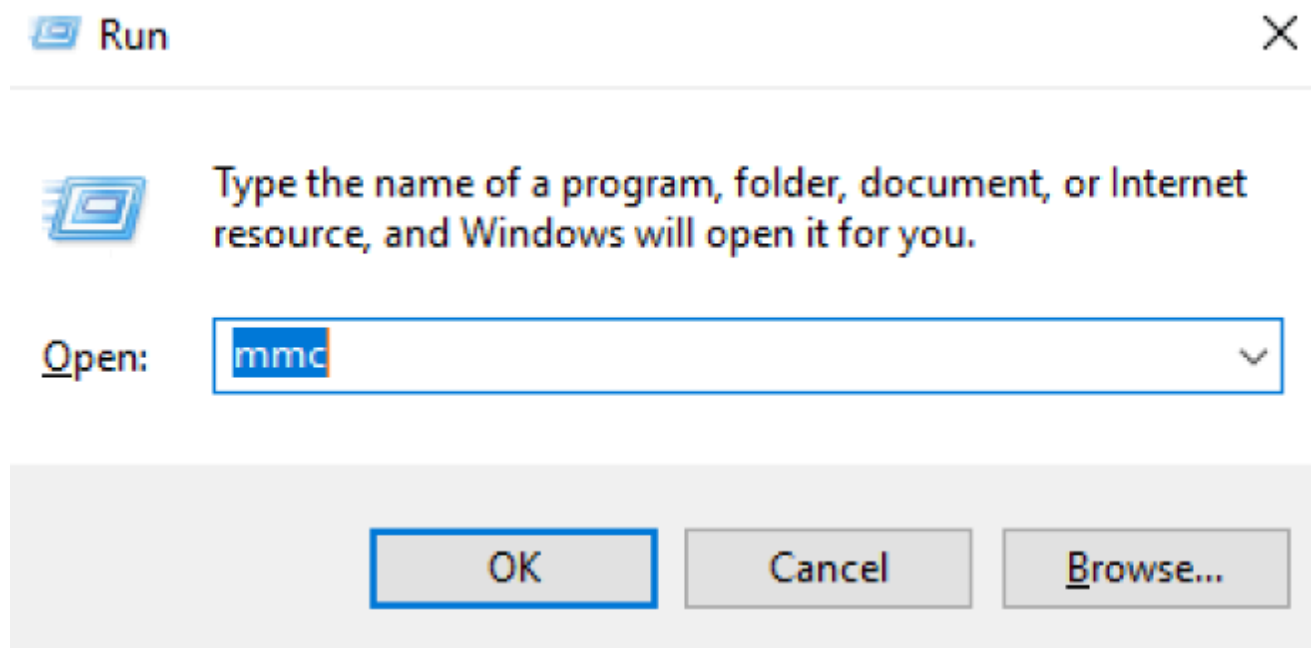
To find these templates, we grep for the **CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT** property flag that should be set to 1. This indicates that we can specify the SAN ourselves.

If we find a template where all three of these conditions are met, then we are in business and are a few clicks away from full Enterprise Admin rights! It should be noted that there are other conditions as well, like the fact that we want a certificate that does not go through an approval process to limit human intervention, but the full list of these parameters are not covered here simply because, by default, the initial certification template generation makes

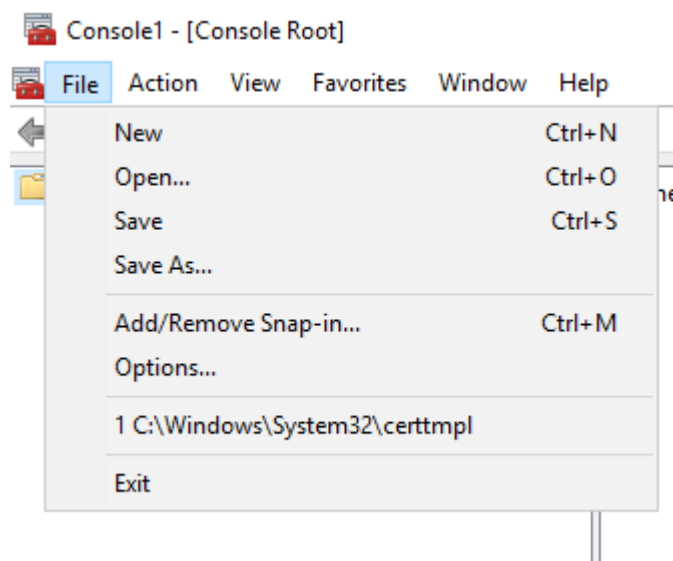
the template vulnerable. These additional template restrictions and EKUs are discussed at length in the whitepaper.

