**Cascade**

Difficulty: Medium

OS: Windows

**Nmap**

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

|  |
| --- |

**SMB / RPC Enumeration**

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

| *crackmapexec smb 10.10.10.182 --user* |
| --- |

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

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

| *cat ldapsearch.txt | awk '{print $1}' | sort | uniq -c | sort -nr*  Print the first item, then sort, then group the output by the number of times it shows up, finally sort again by number from smallest to largest and reverse so we see the least amount at the bottom as in the screenshot above |
| --- |

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

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

| **rY4n5eva** |
| --- |

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

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

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

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

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

|  |
| --- |

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

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

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

| *echo 6bcf2a4b6e5aca0f | xxd -p -r* |
| --- |

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

<https://github.com/frizb/PasswordDecrypts>

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

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

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

Mounting this share, we find a couple interesting files.

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

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

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

**User : s.smith**

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

|  |
| --- |

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

| *Whoami /all*  *Net user s.smith /domain* |
| --- |

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

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

| *Sqlite3 Audit.db*  *.tables*  *SELECT \* FROM Ldap* |
| --- |

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

|  |
| --- |

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

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

|  |
| --- |

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

|  |
| --- |

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

|  |
| --- |

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

| *w3lc0meFr31nd\0\0\0* |
| --- |

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

|  |
| --- |

**User: arksvc**

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

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

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

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

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

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

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

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

|  |
| --- |

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

|  |
| --- |