Now that we identified a certificate template that we can exploit, it is time to generate the certificate request. You could do this step from the command line if you wanted to, but why make your life difficult when Microsoft just allows you to do hacking through the click of some buttons!

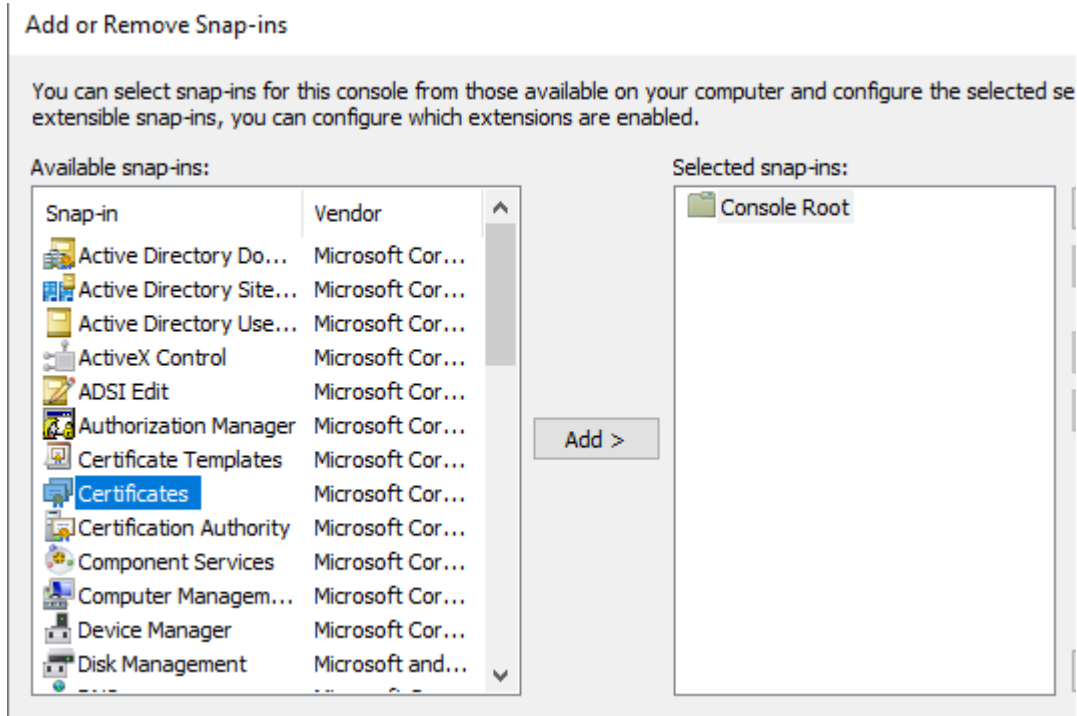
To request a new certificate, we will use the Microsoft Management Console. Load up the console by typing **mmc** in a run window:



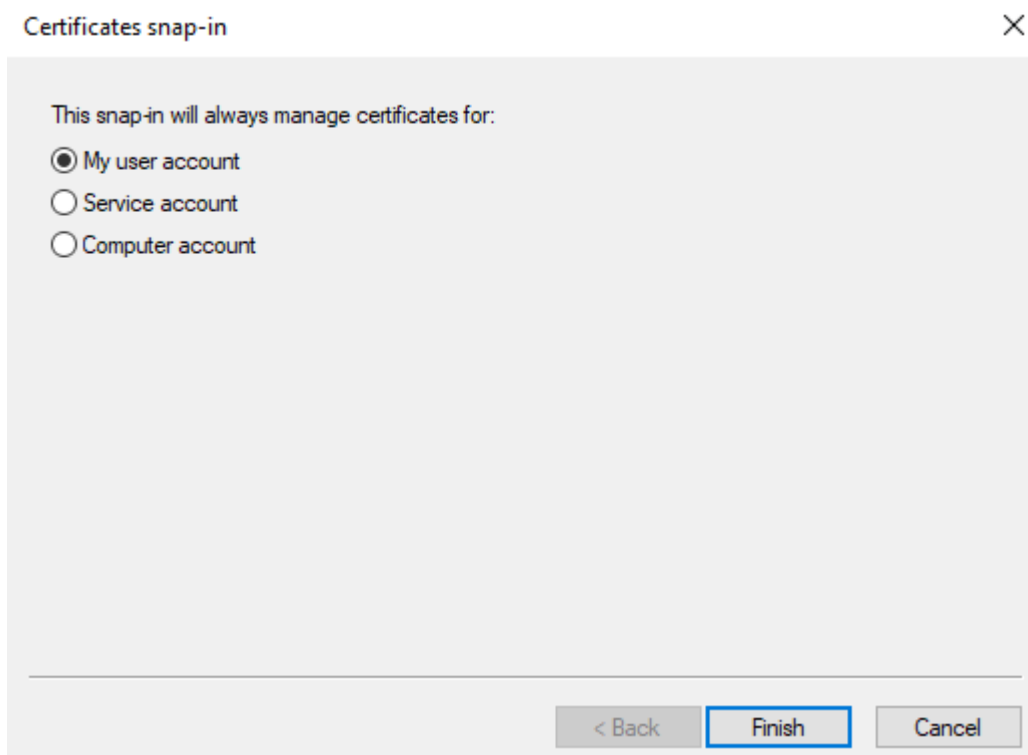
This will bring up the MMC window. In this window, click **File -> Add/Remove Snap-in...**



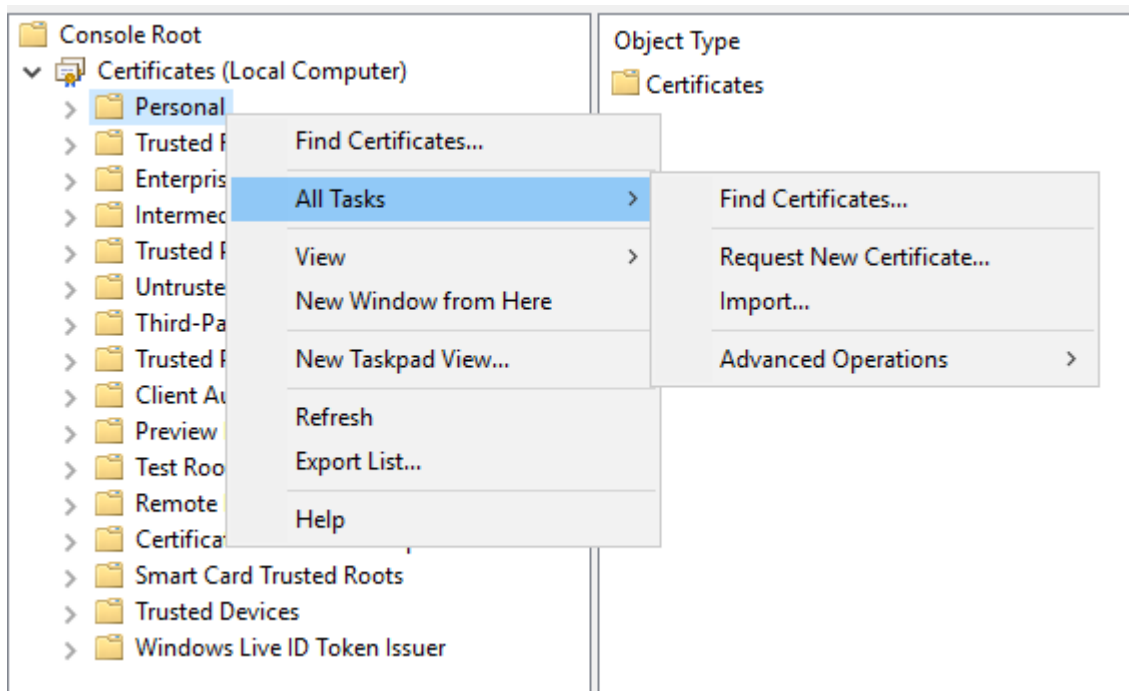
In this window, you want to add the **Certificates** snap-in:



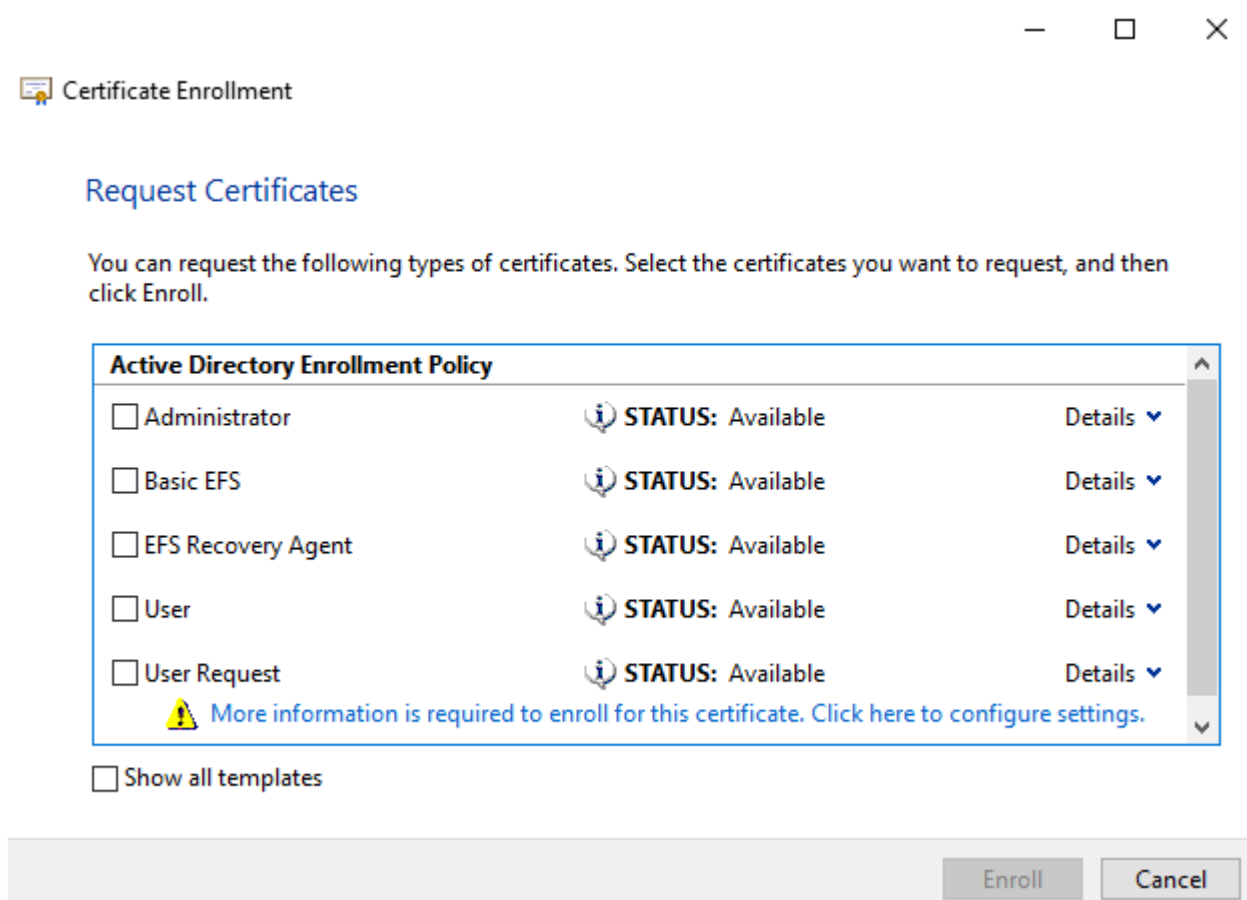
Although it will add the snap-in directly if you had administration privileges, you would see the next prompt:



This prompt allows you to impersonate service accounts or the machine account. But you can only perform these options if you have administration privileges on the host, which we currently don't have. You can now close the snap-in manager and return to the main console screen. Expand the **Certificates** option, right-click on **Personal**, select **All Tasks**, and click on **Request New Certificate**:



For the first option, just select **Next** twice, since we are using the default CA, you should see the following screen showing available templates:



As you can see from the screen, we need to complete the information for our certificate before we can enroll it. Click the "More information is required to enroll this certificate." link to start the process.

Certificate Properties ✕

Subject General Extensions Private Key Certification Authority Signature

The subject of a certificate is the user or computer to which the certificate is issued. You can enter information about the types of subject name and alternative name values that can be used in a certificate.

Subject of certificate
The user or computer that is receiving the certificate

Subject name:

Type:
Full DN Add >

Value:

< Remove

Alternative name:

Type:
Directory name Add >

Value:

< Remove

OK Cancel Apply

On this screen, we need to be very specific about the information that we provide. In order to ensure that the certificate can be exploited for a Kerberos ticket of a privileged user, we require the User Principal Name of the user we wish to impersonate. This can be found using the AD-RSAT tools or something like the PowerView scripts.

For this example, we will impersonate one of the DA users in this domain. Let's target the ***svc.gitlab*** account since it is a service account, meaning Kerberos authentication is likely expected from this account. The UPN of this account is svc.gitlab@lunar.eruca.com. Using this information we can complete the certificate properties.

First, we change the **Subject name** Type to **Common Name** and provide the name we want for this certificate. Then, we alter the **Alternative name** Type to **User principal name** and provide the UPN of the account we want to impersonate. The values should look like this:

Certificate Properties ✕

Subject General Extensions Private Key Certification Authority Signature

The subject of a certificate is the user or computer to which the certificate is issued. You can enter information about the types of subject name and alternative name values that can be used in a certificate.

Subject of certificate
The user or computer that is receiving the certificate

Subject name:

Type:
Common name ▾

Add >

Value:
vulncert

< Remove

Alternative name:

Type:
User principal name ▾

Add >

Value:
svc.gitlab@lunar.eruca.com

< Remove

OK Cancel Apply

We can then add these properties to our certificate:

Certificate Properties ✕

Subject General Extensions Private Key Certification Authority Signature

The subject of a certificate is the user or computer to which the certificate is issued. You can enter information about the types of subject name and alternative name values that can be used in a certificate.

Subject of certificate
The user or computer that is receiving the certificate

Subject name:

Type: Common name Add >

Value: < Remove

CN=vulncert

Alternative name:

Type: User principal name Add >

Value: < Remove

User principal name
svc.gitlab@lunar.eruca.com






OK Cancel Apply

When we click okay, we will now see that we are allowed to enroll this certificate:

Certificate Enrollment

Request Certificates

You can request the following types of certificates. Select the certificates you want to request, and then click Enroll.

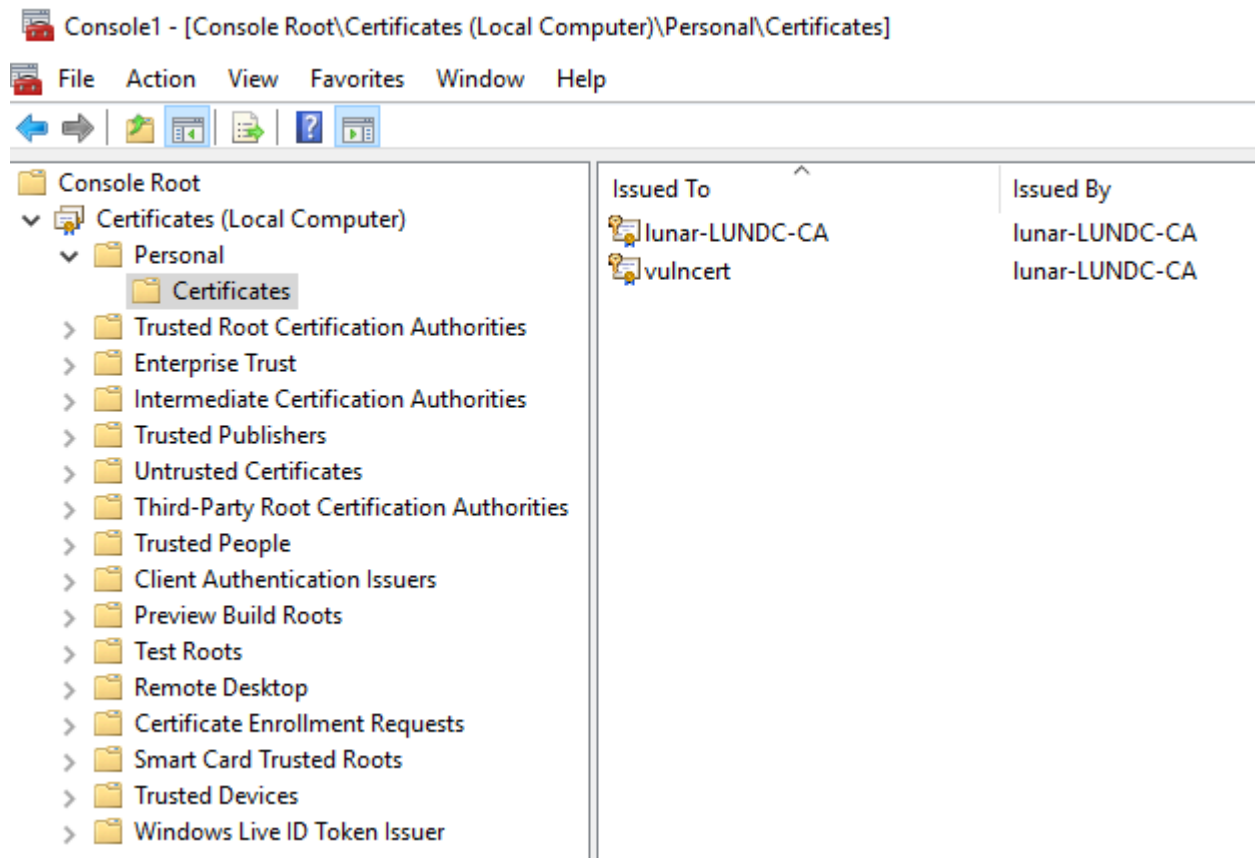
Active Directory Enrollment Policy		
<input type="checkbox"/> Administrator	 STATUS: Available	Details ▼
<input type="checkbox"/> Basic EFS	 STATUS: Available	Details ▼
<input type="checkbox"/> EFS Recovery Agent	 STATUS: Available	Details ▼
<input type="checkbox"/> User	 STATUS: Available	Details ▼
<input checked="" type="checkbox"/> User Request	 STATUS: Available	Details ▼

☐ Show all templates

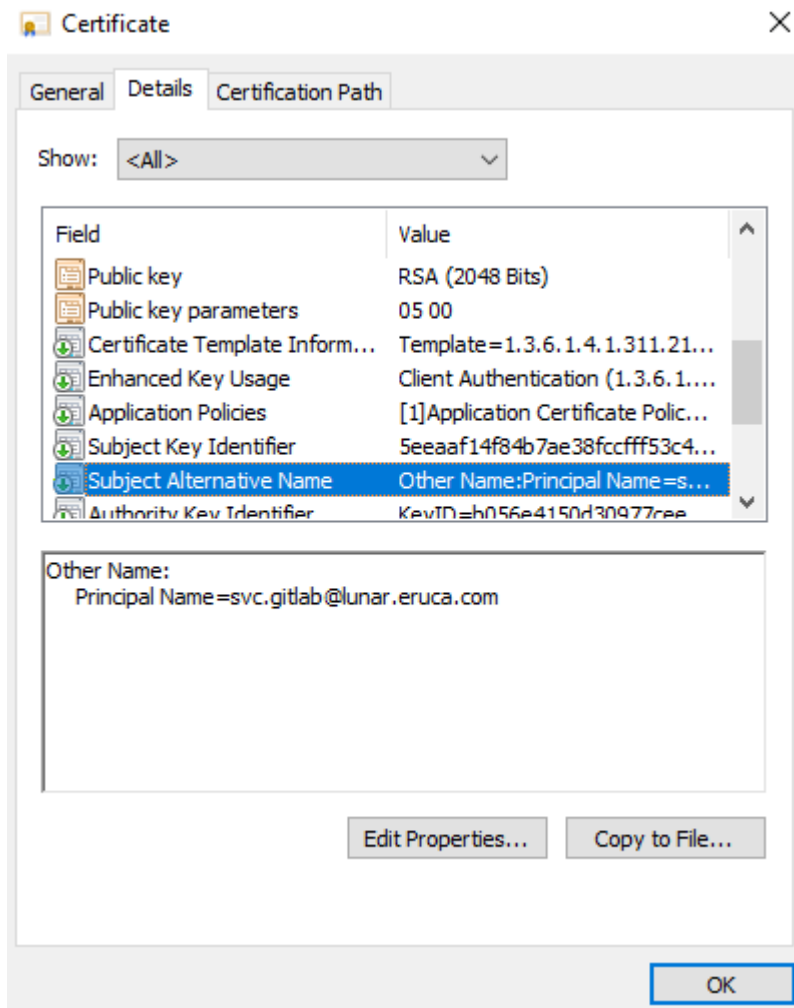
Enroll

Cancel

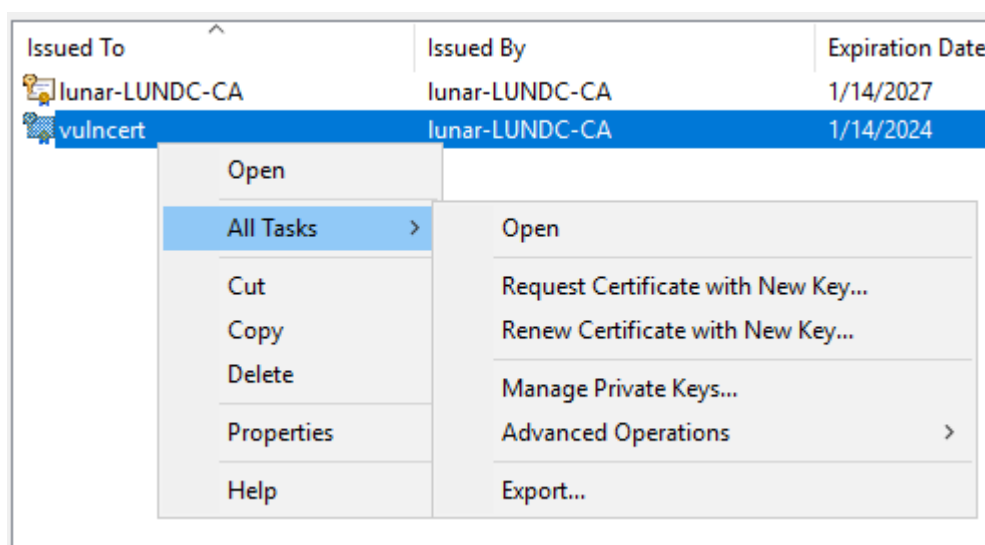
Complete the enrollment of the certificate. Once enrolled you will be able to view the certificate under your personal certificates:



If we review the details of the certificate, you will see that the SAN now specifies the UPN we want to impersonate, definitely not the SAN of our web server!



Congratulations, you have just generated a certificate that will provide you with persistent authentication to the domain for the next two years! The last step is to actually export the certificate to get it ready for use. Right-click on the certificate, select **All Tasks**, and then **Export**:



Follow the prompts but make sure to export the private key as well:

Export Private Key


You can choose to export the private key with the certificate.

Private keys are password protected. If you want to export the private key with the certificate, you must type a password on a later page.

Do you want to export the private key with the certificate?

- ☒ Yes, export the private key
- ☐ No, do not export the private key

The certificate should be in pfx format. Furthermore, configure a password for the certificate to ensure the private key is exported:

←  Certificate Export Wizard

Security

To maintain security, you must protect the private key to a security principal or by using a password.

☐ Group or user names (recommended)

Add

Remove

☒ Password:

.....


Confirm password:

.....

Encryption: TripleDES-SHA1

Next Cancel

Select a filename and export the certificate:

Name	Date modified	Type	Size
 vulncert.pfx	1/14/2022 8:45 AM	Personal Informati...	5 KB

Now your certificate is ready for exploitation!

Now we can finally impersonate a user. To perform this, two steps are required:

- Use the certificate to request a Kerberos ticket granting ticket (TGT)
- Load the Kerberos TGT into your hacking platform of choice

For the first step, we will be using [Rubeus](#). An already compiled version is available in the `C:\users\kali\desktop\adcs` directory. Open a command prompt window and navigate to this directory. We will use the following command to request the TGT:

```
Rubeus.exe asktgt /user:svc.gitlab /enctype:aes256 /certificate:<path to certificate> /password:<certificate file password> /outfile:<name of file to write TGT to> /domain:lunar.eruca.com /dc:<IP of domain controller>
```

Let's break down the parameters:

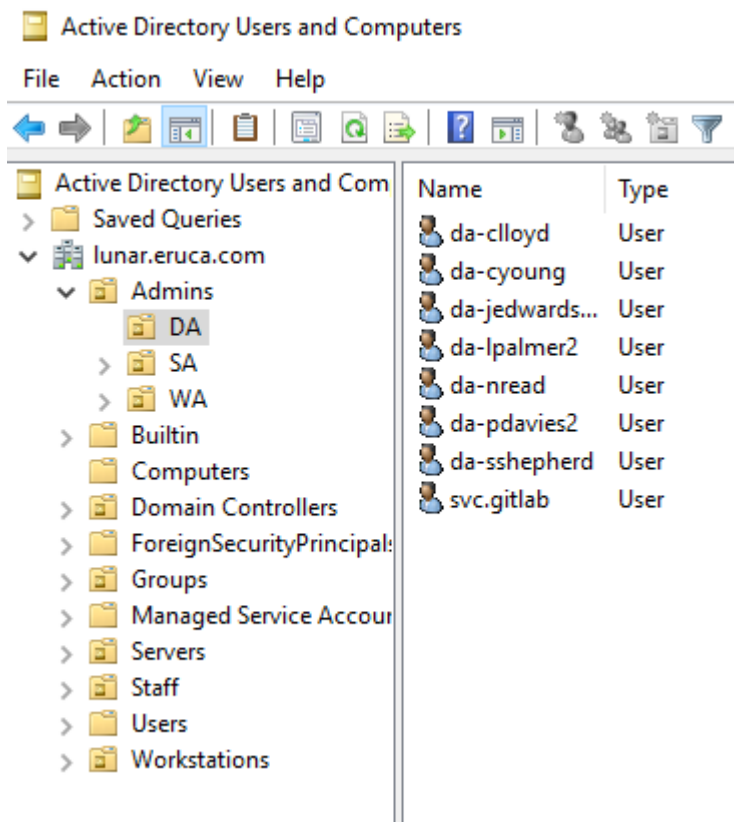
- **/user** - This specifies the user that we will impersonate and has to match the UPN for the certificate we generated
- **/enctype** - This specifies the encryption type for the ticket. Setting this is important for evasion, since the default encryption algorithm is weak, which would result in an overpass-the-hash alert
- **/certificate** - Path to the certificate we have generated
- **/password** - The password for our certificate file
- **/outfile** - The file where our TGT will be output to
- **/domain** - The FQDN of the domain we are currently attacking
- **/dc** - The IP of the domain controller where we are requesting the TGT from. Usually it is best to select a DC that has a CA service running

Once we execute the command, we should receive our TGT:

TGT Request for example

```
.\Rubeus.exe asktgt /user:svc.gitlab /enctype:aes256 /certificate:vulncert.pfx /password:tryhackme /outfile:svc.gitlab.kirbi /domain:lunar.eruca.com /dc:10.10.69.219
```

We now need to use this TGT to gain access. This can be done using your favorite hacking framework like metasploit, cobaltstrike, or covenant. However, for the purpose of this walkthrough, we will be using Rubeus again. We will use the ticket to alter the password of one of the domain administrators. This will allow us to use the DA's credentials to log into the Domain Controller as administrator to recover the final flag. Open the **Active Directory Users and Computers** application and explore the domain structure, looking for the DAs:



Select one of the DAs to target and use the following command to alter their password:

```
Rubeus.exe changepw /ticket:<path to ticket file> /new:<new password for user>
/dc:LUNDC.lunar.eruca.com /targetuser:lunar.eruca.com\<username of targeted DA>
```

Once the command executes, it should alter the password for the associated DA account:

Resetting a user's password

```
.\Rubeus.exe changepw /ticket:svc.gitlab.kirbi /new:Tryhackme!
/dc:LUNDC.lunar.eruca.com /targetuser:lunar.eruca.com\da-nread
```

You can now authenticate as this user to the Domain Controller and recover the final flag. Well done! You have now compromised DA! As an added bonus, let's look at the `runas` command, which we can use to authenticate as another user in the command prompt. We can use the following command in cmd to authenticate as the now compromised DA user:

```
runas /user:lunar.eruca.com\<username of DA> cmd.exe
```

You will be prompted to provide the password for the associated account. If correct, `runas` will spawn a command prompt window for you as the specified user, which you can now use for administrative duties.

So how can you actually defend against these Certificate Template attacks?

There isn't really an easy answer, but there are some good defense techniques:

- Review all the certificate templates in your organisation for poisonous parameter combinations. You can use [PSPKIAudit](#) to assist with this.
- In cases where poisonous parameter combinations cannot be avoided, make sure that there are stopgaps such as having to require Admin Approval before the certificate will be issued.
- Update your playbooks. Most organisations' playbooks will include something along the lines of resetting the credentials for a compromised account. As pointed out here, that's not really going to remedy the persistent access. Therefore, reviewing certificates that were issued would be required and the malicious certificate will have to be revoked.

Inject ticket with [Rubeus](#):

```
.\Rubeus.exe ptt /ticket:<ticket_kirbi_file>
```

Execute a cmd in the remote machine with [PsExec](#):

```
.\PsExec.exe -accepteula \\<remote_hostname> cmd
```